

UNIVERSIDAD NACIONAL
DEL NORDESTE



FACULTAD DE INGENIERÍA

PROYECTO FINAL DE CARRERA

TEMA:

DISEÑO DE UN POSICIONADOR AUTOMÁTICO
DE INSTRUMENTOS DE MEDICIÓN PARA EL
TÚNEL DE VIENTO PROF. JACEK GORECKI.

ALUMNO:

OBREGÓN MARCELO JOAQUIN

CARRERA:

INGENIERÍA ELECTROMECÁNICA

2022



ÍNDICE

	Pag.
1. Resumen	iV
2. Introducción	Vi
1RA PARTE – DISEÑO MECÁNICO	
3. Descripción y resumen del diseño mecánico	002
4. Selección de motores y sistema de transmisión	009
4.1. Selección del tipo de transmisión	009
4.2. Selección del tipo de correas	010
4.3. Motores	010
4.3.1. Selección del tipo de motores	010
4.3.2. Características constructivas de los motores paso a paso	011
4.3.3. Características dinámicas de los motores paso a paso	014
4.3.4. Drivers para motores paso a paso	016
4.3.5. Dimensionamiento de los motores	017
4.3.5.1. Torque constante de la carga	017
4.3.5.2. Torque inercial	019
4.4. Perfil de velocidades	020
4.5. Reductor de velocidad	024
4.6. Procedimiento para la selección de los elementos del sistema de transmisión	024
4.7. Avance por paso, error de avance y tiempo de desplazamiento	025
4.8. Eje vertical - Dimensionamiento del motor y la transmisión	027
4.9. Eje horizontal - Dimensionamiento de los motores y las transmisiones	041
4.10. Rodamientos	053
4.11. Resumen de las transmisiones	057
5. Dimensionamiento de ejes guía	058
5.1. Procedimiento para el dimensionamiento de los ejes	058
5.2. Ejes verticales	061
5.2.1. Determinación de la sección por deformación	061
5.2.2. Verificación de la resistencia mecánica	064
5.3. Ejes Horizontales	065
5.3.1. Determinación de la sección por deformación	065
5.3.2. Verificación de la resistencia mecánica	069
6. Ajuste y tolerancia de ejes y bujes	071
7. Distribución de carga del eje vertical sobre los ejes horizontales	072
8. Accesorios	074
8.1. Cadena porta cables	074
8.2. Acoplamiento entre ejes y reductores horizontales	075
8.3. Mangueras de PVC	076
8.4. Instrumento de medición	076
8.5. Carro y riel porta carro horizontal superior	077



2DA PARTE – SISTEMA DE CONTROL

9. Descripción del sistema de control	080
10. Arduino	081
11. Sistemas de referencia	083
11.1. Variables nvg y nhg	083
11.2. Sistema de referencia inicial	083
11.3. Sistema de referencia global	084
11.4. Cambio de sistema de referencia inicial a global	084
11.5. Sistema de referencia relativo	085
12. Protección contra colisiones	086
12.1. Protección mediante la programación	086
12.2. Protección mediante fines de carrera de protección	087
13. Variables de ajuste.	088
13.1. Descripción	088
13.2. Determinación de las variables de ajuste	088
13.3. Almacenamiento de las variables de ajuste	090
14. Descripción de las funciones	092
15. Programación del sistema de control - Arduino	097
15.1. Diagrama de flujo resumido de la programación.	097
15.2. Descripción del funcionamiento de los procesos del diagrama de flujo resumido.	124

3RA PARTE – INTERFACES

16. Tipos de interfaces del posicionador	151
17. Aplicación de LabVIEW	153
17.1. LabVIEW	153
17.2. Instalación de la aplicación de LabVIEW	155
17.3. Descripción de la pantalla	155
17.4. Manejo de la aplicación de LabVIEW	157
18. APP para celular	166
18.1. App Inventor 2	166
18.2. Instalación de la APP para celular	167
18.3. Descripción de la pantalla	167
18.4. Manejo de la APP para celular	168
19. Botonera	171
19.1. Descripción de la botonera	171
19.2. Manejo de la botonera	172
20. Comunicación entre Arduino y las Interfaces	174
20.1. Desde las Interfaces a Arduino	174
20.1.1. Ordenes	174
20.1.2. Datos	176
20.2. Desde Arduino a la interfaces	177
20.2.1. Confirmaciones e indicaciones	178
20.2.2. Datos	180



21. Duración del envío de cadenas de información	184
21.1. Características de la función Serial.print()	185
21.2. Velocidad de transmisión de bytes de información y duración de ciclos de LabVIEW	185
21.3. Duración de transmisión de cadenas	186

4TA PARTE – ELECTRICIDAD Y ELECTRÓNICA

22. Drivers, Fuentes de alimentación y Cableado	191
22.1. Drivers	191
22.2. Fuentes de alimentación	192
22.3. Cableado	193
23. Ubicación del punto de referencia	197
23.1. Sensores de posición	197
23.2. Tipo de conexión	197
24. Protecciones contra colisiones mediante fines de carrera y contra desenergizado de drivers	200
24.1. Protección contra colisiones mediante fines de carrera	200
24.2. Protección contra desenergizado de drivers	200
24.3. Contactores de alimentación de drivers	201

5TA PARTE – OTROS

25. Análisis económico	203
25.1. Computo de materiales	203
25.2. Proveedores	207
26. Posibles mejoras	209
27. Fuentes consultadas	211

ANEXOS

A. Catálogos de componentes utilizados	A-212
B. Circuitos eléctricos y electrónicos	B-239
C. Programación del Arduino	C-247
D. Planos constructivos	D-298



1. RESUMEN

El presente proyecto final de carrera consiste en el diseño íntegro de un posicionador automático de instrumentos de medición para el túnel de viento Prof. Jacek Gorecki perteneciente al Laboratorio de Aerodinámica de la Facultad de Ingeniería de la Universidad Nacional del Nordeste. El mismo permitirá desplazar el instrumento de medición de forma automática y precisa en un plano transversal al túnel, estando éste en funcionamiento. De esta manera se reduciría de forma considerable el tiempo requerido para la realización de ensayos y se aumentaría significativamente la precisión para ubicar el instrumento en las posiciones deseadas, mejorando así el desempeño de las investigaciones realizadas.

El posicionador propuesto consiste básicamente en una máquina de control numérico computarizado (CNC) en dos direcciones sobre un plano transversal al túnel, el cual podrá ser operado mediante una aplicación para PC, una APP para celulares Android o una botonera analógica.

Este proyecto abarca el diseño estructural, diseño y selección de los componentes del sistema de fuerza motriz, diseño y selección de los circuitos y componentes electrónicos, y la programación del sistema de control con sus diferentes interfaces HMI.

Para ubicar el instrumento de medición en las posiciones deseadas el posicionador contará con diferentes funciones. La función principal permitirá realizar el muestreo de múltiples puntos de forma automática siguiendo una cuadrícula definida por el operador, donde éste indicará la cantidad de mediciones en cada dirección, la distancia entre ellas y los tiempos de muestreo. Las otras funciones permitirán posicionar el instrumento en los puntos deseados ingresando sus coordenadas, o bien, desplazando el instrumento de forma continua o discreta hasta los mismos. Se contará también con dos memorias para guardar posiciones de interés y poder volver a estas fácilmente cuando se desee.

El operador manejará las distintas funciones mediante tres tipos de interfaces. La interfaz principal será una aplicación para computadora desarrollada mediante el software LabVIEW desde donde se tendrá el manejo completo del sistema. Como interfaces secundarias se tendrá una APP para celulares Android desarrollada a través de la plataforma App Inventor 2 y una botonera, desde las cuales se podrá acceder ágilmente aunque solo permite emplear algunas de las funciones.

Para el accionamiento de sistema se utilizarán motores paso a paso con reductores de velocidad planetarios, convirtiendo luego su movimiento de rotación en traslación mediante correas síncronas. La combinación de los distintos elementos de la transmisión permitirá desplazamientos verticales y horizontales de 0,06 y 0,044 mm/paso respectivamente. Por su parte la precisión global del sistema será de 0,2 mm en ambas direcciones, estando esto determinado por la precisión de los sensores de posición.

El sistema de control del posicionador se realizará mediante una placa Arduino (placa electrónica con un microcontrolador programable). Esta placa será el corazón del sistema de control y la misma contendrá la programación de las diferentes funciones, se comunicará con las diferentes HMI, controlará los drivers de los motores paso a paso y monitoreará los sensores de posición.

El posicionador podrá también desplazarse en sentido axial al túnel de viento. En este caso el desplazamiento se realizará de forma manual debiendo ingresar el operador dentro del túnel.

En el diseño de la estructura se tomó como principal criterio de dimensionamiento a la deformación de la misma, ya que esto afectaría directamente a la precisión en el posicionamiento de los instrumentos de medición. Este diseño se enfocó también en la reducción del espacio ocupado por el posicionador, maximizando así la zona de trabajo y desplazamiento del instrumento de medición. A su vez, se dio importancia especial a la simplicidad de montar y desmontar el posicionador dentro del túnel, reduciendo así el tiempo requerido para estas tareas y disminuyendo la posibilidad de cometer errores. Por esto último el diseño de la estructura del posicionador está reducido a sólo tres cuerpos unidos por cinco bulones.



Por su parte el sistema de control buscó ser lo más preciso y seguro posible, protegiendo al sistema, previniendo fallas y dando aviso de ellas en caso que ocurrieran. Se buscó también que las interfaces sean intuitivas y amigables con el operador, minimizar en lo posible su intervención y reduciendo así la posibilidad de cometer errores en su operación.

En general se prestó también especial importancia a la calidad, economía y facilidad para conseguir los elementos necesarios para su construcción, tanto de la parte estructural como la de control.



2. INTRODUCCIÓN

Gracias a las nuevas tecnologías constructivas, métodos de cálculos más precisos, etc, las estructuras civiles son cada vez más livianas, esbeltas y abarcan mayores luces, lo que trae consigo una menor rigidez y menor amortiguamiento estructural. Las fuerzas provocadas por la interacción entre el viento y la estructura pueden generar fenómenos de inestabilidad dinámica, pudiendo provocar el colapso de la estructura. Es entonces de suma importancia profundizar el conocimiento sobre inestabilidad dinámica en este tipo de estructuras civiles. Para ello, un medio confiable y económico es a través de técnicas de análisis experimental, las cuales consisten en construir un modelo a escala reducida, reproduciendo los parámetros estructurales, distribución de masa, amortiguamiento y frecuencia natural de vibración, y ensayarlos luego en un túnel de viento. Las mediciones realizadas sobre el mismo se relacionan luego con el modelo real mediante los coeficientes correspondientes.

Actualmente en el túnel de viento Prof. Jacek Gorecki, perteneciente al Laboratorio de Aerodinámica de la Facultad de Ingeniería de la Universidad Nacional del Nordeste, se llevan a cabo proyectos de investigación acreditados y consultorías. Los ensayos normalmente realizados a los modelos son de dos tipos: 1) Colocación de células de cargas sobre los modelos para la medición de presión en estos; 2) Medición de los perfiles de viento deformados en diferentes planos transversales al flujo, tanto aguas abajo como aguas arriba del modelo.

Para el segundo tipo de ensayo es necesario realizar mediciones en numerosos puntos próximos al modelo. Estas mediciones se realizan con un único instrumento, por lo tanto, éste debe desplazarse a cada punto donde se desee realizar una medición. A su vez, estos instrumentos son frágiles y costosos, y cualquier contacto con el modelo durante su desplazamiento podría dañarlos. Para este tipo de mediciones el túnel de viento cuenta con un posicionador de instrumentos de medición con desplazamiento únicamente en dirección vertical y que abarca solo la mitad de la altura total del túnel. Por ello, realizar mediciones en un plano transversal del túnel significa el constante movimiento de todo el posicionador, tanto su rotación para abarcar la altura total del túnel como su posterior desplazamiento horizontal para cubrir toda la superficie transversal del mismo.

Cada rotación o desplazamiento del posicionador implica ingresar al túnel, con lo cual estas tareas requieren un tiempo considerable del orden de los 20 min. En consecuencia, cuando un ensayo requiere gran cantidad de puntos de medición, esta práctica lleva muchas horas de trabajo y en ocasiones puede extenderse por varios días. La larga duración en este tipo de ensayos trae consigo diferentes inconvenientes.

En primer lugar, dado que el túnel de viento es de tipo abierto, la larga duración de los ensayos impide tener variables controladas durante el mismo, ya que se pueden producir variaciones de temperatura ambiente de más de 10 °C y en algunos casos incluso variaciones de humedad o presión atmosférica, los cuales afectan directamente a la densidad del aire y por tanto a su velocidad y presión dinámica. Por su parte la calibración de los sensores también son afectados por la variación de temperatura, por lo cual éste parámetro genera un doble error con su variación. Estos errores generados en las mediciones actualmente se reducen de forma parcial corrigiendo por temperatura los valores medidos. Para ello se calculan las temperaturas a lo largo del ensayo mediante interpolación lineal, utilizando como referencia la temperatura al inicio y final de cada ensayo. Con este método se pierde mucha información sobre la variación real de temperatura durante el ensayo y, si bien ayuda a reducir los errores, los mismos pueden seguir siendo considerables si la variación de temperatura es importante o el ensayo se extiende por varios días.

Otro aspecto importante al considerar el uso del posicionador actual es la precisión en el posicionamiento del instrumento. Debido a que el desplazamiento transversal del posicionador se realiza a mano, la exactitud de las coordenadas horizontales de los puntos de medición depende de la precisión al ubicar y nivelar el posicionador en el lugar deseado, tarea que, aun realizándose con mucho cuidado, es

afectada por la precisión de las herramientas utilizadas y las limitaciones propias de las personas que realizan el montaje.

Por otro lado, en este posicionador ya existente, el desplazamiento del instrumento se realiza mediante una varilla roscada acoplada a un motor eléctrico accionado por pulsadores. De esta manera el instrumento avanza en una u otra dirección de forma constante mientras el pulsador correspondiente se encuentre accionado, siendo su posición determinada mediante una regla ubicada junto al mismo. Por lo tanto, la determinación de la coordenada vertical del instrumento se ve limitada por la precisión de la visión del operador.

A su vez, la ardua tarea de desplazar numerosas veces el posicionador a las coordenadas deseadas y nivelarlo de forma manual trae consigo el aumento en la posibilidad de cometer errores diversos, a veces no detectados, y que pueden influir en las mediciones realizadas.



Se debe tener muy en cuenta también el costo económico que implican los largos periodos de realización de los ensayos, ya que esto impacta directamente en el consumo energético, teniendo en cuenta que el motor del túnel de viento es de 90 KW y permanece encendido durante todo el ensayo. Además del gasto en energía debe considerarse también el desgaste y mantenimiento del motor y el sistema eléctrico que lo alienta, los cuales son sumamente costosos y consumen su tiempo de vida funcionando por largos periodos donde la mayor parte del tiempo su uso no es productivo.

En esta misma línea debe tenerse en cuenta también que los largos ensayos reducen la disponibilidad del túnel y con ello su productividad, imposibilitando la realización de otros ensayos durante ese tiempo, los cuales, en caso de consultorías, aportan un importante ingreso para el laboratorio.

Por todo lo anterior, estos segundos tipos de ensayos para medición de los perfiles de viento generalmente no se realizan y solo se realizan los ensayos con células de carga sobre los modelos, con lo cual se pierde información importante para complementar ambos tipos de ensayos.

Poder desarrollar el máximo potencial del túnel de viento y de los proyectos de investigación realizados con él, depende en gran medida del equipo de laboratorio disponible. Por ello contar con un mecanismo operado de forma automática y remota que permita el posicionamiento rápido y preciso de los instrumentos de medición, reduciría notablemente la duración de los ensayos y permitiría que estos se realicen con variables controladas, con lo cual se mejoraría la calidad estadística, regularidad y seguridad de las mediciones del túnel del viento, lo cual mejoraría también a su vez la calidad de los proyectos de investigación realizados en él.

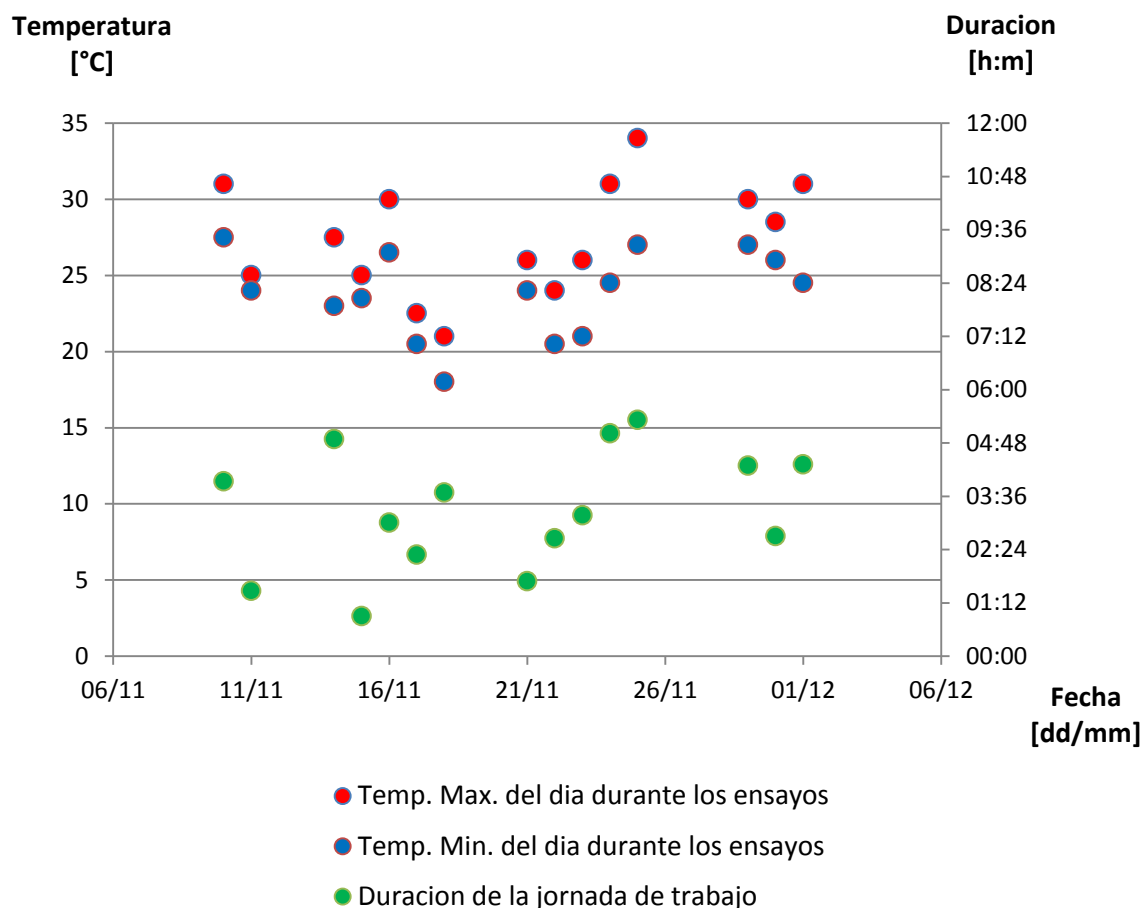
A su vez, los trabajos de consultoría podrían realizarse de forma más rápida y se obtendrían resultados más precisos. Esto implicaría un mayor ingreso económico por este tipo de trabajos ya que, además de poder realizar mayor cantidad de ensayos en un mismo tiempo, también aumentaría el valor de estos por realizarse ensayos de mejor calidad.

Por otra parte este mecanismo reduciría también enormemente el consumo de energía del túnel durante los ensayos, reduciría los costos de mantenimiento del mismo y prolongaría la vida útil de sus distintos componentes.

Con este objetivo se realizó el diseño del posicionador automático del presente proyecto final de carrera, el cual permitirá reducir el tiempo de realización de este tipo de ensayos de horas o días a solamente minutos y con una gran precisión en el posicionamiento.

A modo de comparación se podría poner como referencia el ensayo en vacío del túnel de viento realizado a finales del año 2016. En este ensayo se muestrearon 165 puntos a una velocidad V1, 165 puntos a una velocidad V2, 385 puntos a una velocidad V3 y 385 puntos a una velocidad V4. Se realizó así un total de 1100 muestreos con una duración promedio de 30 seg. cada uno y el ensayo se extendió durante 15 días no consecutivos entre el 10 de noviembre y el 01 de diciembre. La larga duración de los ensayos fue acompañada de importantes variaciones en la temperatura durante su desarrollo.

En la siguiente grafica puede apreciarse la duración total en horas de cada jornada de trabajo y la temperatura máxima y mínima registrada en esos lapsos de tiempo.



El tiempo de trabajo total destinado al ensayo fue de aproximadamente 29 hs desde el muestreo del primer punto hasta el muestreo del último. Durante toda la extensión del ensayo se produjo una variación máxima de temperatura de 16 °C entre la temperatura máxima de 34 °C del día 25/11 y la mínima de 18 °C el día 18/08. La mayor variación de temperatura en un mismo día fue de 7 °C el día 25/11, coincidente con el día de mayor duración de los ensayos.

De las 29 hs del ensayo, solamente 9 hs fueron destinadas al muestreo de los puntos de interés, las demás 20 hs fueron destinadas al posicionamiento del instrumento y otros contratiempos. De estas 20 hs destinadas al posicionamiento, el túnel se encontró encendido aproximadamente durante 11 hs por lo cual, a lo largo de todo el ensayo, el túnel se encontró en funcionamiento al menos 20 hs.



Si el mismo ensayo se llevara a cabo con el posicionador diseñado en este proyecto, el mismo se realizaría en aprox. 12 hs, por lo que el tiempo total del ensayo se reduciría en casi un 60 %. Esta importante reducción de tiempo permitiría realizar todo el ensayo en dos o tres días, con lo cual se reduciría considerablemente las variaciones de temperatura durante el mismo, permitiendo tener mayor control de las variables en el ensayo y con ello mayor exactitud en los resultados.

Por su parte, teniendo en cuenta que las 9 hs necesarias para el muestreo no se modificarían, el ahorro de tiempo se deberá entonces a la disminución del tiempo de posicionamiento, el cual se reduciría de 20 hs a solo 3 hs, lo que representaría un ahorro del 85% en tiempo total de posicionamiento. A su vez el tiempo total que el túnel se encontraría encendido se reduciría de 20 hs a 12 hs, lo que representaría una disminución del 40 % en desgaste mecánico y un ahorro de energía de aprox. 720 KW-h.





1RA PARTE

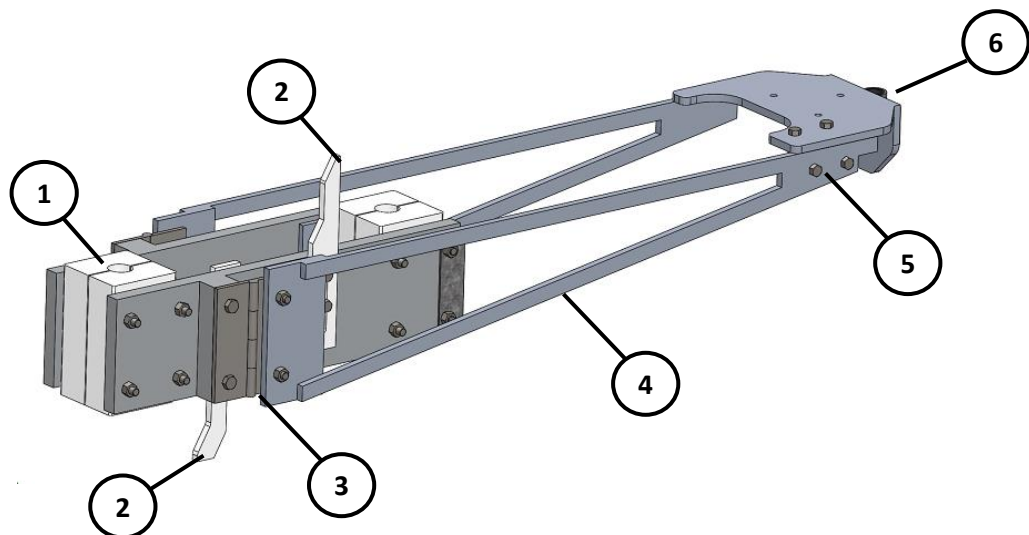
DISEÑO MECÁNICO

3. DESCRIPCIÓN Y RESUMEN DEL DISEÑO MECÁNICO

El posicionador tendrá desplazamiento automático en el plano transversal al túnel de viento, y movimiento manual en dirección axial. El túnel de viento posee una sección transversal de 1600 x 2400 mm, mientras que el posicionador ocupará toda esta sección transversal y tendrá una zona útil de desplazamiento de 1435 x 2110 mm.

El instrumento de medición irá sujeto a una estructura denominada “portainstrumento”. Este portainstrumento poseerá un cuerpo de aluminio y se deslizará sobre un par de ejes guía mediante bujes de grilon. El mismo contará con dos brazos sujetos al cuerpo a través de bisagras, lo que permitirá su apertura frontal.

Para la guía del portainstrumentos se utilizarán ejes de acero al carbono SAE 1010 trefilado de 16 mm de diámetro.

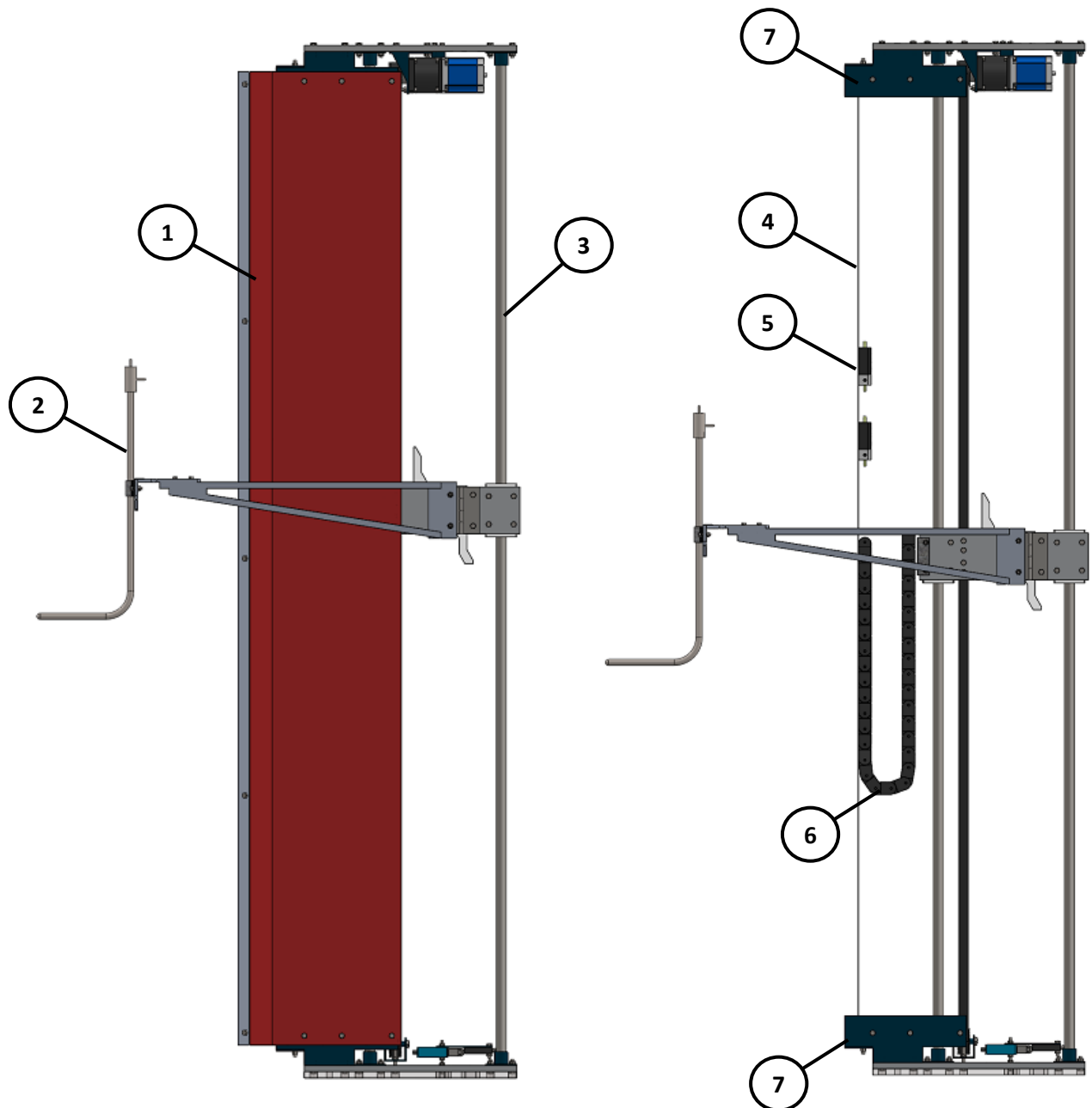


PORTAINSTRUMENTOS: 1) Bujes; 2) Accionamiento de sensor de posición y fines de carrera; 3) Bisagras; 4) Brazo; 5) Bulones de unión entre los brazos; 6) Sujeción del instrumento de medición.

El portainstrumento formará parte de lo que se denominará “eje vertical”. Debe diferenciarse el “eje vertical” el cual se refiere a todo el conjunto de elementos como ser, portainstrumento, motor, transmisión, ejes guía, etc. que permiten el desplazamiento vertical del instrumento, de los “ejes guía vertical” los cuales son únicamente las barras de acero sobre las que se desplazan los bujes del portainstrumento.

El instrumento de medición deberá conectarse a sensores, transductores, fuentes de alimentación o lo que se requiera. Para ello, estos componentes irán montados en el eje vertical y las mangueras, cables, etc, viajarán hasta el instrumento mediante una cadena portacables. Para la sujeción de los sensores, etc, y guía de la cadena portacables se utilizará una chapa de acero galvanizado perforado.

La protección contra el viento de todo lo anterior se realizará mediante una chapa de protección de policarbonato compacto montada al frente del eje vertical. Esta chapa estará sujeta al eje en sus extremos mediante bulones, pudiendo así ser retirada para trabajar en los sensores, mangueras y demás elementos que se encuentren detrás de esta. Para poder retirar esta chapa de protección se deberán abrir los brazos del portainstrumento, de ahí la necesidad de las bisagras en los extremos de los brazos.

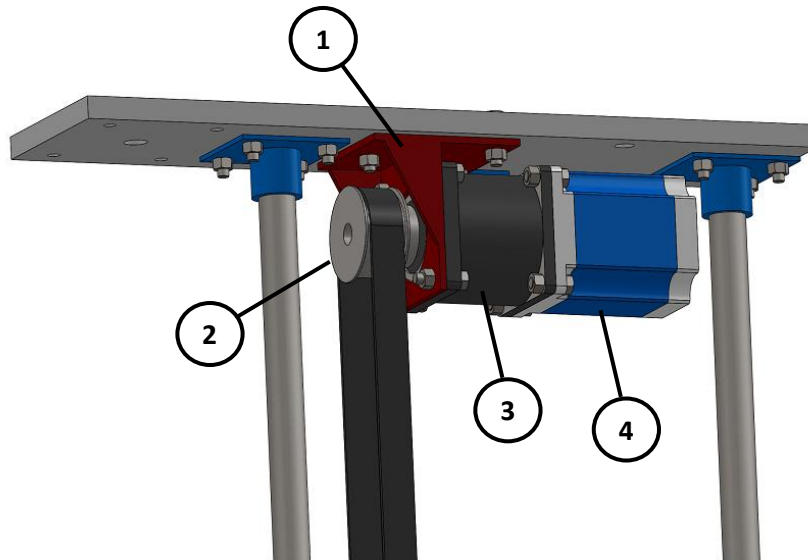


EJE VERTICAL: 1) Chapa de protección vertical; 2) Instrumento de medición; 3) Ejes guía; 4) Guía para cadena portacables y soporte de sensores, etc.; 5) Sensores, Driver, etc.; 6) Cadena portacables vertical; 7) Soportes de chapa de protección vertical.

Para el desplazamiento vertical del portainstrumento se utilizará un motor paso a paso Nema 23 de 2,1 Nm de torque acoplado a un reductor planetario de relación 5:1. Este reductor accionará a su vez una transmisión por correa síncrona la cual estará sujeta al portainstrumento. De esta manera se obtendrá un desplazamiento de precisión de 0,06 mm por paso del motor. El motor se ubicará en la parte superior de eje vertical en una posición fija mientras que la polea de retorno ubicada en la parte inferior podrá variar su posición mediante los bulones que la sujetan, lo que permitirá realizar el tensado de la correa desde ese punto.

Para conocer la ubicación exacta del portainstrumento respecto de la estructura se utilizará un sensor de posición ubicado en la parte inferior del eje vertical. Por su parte para evitar que el portainstrumento colisione contra la estructura si el mismo se desplaza en exceso se montarán fines de carrera en la parte

superior e inferior del eje vertical. Tanto el sensor como los fines de carrera serían accionados mediante placas de PVC de geometría especial ubicadas en el portainstrumento.

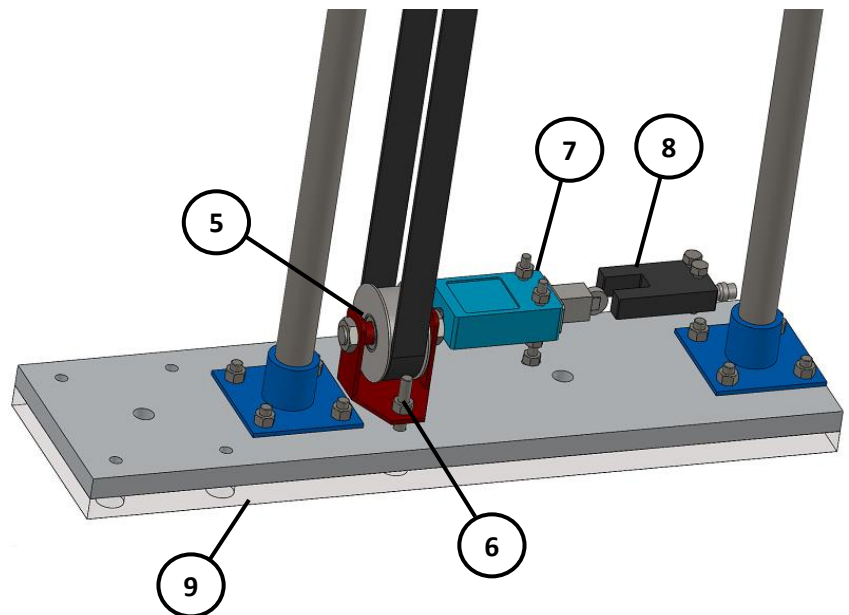


EJE VERTICAL – EXTREMO SUPERIOR:

- 1) Soporte del motor; 2) Polea síncrona; 3) Reductor planetario; 4) Motor paso a paso

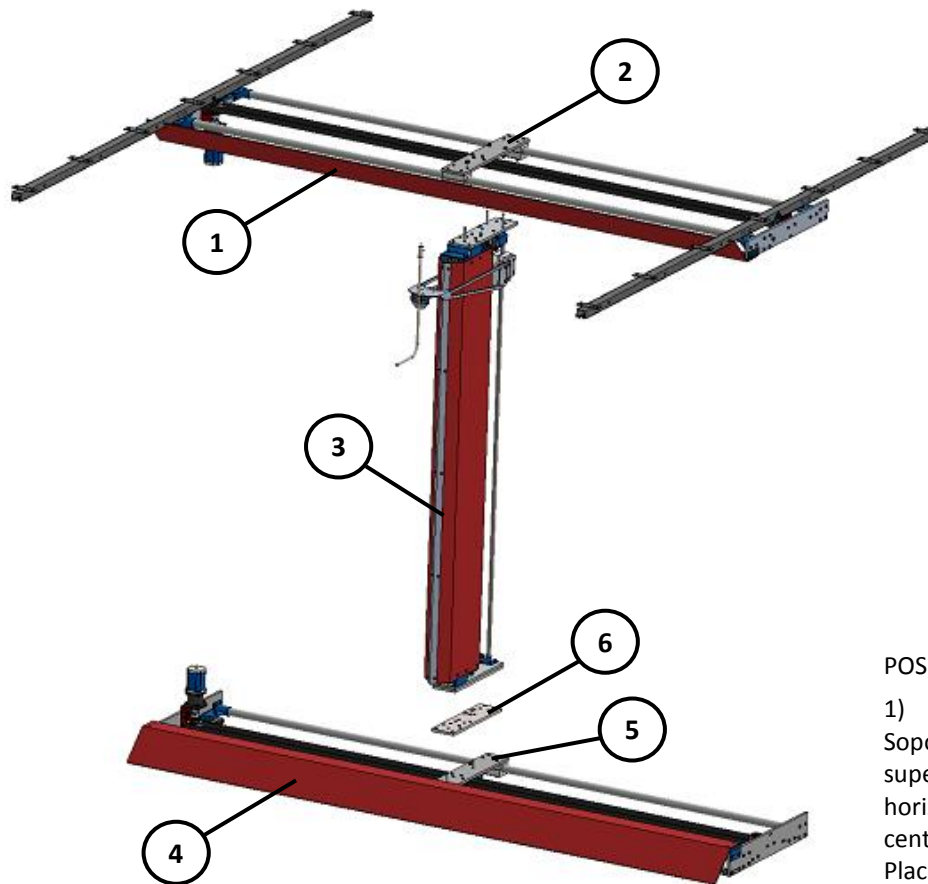
EJE VERTICAL – EXTREMO INFERIOR:

- 5) Polea de retorno; 6) Bulones para sujeción de polea de retorno y tensado de la correa; 7) Fin de carrera vertical inferior; 8) Sensor de posición vertical; 9) Apoyo del eje vertical.



El eje vertical en su totalidad se desplazará horizontalmente soportado por dos ejes horizontales, uno en la parte superior y otro en la parte inferior. Cada uno de estos ejes horizontales poseerá un soporte central al cual irá fijado el eje vertical mediante bulones. De esta manera el posicionador estará compuesto por 3 grandes bloques, el eje vertical, el horizontal superior y el horizontal inferior, de aproximadamente 24, 37 y 42 Kg de peso cada uno respectivamente, lo que permitirá su armado y desarmado de forma práctica cuando se requiera.

Como tanto el extremo inferior del eje vertical como el soporte central del eje horizontal inferior poseerán bulones para la sujeción de sus componentes, los mismos no se podrían apoyar uno sobre el otro, por ello entre estos se colocará una placa de apoyo con perforaciones en los lugares donde se ubiquen bulones.



POSICIONADOR COMPLETO:

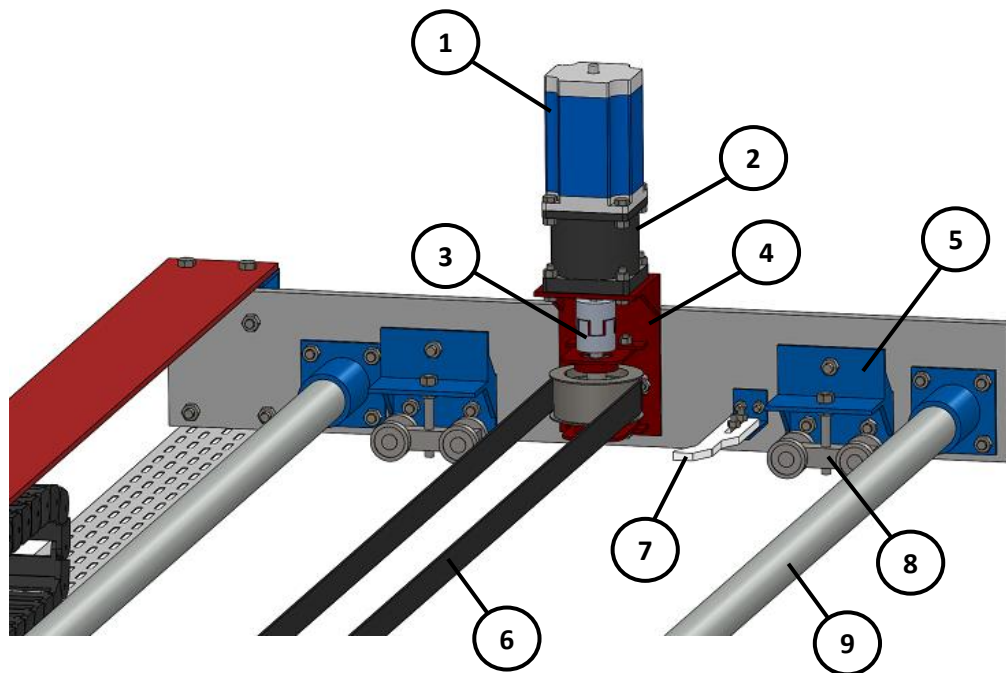
- 1) Eje horizontal superior; 2) Soporte central del eje horiz. superior; 3) Eje vertical; 4) Eje horizontal inferior; 5) Soporte central del eje horiz. inferior; 6) Placa de apoyo

Los soportes centrales de cada eje horizontal serán soportados por dos ejes guía de acero SAE 1010 trefilado de 28 mm, sobre los cuales se deslizarán mediante bujes de grilon. Para el desplazamiento horizontal de estos soportes centrales se utilizará en cada eje horizontal un motor paso a paso Nema 23 de 3,1 Nm de torque, los cuales funcionarán sincronizados para producir el desplazamiento del eje vertical de forma paralela.

Cada motor estará acoplado a un reductor planetario de relación 10:1 el cual a su vez accionará una transmisión por correa síncrona que estará sujeta a los soportes centrales. De esta manera se obtendrá un desplazamiento de precisión de 0,044 mm por paso del motor. Los motores se ubicarán en la parte izquierda de cada eje horizontal (visto desde la dirección del viento dentro del túnel) en una posición fija mientras que la polea de retorno ubicada en la parte derecha podrá variar su posición mediante los bulones que la sujetan, lo que permitirá realizar el tensado de la correa desde ese punto. En este caso la polea motora no podrá montarse directamente sobre el reductor por lo cual poseerá un soporte independiente y se acoplará al reductor mediante un acople flexible.

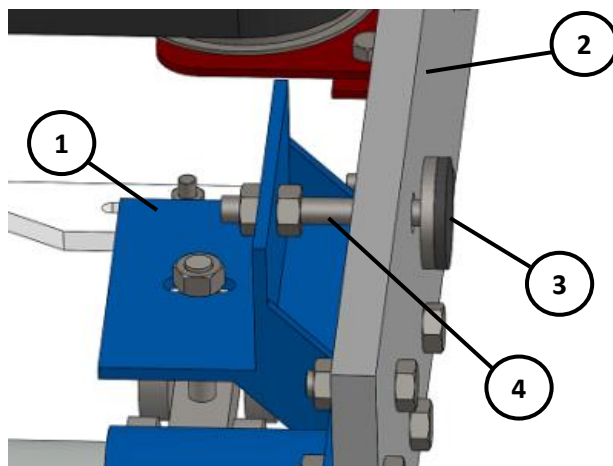
Ambos ejes horizontales podrán desplazarse de forma manual a lo largo del túnel de viento. Para ello, los ejes serán soportados en sus extremos mediante carros con ruedas. En el caso del eje inferior el mismo rodará directamente sobre el suelo del túnel, mientras que en el eje superior los carros se desplazarán dentro de rieles sujetos al techo del túnel.

Estando ya el posicionador en la ubicación axial deseada el desplazamiento en este sentido deberá eliminarse. Para ello en el eje horizontal inferior se utilizarán bulones que comprimirán al eje contras las paredes del túnel, impidiendo así el desplazamiento del mismo debido a la fuerza de rozamiento. En el caso del eje horizontal superior el desplazamiento se bloqueará mediante una sujeción rígida entre los soportes extremos del eje y los rieles. Para esto, los rieles poseerán perforaciones a distancias fijas de manera de sujetar el eje superior a alguna de ellas.



EJE HORIZONTAL INFERIOR – EXTREMO IZQUIERDO:

1) Motor paso a paso; 2) Reductor de velocidad planetario; 3) Acople flexible; 4) Soporte de motor; 5) Soporte de carro y bulón de traba axial; 6) Correa síncrona; 7) Accionamiento de sensor de posición y fin de carrera; 8) Carro; 9) Eje guía horizontal.

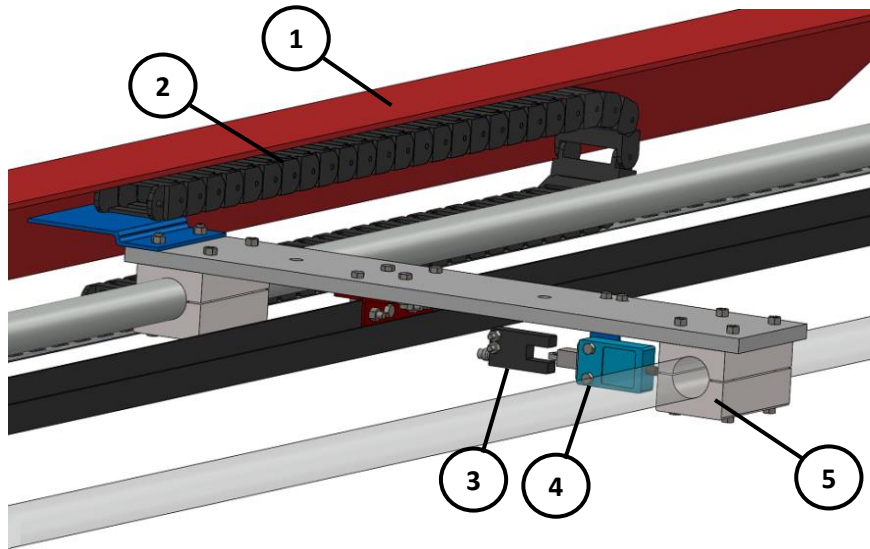


EJE HORIZONTAL INFERIOR – EXTREMO IZQUIERDO:

1) Soporte de carro y bulón de traba axial; 2) Soporte lateral; 3) Goma de contacto; 4) Bulón traba axial.

Para conocer la ubicación exacta del eje vertical y para evitar que el mismo colisione contra la estructura se montarán aquí también un sensor de posición y un único fin de carrera. En este caso los mismos se ubicarán en la parte inferior del soporte central del eje horizontal inferior, y serán accionados mediante placas de PVC de geometría especial ubicadas en los extremos del eje horizontal inferior.

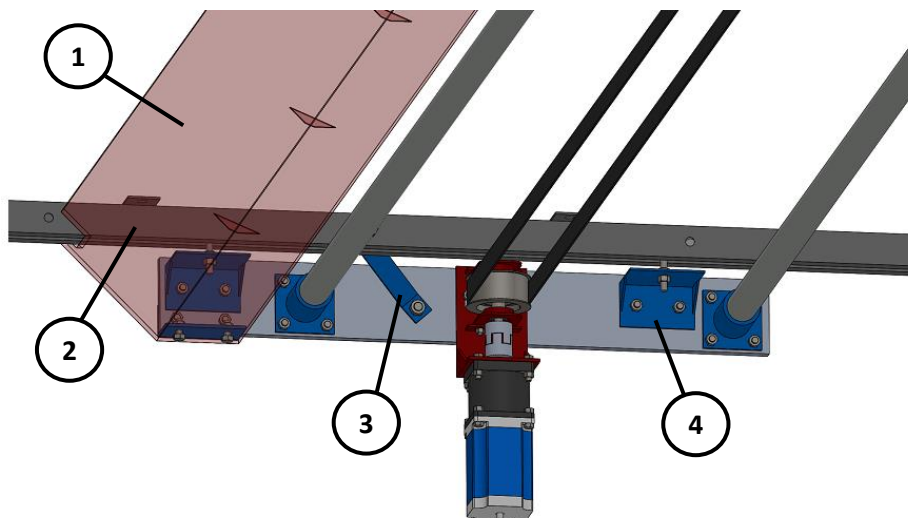
Por su parte en el soporte central del eje horizontal inferior irá conectada la cadena portacables para la conexión eléctrica y de señal del eje vertical.



EJE HORIZONTAL INFERIOR –
SOPORTE CENTRAL:

- 1) Chapa de protección; 2) Cadena portacables horizontal; 3) Sensor de posición; 4) Fin de carrera; 5) Buje.

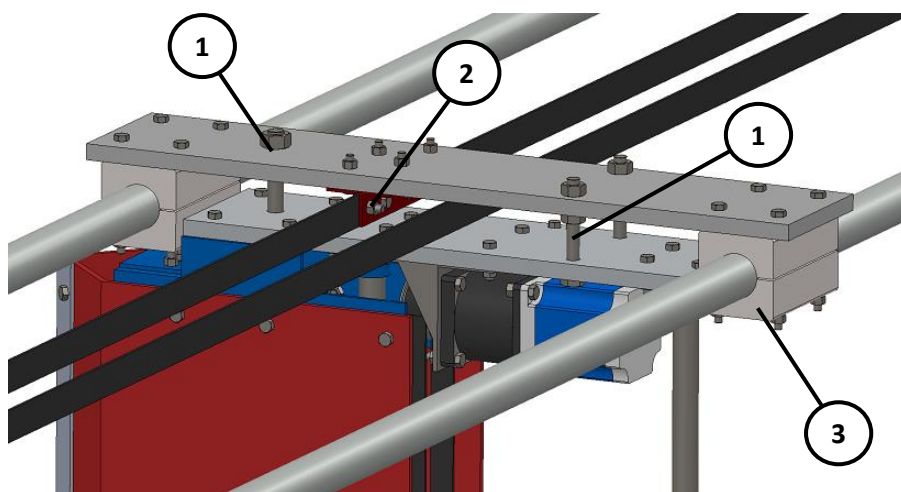
El eje horizontal superior estará suspendido de rieles mediante carros con ruedas. Estos rieles estarán abulonados al techo del túnel en una posición fija, por lo que su extracción demandará tiempo. Aun así, en caso de requerir retirar el posicionador para realizar algún ensayo en el cual no se lo necesite, estos rieles por ubicarse próximo a la superficie del techo no deberían generar un efecto significativo por lo que podría no ser necesario retirarlos.



EJE HORIZONTAL
SUPERIOR – EXTREMO
IZQUIERDO:

- 1) Chapa de protección; 2) Riel porta carro; 3) Sujeción rígida al riel; 4) Sujeción del carro.

Dado que el interior del túnel de viento no posee una altura uniforme pudiendo esta variar hasta 1,5 cm, la altura del posicionador deberá adaptarse a esta variación. Para ello, se regulará la longitud del eje vertical en el punto donde esta se sujeta el soporte central del eje horizontal superior. La unión en este punto se realizará entonces mediante bulones con tuerca y contratuerca tanto en el extremo del eje vertical como en el soporte central.



EJE HORIZONTAL
SUPERIOR – SOPORTE
CENTRAL:

1) Bulón de ajuste; 2)
Sujeción del soporte
central a la correa; 3) Buje.

El peso total del posicionador con todos accesorios rondaría los 110 Kg.



4. SELECCIÓN DE MOTORES Y SISTEMA DE TRANSMISIÓN

4.1 SELECCIÓN DEL TIPO DE TRANSMISIÓN

Para realizar un desplazamiento de precisión como el requerido por el posicionador los sistemas más utilizados son:

- Correas síncronas: Son relativamente económicas y no requieren mantenimiento excepto posibles re tensados de las mismas. Teniendo en cuenta la gran longitud de la correa los desplazamientos muy pequeños pueden ser absorbidos por la elasticidad de la correa, principalmente cuando se invierte el sentido de giro.
- Tornillos de transmisión: Eje de acero con rosca trapezoidal o cuadrada y tuerca de bronce, plástico o metal. Debido a no poseer elementos elásticos permiten realizar desplazamientos sumamente pequeños con mucha exactitud. Poseen mucha superficie de contacto por lo que se pierde mucha potencia por fricción. Debido a que no se comercializa normalmente tornillos de la longitud requerida para este proyecto estos deberían mandar a maquinarse especialmente, por lo que se vuelve una alternativa costosa.
- Tornillos de bolas recirculantes: Similar a los tornillos de transmisión, pero la tuerca en lugar de deslizarse sobre el tornillo posee elementos rodantes en su interior que transmiten la potencia del tornillo a la tuerca. Este sistema hace que el desgaste y el rozamiento sean mínimos, reduciendo también así la potencia requerida del motor. Es el mecanismo ideal para movimientos de extrema precisión. Pero para el nivel de precisión requerido en ese proyecto no se justifica el elevado costo de usar este tipo de sistemas

Los costos de las distintas alternativas se obtuvieron mediante contacto vía telefónica con los proveedores luego de un dimensionamiento aproximado de los elementos. En este punto se pudo comprobar que la transmisión por correa síncrona era la alternativa más económica y que a su vez cumplía con las necesidades de precisión deseadas en el posicionador.

Si se quisiera utilizar tornillos de transmisión de rosca trapezoidal, dos tornillos de 7/8" 4 hpp y 2,2 m y un tornillo de 5/8", 4 hpp y 1,6 m con tuercas cilíndricas de acero poseerían un costo total aproximado de USD 348.

En el caso de tornillos de bola recirculantes, dos ejes de Ø20 mm, paso 5 mm y 2,2 m de longitud y un eje de Ø16 mm, paso 5 mm y 1,6 m, con tuercas de bolas recirculantes y todos sus accesorios poseerían un costo total aproximado de USD 1490.

Teniendo en cuenta que la precisión del posicionador no necesita ser demasiado elevada (en comparación con un Router cnc) el costo de utilizar tornillos de transmisión o de bolas recirculantes no se justifica..

Se decide entonces realizar la transmisión mediante correas síncronas.



4.2 SELECCIÓN DEL TIPO DE CORREA

Se trabajó con correas síncronas Gates por la facilidad de su selección gracias a las herramientas informáticas proporcionadas por el fabricante.

Para la selección de las correas y poleas se utilizó el software libre Design Flex Pro de dicha marca. Con el mismo, a partir de las características requeridas en la transmisión como ser potencia del motor, velocidad, distancia entre ejes y tolerancias, se obtuvieron las distintas combinaciones posibles de correas y poleas que podrían utilizarse. Este a su vez también arrojó para cada alternativa el sobredimensionamiento de la transmisión y la fuerza de tensado requerida por la correa.

Debido a que las dimensiones estarán determinadas por la estructura y no podrán modificarse, las longitudes de las correas deberán adaptarse a estas. Por ello se utilizarán correas abiertas, las cuales se unirán por sus extremos a los elementos a desplazar. Dentro del catálogo de la marca Gates se encuentra una sección de correas sincrónicas abiertas denominadas Long Length las cuales vienen en los formatos GT carbón, GT, HTD y CTB.

Ahora bien, el Software Design Flex Pro solo realiza la selección de correas síncronas sin fin, es decir, correas de longitudes ya determinadas en su fabricación. Por ello, debido a que las distancias entre los ejes de las poleas serán mayores a 1500 mm, las soluciones ofrecidas por el software utilizaban siempre correas de paso mayor a 8 mm (8M), lo que implicaba el uso de poleas de diámetro considerable, lo cual no solo aumentaría el torque necesario de los motores por aumentar el brazo de palanca, sino que además requerirían para su montaje un espacio mayor al disponible. Para utilizar poleas de diámetro aceptables deberían utilizarse correas de paso de 5 mm (5M) como máximo para el eje vertical y 8 mm como máximo para los ejes horizontales, las cuales en formato sin fin poseen longitudes considerablemente menores a las necesarias y por ello el software no las considera como posibles opciones.

Por lo tanto, dado que las correas a utilizar serán en realidad abiertas, pudieron incluirse en el cálculo a las correas 5M y 8M al considerar en el software una desviación importante ($> 25\%$) en la distancia requerida entre centros de las poleas. Así, el software pudo encontrar alguna solución con correas sin fin 5M y 8M cuya longitud fueran lo suficientemente grandes para caer dentro del intervalo admisible. De esta manera se pudo determinar qué modelo de correa podría ser utilizada, su sobredimensionamiento, las posibles poleas a utilizar y la fuerza de tensión requerida por la correa.

Si bien la distancia real entre los centros de las poleas será considerablemente mayor que la utilizada en los cálculos, si se tiene en cuenta el sobredimensionamiento de la transmisión calculada por el software, sumado a que el uso del posicionador no sería continuo si no intermitente u ocasional, se estima que este factor no afectará al normal funcionamiento de la transmisión ni reducirá su vida útil.

4.3 MOTORES

4.3.1 SELECCIÓN DEL TIPO DE MOTORES

A la hora de realizar movimientos de precisión, como el de posicionar el instrumento de medición en un punto determinado, lo más adecuado es el uso de motores paso a paso o motores de corriente continua con encoders.

Los motores paso a paso no realizan avances continuos si no que avanzan de forma discreta, o como su nombre lo indica, avanzan a pasos. Estos motores pueden posicionarse con mucha precisión ya que, una vez alcanzado la posición del paso deseado, permanecen enclavados en dicho lugar. De esta manera puede conocerse con precisión la posición del mismo simplemente sabiendo la cantidad de pasos que ha realizado en una u otra dirección. Además, estos poseen muy buena repetitividad y el error de posición



que pudieran tener no es acumulativo, volviéndose cero luego de un giro completo del mismo. Es por esto que los motores paso a paso, siempre que la carga sea perfectamente conocida, pueden trabajar como sistemas abiertos sin realimentación, por lo que controlar su posición es muy simple. El principal inconveniente del mismo es que para hacerlos avanzar se debe variar la alimentación de sus bobinados en una secuencia determinada, por lo que, por simplicidad, se requiere utilizar drivers especialmente diseñados para esto, los cuales en ocasiones son más costosos que los motores mismos. Estando detenidos poseen par de retención si sus bobinas están energizadas, por lo que pueden soportar carga estando detenidos.

Los motores de corriente continua por su parte poseen avances continuos, es decir, avanzan permanentemente mientras estén alimentados eléctricamente. El posicionamiento en este caso resulta más difícil de controlar ya que, una vez desenergizado el motor, el punto de detención final depende de la inercia del sistema. Debido a su movimiento continuo para conocer con exactitud la posición de los mismos se debe utilizar encoders acoplados a los ejes de los mismos, por lo tanto, se debe trabajar como sistema realimentado lo cual aumenta la complejidad para trabajar con los mismos. Poseen la ventaja de ser fáciles de accionar ya que para ello solo se requiere energizar sus bobinas de forma continua. Para invertir su giro solo se debe invertir el sentido de la corriente en sus bobinas mediante un puente H relativamente económico. Un importante inconveniente es que los mismos estando detenidos no poseen par de retención, por lo que la carga podría moverse por efectos externos como ser por ejemplo su propio peso.

Comparación entre ambos tipos de motores:

Motores Paso a Paso

Avance discreto con intervalos muy pequeños
Exactitud para el posicionamiento
Determinación sencilla de su posición
Requiere Drivers costosos
Posee par de retención

Motores de C.C.

Avance continuo
Menor exactitud para el posicionamiento
Determinación de posición mediante encoder
Requiere puente H relativamente económico
No posee par de retención

Comparando las características de cada uno con las necesidades requeridas en el posicionador se opta por trabajar con motores paso a paso en todos los ejes. Para ello se tuvo en cuenta varios factores como ser la mayor facilidad para determinar la posición, haciendo de esta manera más simple la programación del sistema de control y reduciendo al mismo tiempo la probabilidad de errores. Además, al trabajar con cargas dinámicas, resulta necesario que los motores posean par de retención para así mantener firme el instrumento de medición. En especial es indispensable dicho par de retención para el movimiento del eje vertical, ya que de otra manera el motor dejaría caer el porta instrumentos debido a su propio peso.

4.3.2 CARACTERÍSTICAS CONSTRUCTIVAS DE LOS MOTORES PASO A PASO

En estos motores cada giro de 360° está dividido en pasos y el avance del motor se realiza de a un paso a la vez. La cantidad de pasos en que se divide cada revolución está determinada por la construcción del motor, siendo las configuraciones más comunes de 200 o 400 pasos por revolución, lo que corresponde a avances angulares de 1,8 y 0,9 grados por paso respectivamente.

Estos motores tienen la posibilidad de quedar encavados en una posición, esto quiere decir que una vez alcanzado un paso, mientras sus bobinas se encuentren energizadas, el motor permanecerá fijo pudiendo soportar carga. Una vez desenergizadas sus bobinas el mismo podrá girar libremente u ofrecer cierto par resistente según su tipo de funcionamiento.

Motor paso a paso híbrido bipolar, 200 pasos/rev, Nema 23, torque 2,2 N.m

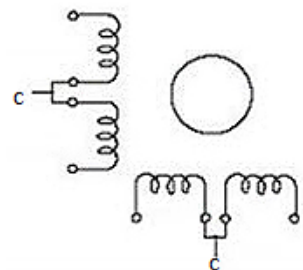


Existen 3 tipos fundamentales de motores paso a paso:

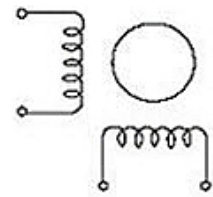
- De reluctancia variable: Posee un estator bobinado y un rotor dentado de hierro generalmente laminado. Una vez energizadas sus bobinas el rotor busca alinearse de manera de reducir la reluctancia del circuito, lo que ocurre cuando los dientes del rotor se alinean con los polos energizados del estator. Con estos pueden obtenerse pasos mucho más pequeños que con los de imán permanente además de mejorar su respuesta dinámica debido a la menor inercia del rotor. Su principal desventaja es el bajo torque comparado con los de imán permanente de dimensiones similares. No ofrece par resistente estando sus bobinas desenergizadas.
- De imán permanente: Posee un estator bobinado y un rotor de imán permanente. Al energizar sus bobinas se produce la atracción entre los polos generados por las mismas y los del imán permanente del rotor. Poseen mejor torque que los de reluctancia variable pero pasos angulares mayores. Debido al imán permanente poseen par resistente al desenergizar sus bobinas.
- Híbridos: Es una combinación de los anteriores mediante lo cual pueden conseguirse torques grandes con pasos pequeños.

En este proyecto se utilizarán motores paso a paso híbridos por ser los más difundidos y por consiguiendo los más fáciles de conseguir. Estos a su vez se clasifican en unipolares y bipolares.

- Motores unipolares: Cuentan con cuatro bobinados alimentados de forma independiente. La corriente en cada una de ellos circula siempre en un mismo sentido, por lo tanto, solo se necesita controlar los instantes en que se energizan o desenergizan. Es por esto que este tipo de motores son los más fáciles de utilizar. El control de la alimentación de las bobinas puede realizarse fácilmente mediante transistores utilizados como elementos de corte o conducción. Su desventaja radica en que en cada instante solo una o dos bobinas se encuentran energizadas al mismo tiempo, por lo que el torque de los mismos es inferior a los bipolares de igual tamaño. Se identifican fácilmente por que salen al exterior 5, 6 u 8 cables, pudiendo los de 6 y 8 cables conectarse también como bipolares. Los puntos comunes C pueden unirse dentro del motor (motor de 5 cables), salir ambos puntos al exterior (motor de 6 cables) o no existir conexión interna de bobinas y salir al exterior los extremos de todas ellas (motor de 8 cables).



- **Motores bipolares:** Cuentan con dos bobinados donde la corriente en los mismos varía su sentido de circulación. Ambas bobinas se encuentran siempre energizadas y para ir de un paso a otro una de ellas debe invertir su polaridad. Por ello cada bobina debe estar alimentada a través de un puente H, haciendo de estos motores más complicados de controlar que los anteriores. Al trabajar con ambas bobinas energizadas permanentemente poseen un mayor torque que los anteriores. Se identifican porque los mismos poseen 4 cables para su conexión.



Como se mencionó los motores unipolares pueden venir de 5, 6 o 8 cables. Los de 5 cables solamente permiten conexión unipolar ya que los puntos comunes se conectan dentro del motor. Los de 6 cables permiten conexión unipolar al conectar los puntos C exteriormente o conexión bipolar serie dejando libres los extremos C y utilizándolo como motor de 4 cables. Los de 8 cables al tener todos los extremos de bobinas accesibles pueden conectarse como unipolar, bipolar serie o bipolar paralelo como puede observarse en la siguiente figura:



En ambas configuraciones siempre que en todas las bobinas circule la misma corriente el torque generado será el mismo. Debido entonces a que en la configuración serie es una misma corriente la que atraviesa cada par de bobinas, esta configuración consumirá la mitad de la corriente que la configuración en paralelo para un mismo torque generado. Para ello, por poseer la configuración serie el doble de resistencia que cada resistencia individual en configuración paralelo, la configuración serie deberá ser alimentada con el doble de tensión. Por lo tanto, ambas configuraciones generaran el mismo torque solo que la configuración serie consumirá la mitad de corriente y deberá ser alimentada con el doble de tensión. Esta reducción de la corriente con la configuración serie tiene la desventaja de que al conectar las bobinas en serie se tiene 4 veces más inductancia por fase que la configuración paralelo, lo que aumenta la constante de tiempo y disminuye el par generado a altas velocidades. Por el contrario, en modo de conexión bipolar paralelo los motores producen más torque en altas velocidades consumiendo para ello mayor corriente.

Teniendo en cuenta que los motores trabajarán a baja velocidad en el proyecto, podrían utilizarse motores paso a paso bipolares o unipolares de 6 u 8 cables con conexión bipolar serie para reducir el consumo de corriente y poder utilizar drivers de menor costo. De todas maneras, los motores, su conexión y los drivers utilizados dependerán de que modelos se encuentren en el mercado al momento de la construcción del posicionador, siendo los bipolares los más comercializados.

En los motores paso a paso bipolares, estando el motor ya energizado y en una posición fija, para hacerlo avanzar un paso se debe modificar la polaridad de una de las bobinas, y según cuál de ellas se modifique, el motor avanzará un paso en uno u otro sentido. La inversión de la polaridad de las bobinas se controla mediante un elemento externo encargado de la alimentación de las mismas. De esta manera para realizar un avance constante del motor las bobinas deben cambiar su polaridad en una secuencia

determinada de manera de realizar un paso tras otro. Según la secuencia el motor girará en un sentido u otro y la velocidad de dicha secuencia determinará la velocidad de rotación del motor.

Si llamamos A y B las bobinas del motor bipolar la secuencia de variación de la polarización de sus bobinas para producir su avance deberá ser la siguiente:

	BOBINA A		BOBINA B	
	Extremo 1	Extremo 2	Extremo 1	Extremo 2
Paso 1	+ V	- V	+ V	- V
Paso 2	+ V	- V	- V	+ V
Paso 3	- V	+ V	- V	+ V
Paso 4	- V	+ V	+ V	- V

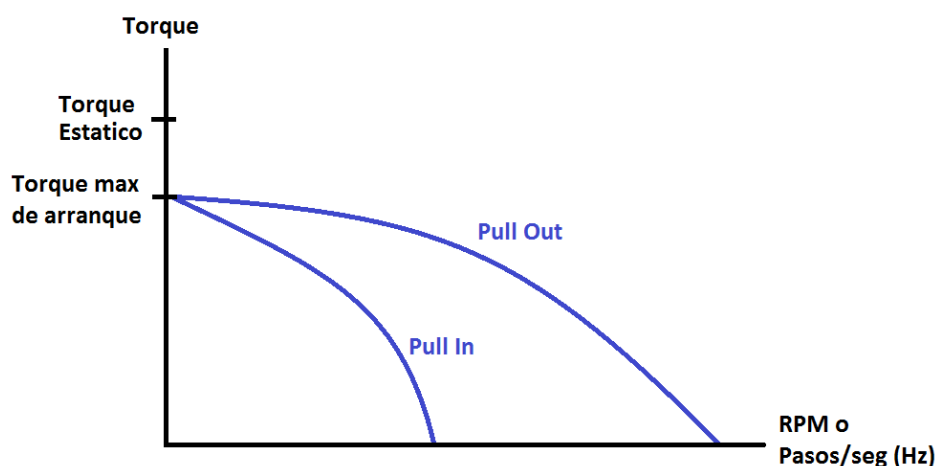
Para cambiar el sentido de giro simplemente se invierte la secuencia.

Las dimensiones de las bridas de montaje se estandarizan, siendo las más comunes las NEMA 17, 23 y 34.

4.3.3 CARACTERÍSTICAS DINÁMICAS DE LOS MOTORES PASO A PASO

▪ CURVA TORQUE/VELOCIDAD

La relación entre la velocidad de rotación o frecuencia de pulsos con el torque generado por un motor determinado se entrega en forma de curva la cual tendrá la siguiente forma:



La curva inferior "Pull In" representa el rango de torque de la carga (no inercial) que puede aplicarse al motor en el momento del arranque a una frecuencia de pasos dada sin que este pierda pasos. Su valor máximo será el "Torque máximo de arranque", el cual se refiere a un tren de pulsos menor a 10 Hz. Por su parte la curva superior "Pull Out" representa el torque máximo que puede aplicarse al motor a una frecuencia de pasos dada estando este ya en funcionamiento. El área entre ambas curvas será la zona de aceleración que permite ir desde la velocidad de arranque a la velocidad final del motor.



Esta curva es dada en cada caso especificando el tipo de excitación que recibe el motor como ser, corriente, tensión y configuración de micropasos utilizado al momento de confeccionar la curva.

El “Torque estatico” o “Holding Torque” es el par de retención generado por el motor cuando el mismo se encuentra energizado y en reposo.

▪ RESONANCIA

Cuando el motor trabaja sin carga o con poca carga, al realizar un paso el mismo oscila más allá del punto donde debería detenerse debido a la baja inercia. Si en este punto se produce la excitación del siguiente paso estando el motor en una posición desfavorable el mismo entrará en resonancia. Para reducir este efecto debe realizarse lo siguiente:

- La frecuencia de arranque debe ser mayor a 200 Hz ya que a bajas frecuencias es más probable que se produzcan resonancias.
- Puede eliminarse la resonancia a baja frecuencia reduciendo la corriente en las fases
- Es de ayuda para la reducción de las resonancias el uso de medios o cuartos de pasos
- Se debe buscar hacer trabajar al motor a su carga nominal cuando se lo alimente con su corriente nominal. Si ello no es posible se debe reducir la corriente en las fases.

▪ CORRIENTE DE FASE

La corriente nominal es la corriente máxima de trabajo del motor. No siempre es necesario alimentar al motor con su corriente nominal, por el contrario, si la carga es reducida o el motor funciona sin carga el mismo podría resonar.

Cuando el motor está detenido presenta par de retención si sus bobinas se mantienen energizadas. Esto produce calentamiento del mismo por lo que, si no es necesario la retención o se requiere un par de retención inferior al de trabajo, la corriente puede reducirse cuando el motor se encuentra detenido.

▪ TENSIÓN DE FASE

La corriente que circule por los bobinados del motor dependerá de la resistencia de los mismos y la tensión a la cual es alimentada. Debido a la constante de tiempo de estos circuitos, la corriente en las bobinas no alcanzará su valor máximo instantáneamente si no recién luego de un determinado lapso de tiempo. A velocidades bajas, donde la duración de cada paso es relativamente grande, el lapso en el que se alcanza la máxima corriente no tiene mayor influencia, en cambio a altas velocidades el tiempo que lleva a la corriente alcanzar su valor máximo es comparable con la duración del paso, lo que se traduce en una menor corriente eficaz y menor par producido por el motor.

Para reducir este efecto una de las alternativas es aumentar la tensión de alimentación, de esta manera la corriente crece a mayor ritmo alcanzando en menor tiempo la corriente deseada. Ahora bien, por utilizarse una mayor tensión la corriente final también aumentaría, para que esto no ocurra un circuito de control mide el valor de la corriente limitándola cuando se haya alcanzado el valor deseado. Como contraparte negativa cuanto más elevada la tensión se generará más ruido y calentamiento en el motor.

Por su parte, si se estuviera trabajando a bajas velocidades una tensión alta generaría vibraciones, por lo que en estos casos es preferible una baja tensión de alimentación.

4.3.4 DRIVERS PARA MOTORES PASO A PASO

- CARACTERÍSTICAS DE LOS DRIVERS

A la hora de controlar los motores bipolares, por simplicidad, generalmente se utilizan drivers que realizan la secuencia de variación de la polaridad de las bobinas. Estos drivers son comandados mediante dos señales digitales externas que le indican el sentido de giro deseado y el instante en que se debe realizar cada paso. El sentido de giro dependerá de si señal externa destinada a esto se encuentra en estado HIGH o LOW. La otra señal externa al realizar un cambio de flanco de HIGH a LOW indica al driver el momento de realizar cada paso invirtiendo la polaridad de la bobina correspondiente según el sentido de giro.

Los drivers son en realidad dos puentes H, uno para cada bobina, con un circuito de control que permite crear la secuencia necesaria de inversión de polaridad. Además de los puentes H los mismos generalmente incorporan elementos de control y protección como ser limitadores de corrientes a los motores, control de temperatura, control de sobretensión, sistemas de inyección de corriente continua, reducción del consumo de energía con el motor detenido, etc.

A demás, estos driver traen la posibilidad de crear micropasos, es decir, subdividir los pasos. Con esto, en lugar de avanzar de un paso a la vez, puede avanzarse fracciones de paso. Los driver más comunes tienen la posibilidad de subdividir cada paso en 16 partes, logrando así en un motor estándar de 200 pasos obtener 3200 micropasos. Su control en este caso no varía, con una señal digital se le indica el sentido de giro y con otra el momento de realizar cada micro paso, luego el driver es el encargado de generar los micropasos variando la corriente aplicada en cada bobina. Actualmente los driver más sofisticados permiten subdividir cada paso en 100, 125, 250 o más partes. Con esta enorme cantidad de micro pasos y aplicando un sistema de transmisión adecuado pueden conseguirse desplazamientos extremadamente pequeños, los cuales se utilizan más comúnmente en Routers CNC.



Driver Leadshine modelo DM870, Alim. 48VDC, I de salida 0,5 a 7A, micropasos 25.000/rev (en motores de 1,8°), freq. de entrada 200Khz

- FUENTES DE ALIMENTACIÓN PARA LOS DRIVERS

Para obtener un buen rendimiento es importante la correcta selección de la fuente de alimentación, ya que el voltaje suministrado determina el correcto funcionamiento a altas velocidades y la corriente de salida determina el par generado por el motor (sobre todo a bajas velocidades). Por ello se debe elegir una tensión de alimentación suficientemente alta para el uso dado y la tensión teórica de la fuente debe ser al menos 10 % menor a la tensión de entrada máxima del driver dejando así margen para la fluctuación de potencia y la fuerza contraelectromotriz.

Para esta aplicación es preferible el uso de fuentes lineales por su rápida respuesta y capacidad de soportar picos de corriente. Utilizando este tipo de fuentes la corriente nominal de la misma puede ser menor que la corriente del motor, entre un 50 y 70 %. En caso de utilizar fuentes de alimentación conmutadas es importante que la corriente de salida sea superior a la corriente del motor en al menos un 25%.



4.3.5 DIMENSIONAMIENTO DE LOS MOTORES

El torque necesario T_{Mot} del motor para realizar los desplazamientos viene determinado por la expresión:

$$T_{Mot} = (T_{Carga} + T_{Inercial} + T_{Carga(V)}) \cdot \frac{F_S}{\eta}$$

Dónde:

T_{Mot} : Torque necesario del motor.

T_{Carga} : Torque constante de la carga. Torque independientes de la velocidad.

$T_{Inercial}$: Torque inercial. Es el requerido para vencer la inercia de las distintas masas y acelerarlas.

$T_{Carga(V)}$: Torque variable de la carga. Torque proporcional a la velocidad.

F_S : Factor de seguridad. Este factor contempla posibles errores en los cálculos por falta de exactitud en los datos e inexactitud del torque real de los motores respecto de lo indicado por los fabricantes¹. Se utiliza como seguridad un factor $F_S = 2$.

η : Rendimiento total del sistema. El mismo está compuesto por:

η_T Rendimiento de la transmisión por correa síncrona $\eta_T = 97\%^2$.

η_R Rendimiento del reductor.

η_M Rendimiento del motor.

4.3.5.1 TORQUE CONSTANTE DE LA CARGA T_{Carga} :

Este torque es requerido para vencer el efecto de las cargas que no dependen de la velocidad del posicionador, como por ejemplo el peso propio de los componentes, de la fuerza de arrastre del viento, ajuste de los bujes, entre otros.

Las mismas se clasificaran en fuerzas paralelas a los ejes F_P y fuerzas de fricción F_F producto de las fuerzas normales a los ejes. Estas fuerzas serán transmitidas a la correa, y por lo tanto, crearán un par resistente sobre el eje de la polea cuyo brazo de palanca será igual al radio primitivo de esta última. Luego el par ejercido sobre el eje de la polea se transmitirá al motor según la relación de transmisión i del reductor utilizado.

Llamando fuerza resistente F_R a la suma de F_P y F_F y siendo $D_{prim.}$ el diámetro primitivo de la polea e i la relación de transmisión del reductor, el torque T_{Carga} de la carga se obtiene como:

$$T_{Carga} = F_R \cdot \frac{D_{prim.}}{2} \cdot \frac{1}{i} = (F_P + F_F) \cdot \frac{D_{prim.}}{2} \cdot \frac{1}{i}$$

¹ A partir de foros en internet los usuarios frecuentes de motores paso a paso resaltan que al utilizar motores de marcas no reconocidas el torque real de los mismos suele ser inferior al indicado por el fabricante. Esto no ocurre con motores de marcas reconocidas, pero el costo de estos es muy superior. Resulta más económico utilizar motores sobredimensionados de marcas no reconocidas.

² Diseño en ingeniería mecánica de Shigley, 8va Ed. - Capitulo 17-4

- FUERZA PARALELA A LOS EJES F_P

En el caso del posicionador estas fuerzas estarán compuestas únicamente por las fuerzas másicas de los componentes móviles siempre que estas actúen en dirección paralela a los ejes y por lo tanto sean soportadas directamente por la correa. La misma se presentará únicamente en el eje vertical, y será producto del peso propio del portainstrumento y los accesorios relacionados a este. Para su determinación se obtiene la masa de los mismos mediante el modelado realizado en SolidWorks, el cual permite obtener estos datos a partir los componentes ya dimensionados.

Luego la fuerza F_P se determina entonces mediante la expresión:

$$F_P = m_{ed} \cdot g \cdot \cos(\alpha)$$

Dónde:

m_{ed} : Suma de las masas de los elementos a desplazar

α : Angulo entre la dirección de la fuerza peso y la dirección de los ejes

g : Aceleración de la gravedad

- FUERZAS DE FRICCIÓN F_F

Por su parte las fuerzas de fricción F_F se deberán a las cargas que actúan perpendicularmente a los ejes produciendo rozamiento entre estos y los bujes. Estas fuerzas perpendiculares poseerán 4 componentes:

- I. Fuerzas másicas normales a los ejes F_{MN} :

Estas fuerzas serán la suma de los pesos de los diferentes componentes móviles cuando la dirección de estas fuerzas actúe en forma normal a la dirección de los ejes y por lo tanto sean soportadas por estos. Para su determinación se obtiene la masa de los mismos mediante el modelado realizado en SolidWorks.

Luego la fuerza F_{MN} se determina mediante la expresión:

$$F_{MN} = m_{ed} \cdot g \cdot \sin(\alpha)$$

Dónde:

m_{ed} : Suma de las masas de los elementos a desplazar

α : Angulo entre la dirección de la fuerza peso y la dirección de los ejes

g : Aceleración de la gravedad

- II. Fuerza de arrastre del viento F_{ARR} :

Las fuerzas de arrastre actuantes sobre los diferentes elementos móviles producto del viento se transmitirán a los ejes debido a la disposición transversal de estos respecto al flujo del viento.

- III. Fuerzas debido a la excentricidad entre el centro de gravedad de los elementos móviles y el punto de sujeción de la correa F_{Ex} :



Si se tiene en cuenta que el punto de sujeción entre las correas y los elementos a desplazar no se realizará en el centro de gravedad de estos últimos, al momento de ejercer fuerza para moverlos estos tenderán a rotar. Esta rotación será impedida por un par producido por fuerzas F_{Ex} ejercidas por los ejes sobre los bujes. La excentricidad poseerá componentes en dirección Z y X en el caso del eje vertical y solamente en Z en el caso de los horizontales.

Estas fuerzas serán entonces perpendiculares a los ejes y su valor estará en función del peso de los cuerpos a mover, la excentricidad entre el centro de masa y el punto de sujeción de la correa y de la longitud de los bujes. Para determinar su valor se deberá hacer sumatoria de momentos igual a cero respecto de algún punto en particular como se verá con detalle más adelante.

IV. Fuerza por ajuste de los bujes sobre los ejes:

Esta fuerza es difícil de cuantificar y por lo tanto se la contempla dentro del factor de seguridad F_S .

Por lo tanto el torque constante de la carga vendrá dado por la expresión:

$$T_{Carga} = (F_P + (F_{MN} + F_{ARR} + 2 \cdot F_{Ex}) \cdot \mu) \cdot \frac{D_{prim.}}{2} \cdot \frac{1}{i}$$

Siendo μ el coeficiente de rozamiento entre los ejes y bujes.

4.3.5.2 TORQUE INERCIAL $T_{Inercial}$:

Este torque es el requerido para vencer las fuerzas inerciales de las distintas masas y sacarlas del estado de reposo o movimiento constante. El torque inercial se obtiene de la expresión:

$$T_{Inercial} = J_T \cdot \frac{d\omega}{dt}$$

Dónde:

J_T : Momento de inercia combinado del motor y la carga referidos al eje del motor.

ω : Velocidad angular del motor.

Por practicidad la velocidad angular conviene expresarla en términos de la frecuencia de los pulsos, por lo que la expresión del torque inercial sería:

$$T_{Inercial} = \theta_{Pulso} \cdot J_T \cdot \frac{df}{dt} = \frac{\theta_{Motor}}{n} \cdot J_T \cdot \frac{df}{dt} = \frac{2 \cdot \pi / p_m}{n} \cdot J_T \cdot \frac{df}{dt}$$

Dónde:

θ_{Pulso} : Angulo de rotación del motor por cada paso (pulso enviado al driver) [rad].

θ_{Motor} : Angulo de rotación del motor por cada paso constructivo del mismo [rad].

p_m Numero de pasos constructivos del motor por revolución. Normalmente 200

f : Frecuencia del tren de pulsos [Pulsos/Seg].



n : Según la configuración de micropasos utilizada [].

n = 1 Pasos completos

n = 2 Medio paso

n = 4 Cuarto de paso

4.4 PERFIL DE VELOCIDADES

El torque requerido aumentará tanto al aumentar la inercia del sistema como al aumentar la aceleración del mismo. Como puede verse el torque inercial $T_{Inercial}$ dependerá del cociente $\Delta\omega/\Delta t$ lo que se traduce como la aceleración angular del motor al momento del arranque o la parada. Según el tipo de carga utilizada se aplican diferentes tipos de perfiles de velocidad:

- Perfil cuadrado: se utiliza cuando la velocidad final es baja y la carga no muy grande. Con este tipo de perfil se pasa directamente desde el reposo a la velocidad final en el primer paso:

$$f = \text{Cte}$$

- Perfil trapezoidal: Se utiliza cuando la velocidad final es considerable y puede despreciarse el torque variable de la carga $T_{Carga(V)}$. Con este perfil el motor aumenta linealmente su velocidad hasta alcanzar la velocidad máxima luego de determinado número de pasos. Al hacer esto se reduce el incremento de velocidad entre pasos, lo que se traduce en menor aceleración angular y por lo tanto menor torque requerido del motor. El perfil del tren de pulsos posee la forma:

$$f = \frac{T_{Mot} - T_{Carga}}{\theta_{Pulso} \cdot J_T} t + f_1$$

- Perfil exponencial: Se utiliza cuando la velocidad final es considerable y no puede despreciarse el torque variable de la carga $T_{Carga(V)}$. Si el torque variable se relaciona linealmente con la velocidad mediante la constante de fricción viscosa D , la expresión del perfil exponencial tendrá la forma:

$$f = \frac{T_{Mot} - T_{Carga}}{\theta_{Pulso}} - \left(\frac{T_{Mot} - T_{Carga}}{\theta_{Pulso}} - f_1 \right) e^{(D/J_T) \cdot t}$$

El posicionador se desplazará a baja velocidad en un ambiente formado únicamente por aire, por lo que se puede despreciar el torque en función de la velocidad y por ello utilizarse un perfil de velocidades trapezoidal. En este caso el término df será la diferencia entre la frecuencia final f_s y la inicial f_1 del tren de pulsos y dt el tiempo de duración de la rampa de aceleración.

Para determinar los intervalos de tiempos entre pasos durante la rampa de aceleración se utiliza la "Teoría de intervalos de pulsos para aceleración lineal" del libro Stepping motors and their microprocessor controls de Takashi Kenjo Cap 6.4.2.

Para la utilización de esta teoría se parte de las siguientes dos hipótesis:



- El motor es capaz de iniciar su movimiento a la frecuencia f_1 sin perder pasos.
- El motor es capaz de acelerar β [pasos/s²] hasta la frecuencia final f_s .

Luego, si mediante la rampa de aceleración se quiere pasar de la frecuencia f_1 a la frecuencia f_s mediante M pasos, el intervalo de tiempo Δt_m entre el pulso m y el $m + 1$ siendo $1 \geq m < M$, se determina mediante la expresión:

$$\Delta t_m = \frac{2}{\sqrt{f_s^2 - 2 \cdot (M - m - 1) \cdot \beta} + \sqrt{f_s^2 - 2 \cdot (M - m) \cdot \beta}} \quad \text{si } 1 \geq m < M$$

Luego a partir de $m > M$:

$$\Delta t = \frac{1}{f_s} \quad \text{si } m \geq M$$

Dónde:

$$\beta = \frac{2(f_s^2 - f_1^2)}{\sqrt{(2M - 3)^2 + (f_s/f_1)^2 - 1} + (2M - 3)^2}$$

Si a partir de estos intervalos de tiempo se determina la frecuencia del tren de pulsos la misma se obtendría mediante:

$$f_m = \frac{1}{\Delta t_m} \quad \text{si } 1 \geq m < M$$

$$f_m = f_s \quad \text{si } m \geq M$$

Por su parte el instante t_m en que se realiza el paso m se obtiene como:

$$t_m = \frac{2 \cdot (m - 1)}{\sqrt{f_s^2 - 2 \cdot (M - m) \cdot \beta} + \sqrt{f_s^2 - 2 \cdot (M - 1) \cdot \beta}}$$

Y el tiempo de aceleración total Δt_T se obtiene de la expresión anterior dando a m el valor M :

$$\Delta t_T = t_M = \frac{2 \cdot (M - 1)}{\sqrt{f_s^2} + \sqrt{f_s^2 - 2 \cdot (M - 1) \cdot \beta}}$$



NOTA: A partir de pruebas realizadas con un prototipo para probar la programación y las configuraciones, se determinó un buen comportamiento del motor y desplazamiento suave a $f_s = 1250 \text{ Hz}$ ($\Delta t = 0,8 \text{ mSeg}$) utilizando configuración de medios pasos y rampas de aceleración y desaceleración de $M = 20 \text{ pasos}$ a partir de una frecuencia inicial $f_1 = 300 \text{ Hz}$ ($\Delta t = 3,33 \text{ mSeg}$).

Respecto a esta configuración cuando a la misma frecuencia se utilizó pasos completos el motor vibró mucho durante su funcionamiento, y cuando se utilizó cuartos de pasos el desplazamiento resultó muy lento.

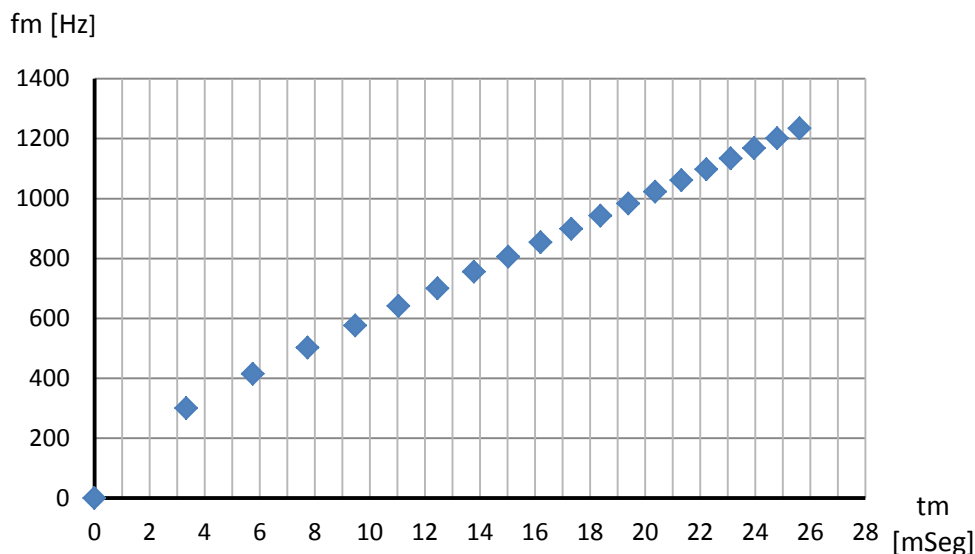
A su vez manteniendo la configuración de medios pasos y el número de pasos M , cuando se aumentó la frecuencia al motor le faltó fuerza para la aceleración, en cambio cuando se redujo la frecuencia se observaron vibraciones en el funcionamiento.

Por lo tanto la rampa de aceleración lineal se determina para los parámetros:

- $f_s = 1250 \text{ Hz}$
- $f_1 = 300 \text{ Hz}$
- $M = 20 \text{ pasos}$
- $n = 2 \text{ (medios pasos)}$

Obteniéndose los tiempos entre pasos y frecuencias que se indican en el siguiente cuadro:

$N^\circ \text{ paso} = m$	t_m [mSeg]	f_m [Hz]	Δt_m [mSeg]	Delay para Arduino [μSeg]	$\frac{df}{dt} = \frac{f_m}{\Delta t_m}$ [Hz/mSeg]
1	0	0	3,333	1667	
2	3,333	300	2,415	1208	90
3	5,748	414	1,993	997	47
4	7,742	502	1,737	869	44
5	9,479	576	1,560	780	43
6	11,040	641	1,428	714	42
7	12,468	700	1,324	662	42
8	13,792	755	1,241	621	41
9	15,033	806	1,171	586	41
10	16,205	854	1,112	556	41
11	17,317	899	1,061	531	41
12	18,379	942	1,016	508	41
13	19,396	983	0,977	489	41
14	20,373	1023	0,942	471	40
15	21,316	1061	0,910	455	40
16	22,227	1098	0,882	441	40
17	23,109	1133	0,856	428	40
18	23,965	1168	0,832	416	40
19	24,798	1201	0,810	405	40
20	25,608	1234	0,8	400	40
21		1250	0,8	400	0



NOTA: Si se tomara por definición al término df como la diferencia entre la frecuencia final $f_s = 1250 \text{ Hz}$ y la inicial $f_1 = 300 \text{ Hz}$, y a dt como el tiempo de duración total de la rampa de aceleración $dt = 25,608 \text{ mSeg}$, se tendría un cociente:

$$\frac{df}{dt}_{\text{Definicion}} = 37100 \frac{\text{Hz}}{\text{seg}} = 37100 \frac{1}{\text{seg}^2}$$

En cambio si se determina este cociente para cada intervalo Δt_m teniendo en cuenta la frecuencia final e inicial en cada uno de estos, el máximo valor del término df/dt corresponderá al del primer paso siendo su valor:

$$\frac{df}{dt}_{\text{Max Por paso}} = 90 \frac{\text{Hz}}{\text{mSeg}} = 90000 \frac{1}{\text{Seg}^2}$$

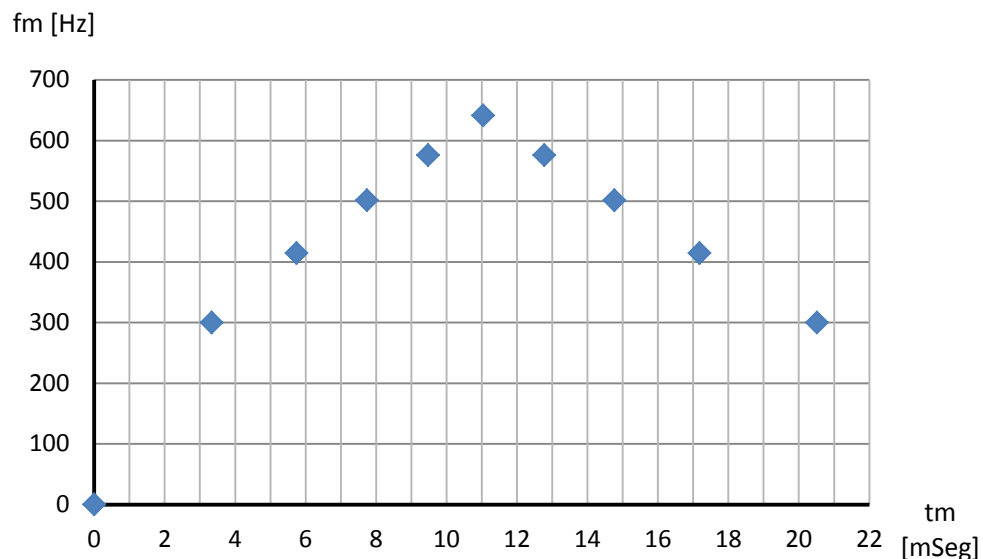
Por ello, por seguridad se utilizará este último valor de df/dt para el cálculo del Torque Inercial.

Para el caso de avances menores a 40 pasos (20 pasos en la rampa de aceleración más 20 en la de desaceleración) la rampa deberá interrumpirse anticipadamente para iniciar la rampa de desaceleración. En el caso de un avance de por ej 10 pasos el sistema acelerará normalmente hasta el paso 6 a partir del cual empezará a desacelerar de manera que el paso 10 posea la misma frecuencia que el paso 2. Esto puede apreciarse en la siguiente tabla que representa a este desplazamiento:

$N^\circ \text{ paso} = m$	t_m [mSeg]	f_m [Hz]	Δt_m [mSeg]	Delay para Arduino [μSeg]	$\frac{df}{dt} = \frac{f_m}{\Delta t_m}$ [Hz/mSeg]
1	0	0	3,333	1667	
2	3,333	300	2,415	1208	90
3	5,748	414	1,993	997	47
4	7,742	502	1,737	869	44
5	9,479	576	1,560	780	43
6	11,040	641	1,737	869	42



7	12,777	576	1,993	997	-38
8	14,771	502	2,415	1208	-37
9	17,186	414	3,333	1667	-36
10	20,519	300		0	-34



4.5 REDUCTOR DE VELOCIDAD

Para reducir el par requerido por el motor a la vez que se aumenta la precisión del posicionador se requerirá de un reductor de velocidad. En este caso se utilizarán reductores tipo planetarios teniendo en cuenta las siguientes ventajas:

- Espacio reducido para su instalación incluso para grandes relaciones de transmisión. Los mismos se acoplan directamente a los motores paso a paso y en su otro extremo se monta o acopla la polea motora, siendo por ello un sistema sumamente compacto.
- Al ser una transmisión mediante engranajes posee alta eficiencia y poco juego (Backlash).
- Soportan considerable carga radial lo que en ocasiones permite el montaje de las poleas motoras directamente sobre el eje del reductor.

Se toma como referencia para su selección la línea NemaTRUE de la marca Micron cuyo catálogo se adjunta en el Anexo A.1.

4.6 PROCEDIMIENTO PARA LA SELECCIÓN DE LOS ELEMENTOS DEL SISTEMA DE TRANSMISIÓN

La selección de cada motor y su sistema de transmisión debe realizarse simultáneamente ya que depende el uno del otro. El torque necesario de cada motor estará en función del diámetro de las poleas utilizadas, siendo este diámetro dependiente de la correa. Pero a su vez, la selección de la correa se realizará en función de la potencia del motor que la acciona. Debe recurrirse entonces a un proceso iterativo.



Para la selección de cada motor y su sistema de transmisión en un primer momento se omitieron las masas e inercias del motor, reductor, correa y poleas, y se adoptó un diámetro para estas últimas, así como el número de pasos por revolución del motor (o micropasos) y el tiempo entre pasos. Con estos datos y otras simplificaciones pudo aproximar el torque necesario T_{Mot} del motor para su preselección. Luego, en función al motor preseleccionado se preseleccionó la correa y las poleas.

Los elementos hasta aquí preseleccionados debieron luego verificarse ya que fueron calculados omitiendo datos o adoptándolos. Para ello se repitió nuevamente el procedimiento anterior pero esta vez teniendo en cuenta todas las masas y fuerzas omitidas y utilizando las dimensiones reales. Se utilizaron para ello las características y dimensiones del motor, reductor, la correa y poleas obtenidos en el cálculo anterior. Se determinó luego el nuevo torque requerido y se comparó con el torque del motor ya preseleccionado, de manera de verificar si este podría ser utilizado o debería reemplazarse por otro. A continuación, en función al motor se determinó nuevamente la correa y poleas comprobando si podrían utilizarse las ya preseleccionadas o si estas deberían modificarse. En el caso de modificarse alguno de los elementos se repitió el procedimiento utilizando ahora las características de los nuevos elementos. Ya seleccionado el motor, correa y poleas se verificó el reductor de velocidad según torque y carga radial admisible.

Lo dicho anteriormente se organiza en los siguientes pasos:

1. Adopción del diámetro de las poleas y cálculo del avance por paso y tiempo de desplazamiento.
2. Determinación del torque del motor T_{Motor} para vencer las fuerzas resistentes omitiendo las masas de las poleas, la correa, el motor y el reductor.
3. Preselección del motor en base al torque necesario.
4. Preselección de la correa y las poleas en función al motor preseleccionado.
5. Nueva determinación del torque del motor T_{Motor} , teniendo en cuenta las masas de las poleas, la correa, el reductor y el motor ya preseleccionados y considerando las nuevas dimensiones de estos componentes.
6. Verificación del motor preseleccionado utilizando el nuevo torque calculado en el paso 5. Si no verifica se selecciona otro motor.
7. Nueva selección de la correa y las poleas con el motor resultante del paso 6.
Si las poleas, la correa o el motor obtenidos en los pasos 6 y 7 son distintos de los utilizados para los cálculos en el paso 5, deben repetirse los pasos 5 en adelante utilizando ahora los nuevos elementos seleccionados en 6 y 7. En el caso de que estos no se modifiquen los elementos utilizados en el paso 5 serían entonces los definitivos.
8. Cálculo del torque de retención necesario.
9. Verificación del reductor de velocidad.

4.7 AVANCE POR PASO, ERROR DE AVANCE Y TIEMPO DE DESPLAZAMIENTO

▪ AVANCE POR PASO

El avance por paso Av_p del posicionador será el menor desplazamiento que puede conseguirse con este y su valor se obtiene a partir de la expresión:

$$Av_p = \frac{\text{Avance por revolucion de la polea}}{N^{\circ} \text{ de pasos por revolucion de la polea}} = \frac{\pi \cdot D_{prim.}}{p_m \cdot n \cdot i}$$

Siendo:

p_m : Numero de pasos constructivos del motor. Normalmente 200.

n : Según la configuración de micropasos utilizada [].

$n = 1$ Pasos completos.

$n = 2$ Medio paso.

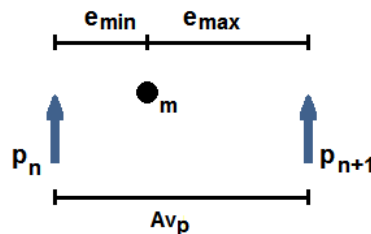
$n = 4$ Cuarto de paso.

i : Relación de transmisión del reductor de velocidad.

$D_{prim.}$: Diámetro primitivo de la polea.

■ ERROR DE AVANCE

Como los motores paso a paso poseen avances discretos la medición no siempre se realizará exactamente en el lugar deseado sino en la posición más cercana que los pasos lo permitan, dando así lugar al error de avance. Como puede verse en la figura siguiente, el punto m indica la posición donde se desea realizar la medición. Ahora bien, debido al avance discreto de los motores el instrumento solo se podrá posicionar en los lugares indicados por las flechas como paso P_n y paso P_{n+1} . Entre estas dos posiciones y el punto m existen errores donde uno será máximo y otro mínimo. Antes de avanzar hasta una de estas posiciones, la programación calculará en cuál de ellas el error será menor para así avanzar hasta esta. En el caso de la figura la programación determinaría que debe avanzarse hasta el paso P_n .



Debido a esto, el error de avance máximo se produciría cuando el punto m se ubica justo en medio de P_n y P_{n+1} ya que ambos errores serán iguales e igual a la mitad del avance por paso Av_p . Por lo tanto el error de avance estará determinado por la expresión:

$$error_{Av} = \frac{Av_p}{2}$$

Cuanto menor sea el avance por paso menor será también el error de avance y mayor la exactitud del posicionamiento. El avance por paso estará entonces acotado ya que avances extremadamente pequeños serian innecesarios mientras que avances muy grandes producirían errores de avance importantes, lo cual crearía problemas a la hora de analizar los datos obtenidos en los ensayos.



▪ TIEMPO DE DESPLAZAMIENTO

La distancia que podrá recorrer el posicionador en cada dirección se denominará Distancia Útil (menor a la distancia entre centros de las poleas correspondientes). El tiempo $t_{Despl.}$ aproximado que le llevará al posicionador recorrer la distancia útil vendrá determinado por:

$$t_{Despl.} = \frac{Distancia\ util}{Av_p} \cdot \Delta t$$

4.8 EJE VERTICAL – DIMENSIONAMIENTO DEL MOTOR Y LA TRANSMISIÓN

1) Adopción del diámetro de las poleas y cálculo del avance por paso y tiempo de desplazamiento:

Se adopta inicialmente:

- Polea de diámetro primitivo $D_{Prim} = 35mm$.
- Motor típico de 200 pasos por revolución ($p_m = 200$).
- Configuración de micropasos como medios pasos ($n = 2$).
- Reductor relación $i = 5$.

Se tendría entonces un avance por paso de:

$$Av_p = \frac{\pi \cdot 35mm}{200 \cdot 2 \cdot 5} \approx 0,055\ mm$$

Con lo que se obtiene un error de avance de 0,0275 mm lo cual se considera una exactitud elevada y por lo tanto aceptable.

Teniendo en cuenta que el tren de pulsos a velocidad constante (luego de la rampa de aceleración) poseerá una frecuencia $f_s = 1250\ Hz$ que implica un tiempo entre pasos de $\Delta t = 1/f_s = 0,8\ mSeg$, se obtendría una velocidad angular en la polea motora de:

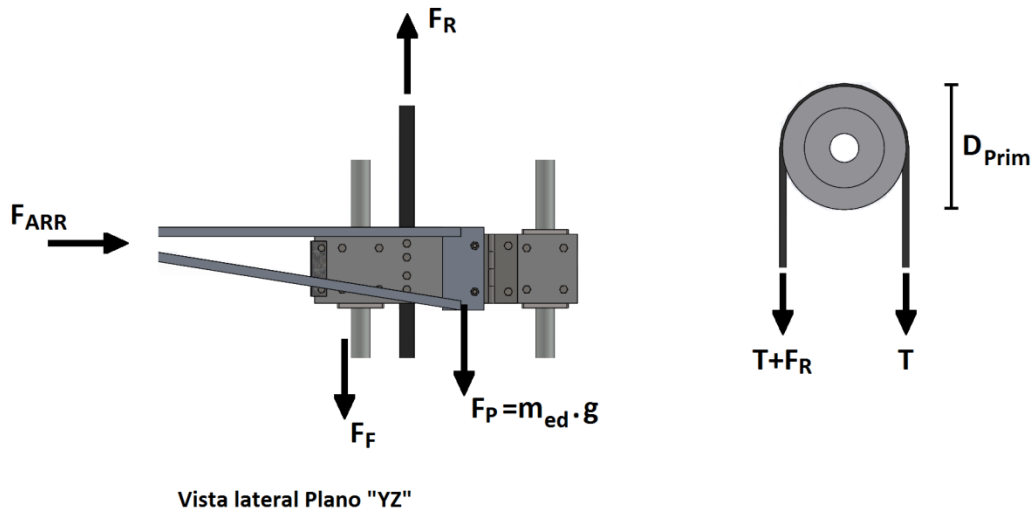
$$\omega = \frac{1}{n \cdot p_m \cdot i} \cdot \frac{60000}{\Delta t} = \frac{1}{2 \cdot 200 \cdot 5} \cdot \frac{60000}{0,8\ mSeg} = 37,5\ RPM$$

Y un tiempo aproximado de desplazamiento para recorrer la Distancia Útil Vertical, estimada en 1435 mm, de:

$$t_{Despl.} = \frac{Distancia\ util}{Av_p} \cdot \Delta t = \frac{1435\ mm}{0,055\ mm} \cdot 0,8\ mSeg \approx 21\ Seg$$

2) Determinación del torque del motor:

TORQUE CONSTANTE DE LA CARGA T_{carga} :



▪ FUERZA PARALELA A LOS EJES F_p

Ya que la dirección de la fuerza de gravedad coincidirá con la dirección de los ejes verticales las fuerzas másicas estarán soportadas directamente por la correa. Esta fuerza estará formada por el peso propio de los elementos que componen el portainstrumento (brazos, soportes, bisagras, bujes, soporte de la correa y bulones), el instrumento de medición con sus accesorios y la cadena portacables vertical.

A partir de la modelación en SolidWorks de la estructura del porta instrumento y de la estimación del peso del instrumento de medición con sus accesorios, la cadena porta cables y las mangueras se estimó una masa de aproximadamente 4,5 Kg. Se tendrá entonces una fuerza paralela al eje de:

$$F_p = 4,5 \text{ Kg} = 44,2 \text{ N}$$

▪ FUERZA DE FRICCIÓN F_f

Esta fuerza será generada por la fricción entre los bujes y los ejes verticales debido a las fuerzas normales actuantes sobre estos últimos. Como se explicó anteriormente estas fuerzas normales estarán compuestas por:

I. Fuerzas másicas normales a los ejes F_{MN} :

En este caso las fuerzas másicas no tendrán componente normal a los ejes debido a que ambas poseerán la misma dirección. Se tendrá por lo tanto:

$$F_{MN} = 0$$

II. Fuerza de arrastre del viento F_{ARR} :

Debido a la forma compleja del porta instrumento y a que parte de este se encontrará tapado por la chapa de protección vertical, un cálculo preciso de la fuerza de arrastre producido por el viento sobre el mismo requeriría de simulaciones de flujo muy precisas mediante software. En cambio, como simplificación, se aproximará su valor mediante el cálculo analítico aplicando la teoría de flujos externos¹. Según esta, la fuerza de arrastre que ejerce un fluido sobre un cuerpo sumergido en el mismo puede determinarse mediante la expresión.

$$F_{ARR} = \frac{1}{2} \cdot C_D \cdot \rho \cdot V^2 \cdot A$$

Dónde:

F_{ARR} : Fuerza de arrastre sobre el cuerpo en [N].

C_D : Coeficiente de arrastre adimensional que depende de la geometría del cuerpo y el número de Reynolds.

ρ : Densidad del fluido en [Kg/m³].

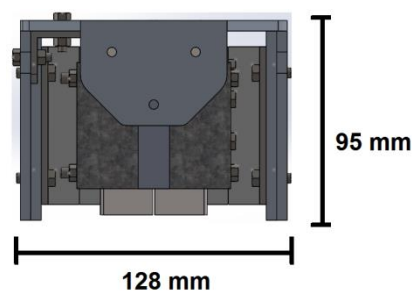
V : Velocidad del fluido en [m/seg].

A : Área del cuerpo proyectada sobre un plano normal a la dirección del flujo en [m²].

Esta ecuación puede aplicarse si se tiene en cuenta que el aire puede modelarse como flujo inviscido (viscosidad despreciable) cuando se trata de flujos externos y se considera incompresible por trabajarse con velocidades inferiores a los 100 m/seg².

El coeficiente de arrastre C_D se obtuvo mediante la Tabla 8.2 del libro Mecánica de los Fluidos, M. Potter y D. Wiggert, 3ra Ed. Allí para un cilindro cuadrado de lados w y un $Re > 10^4$ corresponde un coeficiente $C_D = 2$. Ahora bien, debido a que el porta instrumento estará compuesto por varillas y que gran parte del mismo se encuentra cubierto por la chapa de protección vertical, se puede tomar un coeficiente de menor valor, en este caso se adoptó $C_D = 1$.

Teniendo en cuenta la máxima velocidad del viento dentro del túnel $V = 25 \text{ m/seg}$, la densidad $\rho = 1,247 \text{ kg/m}^3$ del aire atmosférico a 10° C y el área proyectada obtenida de la siguiente imagen frontal del portainstrumento:



La fuerza de arrastre sobre el portainstrumento será:

$$F_{ARR} = \frac{1}{2} \cdot 1 \cdot 1,247 \frac{\text{kg}}{\text{m}^3} \cdot \left(25 \frac{\text{m}}{\text{seg}}\right)^2 \cdot (0,128 \cdot 0,095) \text{m}^2 = 4,8 \text{ N}$$

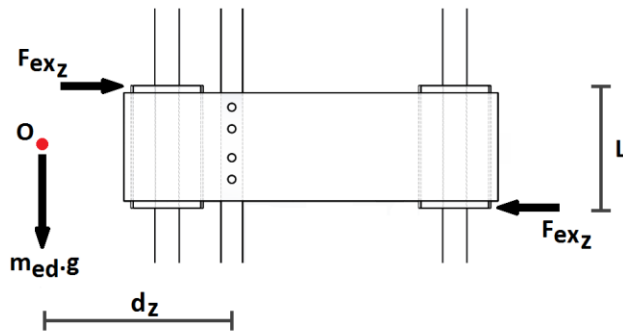
¹ Mecánica de los Fluidos, M. Potter y D. Wiggert, 3ra Ed. - Capitulo 8.3.1

² Mecánica de los Fluidos, M. Potter y D. Wiggert, 3ra Ed. - Capitulo 3.3.2 y 3.3.4

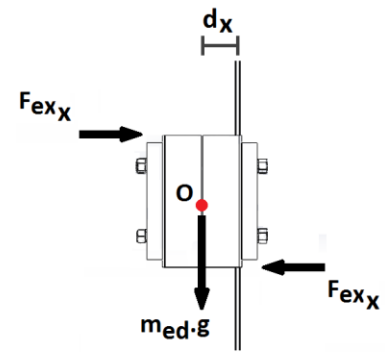
Esta fuerza en realidad se calculó en exceso ya que el área proyectada utilizada es mayor a la real si se tiene en cuenta que parte del portainstrumento se encontrará detrás de la chapa de protección vertical, por lo que el viento no actuará directamente sobre esa área. Aun así, siendo esta fuerza de fricción calculada con bastante exceso, se observa que su valor será despreciable comparado con las fuerzas máxicas.

III. Fuerzas debido a la excentricidad entre el centro de gravedad de los elementos móviles y el punto de sujeción de la correa F_{Ex} :

Esta fuerza tendrá dos componentes F_{ExZ} y F_{ExX} debido a que existirá excentricidad tanto en dirección Z como en dirección X entre la posición del centro de gravedad del porta instrumento y el punto de sujeción de la correa con el mismo, como se aprecia en los siguientes esquemas:



Vista lateral Plano "YZ"



Vista frontal Plano "XY"

Si se toma como referencia el punto de sujeción de la correa, en ambos casos se observa como la excentricidad tenderá a hacer rotar al portainstrumento en sentido anti horario. Esta rotación será impedida por el par ejercido por las fuerzas F_{ExZ} y F_{ExX} sobre los bujes. Para determinar la magnitud de estas fuerzas se plantea sumatoria de momentos igual a cero respecto del punto de sujeción de la correa. Se tiene entonces:

$$\begin{aligned} \text{Plano "YZ"} \quad \sum M &= m_{ed} \cdot g \cdot d_Z = F_{exZ} \cdot L = 0 \\ \text{Plano "XY"} \quad \sum M &= m_{ed} \cdot g \cdot d_X = F_{exX} \cdot L = 0 \end{aligned}$$

Siendo entonces:

$$\begin{aligned} F_{exZ} &= \frac{m_{ed} \cdot g \cdot d_Z}{L} \\ F_{exX} &= \frac{m_{ed} \cdot g \cdot d_X}{L} \end{aligned}$$

Dónde:

- m_{ed} : Suma de las masas de los elementos a desplazar
- d_Z : Excentricidad en dirección Z
- d_X : Excentricidad en dirección X
- L : Longitud de los bujes



g : Aceleración de la gravedad

Mediante el modelado en SolidWorks se estimó una masa de aproximadamente 4,5 Kg; $d_z = 61,1 \text{ mm}$; $d_x = 24,7 \text{ mm}$ y $L = 85 \text{ mm}$. Luego:

$$F_{exz} = \frac{m_{ed} \cdot g \cdot d_z}{L} = \frac{4,5 \text{ Kg} \cdot g \cdot 61,1 \text{ mm}}{85 \text{ mm}} = 31,7 \text{ N}$$

$$F_{exx} = \frac{m_{ed} \cdot g \cdot d_x}{L} = \frac{4,5 \text{ Kg} \cdot g \cdot 24,7 \text{ mm}}{85 \text{ mm}} = 12,8 \text{ N}$$

Luego:

$$F_{ex} = \sqrt{F_{exz}^2 + F_{exx}^2} = 34,2 \text{ N}$$

Por lo tanto el torque constante de la carga T_{Carga} será entonces:

$$T_{Carga} = (F_P + (F_{MN} + F_{ARR} + 2 \cdot F_{Ex}) \cdot \mu) \cdot \frac{D_{prim.}}{2} \cdot \frac{1}{i}$$

$$T_{Carga} = (44,2 \text{ N} + (0 + 4,8 \text{ N} + 2 \cdot 34,2 \text{ N}) \cdot 0,54) \cdot \frac{0,035 \text{ m}}{2} \cdot \frac{1}{5} = 0,293 \text{ N.m}$$

Siendo $\mu = 0,54$ ¹ el coeficiente de rozamiento entre los ejes y bujes de grilon.

TORQUE INERCIAL $T_{Inercial}$:

Despreciando en un primer momento la inercia de la correa, la polea, el motor y el reductor, solo se toma en cuenta el momento de inercia del portainstrumento. Como la fuerza para moverlo se transmitirá por la correa y a su vez esta se acoplará al motor mediante el reductor de relación i , el valor del momento de inercia del portainstrumento respecto del eje del motor será de:

$$J_{Carga} = masa \cdot \left(\frac{D_{prim.}}{2} \right)^2 \cdot \frac{1}{i^2} = 4,5 \text{ Kg} \cdot \left(\frac{0,035 \text{ m}}{2} \right)^2 \cdot \frac{1}{5^2} = 0,000055125 \text{ Kg.m}^2$$

Luego, teniendo en cuenta el valor df/dt explicado en el capítulo 4.4, se tendrá un torque inercial de:

$$T_{Inercial} = \frac{2 \cdot \pi \cdot p_m}{n} \cdot J_r \cdot \frac{df}{dt}$$

¹ De la página: www.jq.com.ar de Industrias JQ, empresa especializada en plásticos de ingeniería.



$$T_{Inercial} = \frac{2 \cdot \pi / 200}{2} \cdot 0,000055125 \text{ Kg} \cdot \text{m}^2 \cdot 90000 \frac{1}{\text{Seg}^2} = 0,078 \text{ N} \cdot \text{m}$$

TORQUE DEL MOTOR T_{Motor} :

Considerando:

$F_S = 2$ Factor de seguridad

$\eta_T = 97\%$ Rendimiento de la transmisión por correa síncrona

$\eta_R = 1$ Rendimiento del reductor (aproximación)

$\eta_M = 1$ Rendimiento del motor (aproximación)

Se obtiene un torque motor aproximado de:

$$T_{Mot} = (0,293 \text{ N} \cdot \text{m} + 0,078 \text{ Nm}) \cdot \frac{2}{0,97 \cdot 1 \cdot 1} = 0,77 \text{ N} \cdot \text{m} = 77 \text{ N} \cdot \text{cm}$$

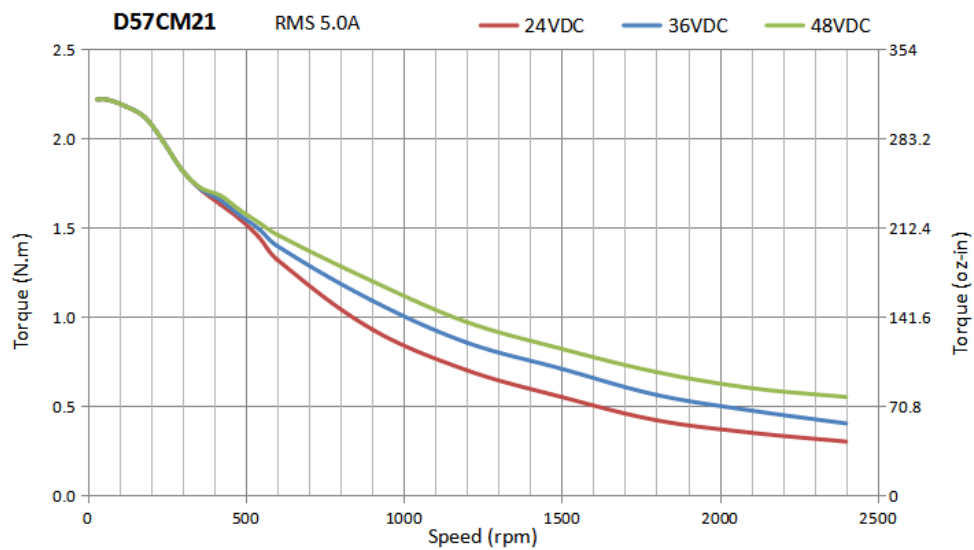
3) Preselección del motor:

Según la lista de motores disponibles en la página de internet del proveedor, el motor Nema 17 de mayor torque produce 0,8 Nm. Ahora bien el torque de este motor es muy próximo al torque necesario recién calculado, considerando que varios factores no se tuvieron en cuenta los cuales aumentarían el torque requerido. Por ello se decide preseleccionar un motor Nema 23, en este caso un motor de 2,1 Nm y 67 mm de longitud.

Se preselecciona:

- Motor paso a paso marca Leadshine modelo D57CM21.
- Tamaño de brida: NEMA 23.
- Nº de cables 4.
- Holding Torque: 2,1 N.m.
- Corriente: 5 A/fase.
- Tension: 1,26 V.
- Momento de inercia: 0,57 Kg.cm².
- Diámetro de los ejes: 8 mm.
- Long. eje principal: 19,4 mm.
- Peso: 1,2 Kg.

Curva Torque/Velocidad del motor preseleccionado:



Aquí el fabricante no aclara que configuración de micropasos se utilizó para la elaboración del gráfico. Teniendo en cuenta que a determinada velocidad de rotación el torque entregado utilizando micropasos es menor que utilizando pasos completos, si se considera que el grafico está referido a pasos completos el mismo indicaría el torque máximo que puede entregar el motor a cada velocidad. Luego, para la configuración de medio paso, el torque entregado sería ligeramente menor al mostrado en el gráfico.

Para el caso requerido el motor girará a 187,5 RPM en configuración de medios pasos, por lo que, según el grafico, el motor entregaría menos de 2,1 Nm de torque, pero al estar este valor considerablemente por encima de los 0,77 Nm se tiene la seguridad de que el torque entregado por el motor superaría al torque requerido.

4) Preselección de las correas y poleas:

Utilizando el software Design Flex Pro con los siguientes datos:

- Correas evaluadas: PowerGrip GT3, GT4, GTX y HTD
- Torque: 2,1 N.m
- Velocidad: 187,5 RPM
- Potencia del motor: 42 W (A partir de torque y la velocidad de rotación)
- Eficiencia del motor: 1
- Relación del reductor de velocidad: 5
- Eficiencia del reductor de velocidad: 1
- Factor de servicio: 1,3
- Distancia entre ejes: 1500 mm $\pm$ 25%

Se obtuvieron las siguientes alternativas:



Línea de prod.	Correa	Motriz (Dientes)	Conducida (Dientes)	RPM conducida	Δ RPM	sobredimens.	Coste relativo Δ	Entre centros (mm)	Δ dist. entre ejes (mm)	F. Tracc (kgf)	Ancho (mm)
PowerGrip GT3	2440-5MGT3-9	28	28	37,5	0	1,0	1,0	1.150	-350	63	18
PowerGrip GT3	2440-5MGT3-9	30	30	37,5	0	1,1	1,0	1.145	-355	59	18
PowerGrip GT3	2440-5MGT3-9	32	32	37,5	0	1,3	1,0	1.140	-360	55	18
PowerGrip GT3	2440-5MGT3-15	18	18	37,5	0	1,0	1,3	1.175	-325	98	30
PowerGrip GT3	2440-5MGT3-15	20	20	37,5	0	1,2	1,3	1.170	-330	88	30
PowerGrip GT3	2440-5MGT3-15	21	21	37,5	0	1,3	1,3	1.168	-332	84	30
PowerGrip GT3	2440-5MGT3-15	22	22	37,5	0	1,3	1,3	1.165	-335	80	30
PowerGrip GT3	2440-5MGT3-15	24	24	37,5	0	1,5	1,3	1.160	-340	73	30
PowerGrip GT3	2440-5MGT3-15	26	26	37,5	0	1,7	1,3	1.155	-345	68	30
PowerGrip GT3	2440-5MGT3-9	34	34	37,5	0	1,4	1,4	1.135	-365	52	18
PowerGrip GT3	2440-5MGT3-9	36	36	37,5	0	1,5	1,5	1.130	-370	49	18
PowerGrip GT3	2440-5MGT3-9	38	38	37,5	0	1,6	1,5	1.125	-375	46	18

En este punto se tuvo en cuenta el diámetro de las poleas a utilizar. Según las alternativas con correas de ancho 9 mm la menor polea a utilizarse sería de 28 dientes con un diámetro primitivo de 44,56 mm lo que se alejaría bastante del diámetro adoptado de 35 mm.

Por su parte las alternativas con correas de espesor 15 mm utilizarían poleas de menor diámetro que van de 18 a 26 dientes, correspondiente a diámetros primitivos de 28,65 mm y 41,38 mm respectivamente. En este punto se tuvo en cuenta que a mayor diámetro de la polea menor la fuerza de tracción para tensado, lo que mejoraría la posibilidad de montar la polea directamente sobre el eje del reductor de velocidad.

Correa preseleccionada:

- Correa: PowerGrip GT3 – 5MGT
- Ancho: 15 mm
- Paso: 5 mm
- Longitud: 3120 mm
- Fuerza de tracción: 720 N
- Sobredimensionamiento: 1,5
- Cuerdas: Fibra de vidrio
- Peso específico: 61,5 gr/m
- Peso total: 192 gr

La fuerza de tensionado se toma solo como referencia, ya que al poseer la transmisión real una distancia entre ejes aproximadamente 340 mm mayor, puede ser necesario una mayor fuerza de tensado. Eso se comprobará en el momento de construcción de la maquina según su funcionamiento.

Poleas preseleccionadas:

- Modelo: GT3 24-5MR-15-6F
- Nº de dientes: 24
- Diámetro primitivo: 38,2 mm
- Diámetro correa: 37,06 mm
- Diámetro extensión: 27 mm
- Ancho correa: 20,5 mm
- Ancho total: 28 mm
- Pestañas: 2
- Material: Acero
- Peso: 180 gr
- Macizas con taladro cilíndrico



5) Nueva determinación del torque del motor T_{Motor} :

En este caso debe tenerse en cuenta que la polea preseleccionada es de diámetro primitivo 38,2 mm, mayor a la adoptada en el paso 1 por lo tanto se modifica el torque necesario.

Se tendría entonces un avance por paso de:

$$Av_p = \frac{\pi \cdot 38,2 \text{ mm}}{200 \cdot 2 \cdot 5} \approx 0,06 \text{ mm}$$

Con lo que se tendría un error de avance de 0,03 mm lo cual se considera una exactitud elevada y por lo tanto aceptable.

La velocidad angular de la polea motora no dependerá del diámetro de esta por lo que mantendrá su valor calculado inicialmente de 37,5 RPM.

El tiempo aproximado para recorrer la Distancia Útil Vertical, estimada en 1435 mm, será de:

$$t_{Despl.} = \frac{Distancia\ util}{Av_p} \cdot \Delta t = \frac{1435 \text{ mm}}{0,06 \text{ mm}} \cdot 0,8 \text{ mSeg} \approx 20 \text{ Seg}$$

TORQUE CONSTANTE DE LA CARGA T_{carga} :

▪ FUERZA PARALELA A LOS EJES F_p

Esta fuerza no modificará su valor ya que dependerá únicamente del peso del portainstrumento y no del motor ni el sistema de transmisión. Se mantiene entonces:

$$F_p = 4,5 \text{ Kg} = 44,2 \text{ N}$$

▪ FUERZA DE FRICCIÓN F_f

I. Fuerzas másicas normales a los ejes F_{MN} :

El valor de esta fuerza no dependerá del motor ni del sistema de transmisión, por lo tanto no se modificará:

$$F_{MN} = 0$$

II. Fuerza de arrastre del viento F_{ARR} :

El valor de esta fuerza no dependerá del motor ni del sistema de transmisión, por lo tanto no se modificará:

$$F_{ARR} = 4,8 \text{ N}$$

III. Fuerzas debido a la excentricidad entre el centro de gravedad de los elementos móviles y el punto de sujeción de la correa F_{Ex} :



El valor de esta fuerza no dependerá del motor ni del sistema de transmisión, por lo tanto no se modificará:

$$F_{ex} = 34,2 \text{ N}$$

Luego, el torque constante de la carga T_{Carga} se modificará por modificarse el diámetro primitivo de la polea. Será entonces:

$$T_{Carga} = (F_P + (F_{MN} + F_{ARR} + 2 \cdot F_{Ex}) \cdot \mu) \cdot \frac{D_{prim.}}{2} \cdot \frac{1}{i}$$

$$T_{Carga} = (44,2 \text{ N} + (0 + 4,8 \text{ N} + 2 \cdot 34,2 \text{ N}) \cdot 0,54) \cdot \frac{0,0382 \text{ m}}{2} \cdot \frac{1}{5} = 0,32 \text{ N.m}$$

TORQUE INERCIAL $T_{Inercial}$:

Se considera ahora la inercia del motor, el reductor, las poleas y la correa.

▪ Carga

El momento de inercia de la carga variará por modificarse el diámetro de la polea. En este caso su valor será de:

$$J_{Carga/Motor} = masa \cdot \left(\frac{D_{prim.}}{2} \right)^2 \cdot \frac{1}{i^2} = 4,5 \text{ Kg} \cdot \left(\frac{0,0382 \text{ m}}{2} \right)^2 \cdot \frac{1}{5^2} = 0,00006567 \text{ Kg.m}^2$$

▪ Correa:

$$J_{Correa/Motor} = masa \cdot \left(\frac{D_{prim.}}{2} \right)^2 \cdot \frac{1}{i^2} = 0,0615 \frac{\text{Kg}}{\text{m}} \cdot (3,12 \text{ m}) \cdot \left(\frac{0,0382 \text{ m}}{2} \right)^2 \cdot \frac{1}{5^2} = 0,0000028 \text{ Kg.m}^2$$

▪ Poleas:

La inercia respecto de su propio eje de una polea que posee dos diámetros viene dado por la expresión:

$$J_{Pol/Pol} = \frac{1}{2} masa_1 \cdot R_1^2 + \frac{1}{2} masa_2 \cdot R_2^2 = \frac{1}{2} \cdot \left(\frac{\pi \cdot D_1^2 \cdot L_1}{4} \cdot \rho \right) \cdot \left(\frac{D_1}{2} \right)^2 + \frac{1}{2} \cdot \left(\frac{\pi \cdot D_2^2 \cdot L_2}{4} \cdot \rho \right) \cdot \left(\frac{D_2}{2} \right)^2$$

$$J_{Pol/Pol} = \frac{\pi \cdot \rho}{32} \cdot (D_1^4 \cdot L_1 + D_2^4 \cdot L_2)$$

Considerando las poleas de acero ($\rho = 7800 \text{ Kg/m}^2$) la inercia de cada una de ellas respecto del motor será:

$$J_{Pol/Motor} = \frac{\pi \cdot \rho}{32} \cdot (D_1^4 \cdot L_1 + D_2^4 \cdot L_2) \cdot \frac{1}{i^2} =$$



$$= \frac{\pi \cdot 7800 \text{ kg/m}^3}{32} \cdot ((0,03706 \text{ m})^4 \cdot 0,0205 \text{ m} + (0,027 \text{ m})^4 \cdot 0,0075 \text{ m}) \cdot \frac{1}{i^2}$$

$$J_{Pol/Motor} = 0,00000131 \text{ Kg. m}^2$$

- Motor:

Según catalogo:

$$J_{Motor} = 0,57 \text{ Kg. cm}^2 = 0,000057 \text{ Kg. m}^2$$

- Reductor:

Se preselecciona en este punto el reductor de velocidad en función de tamaño de brida del motor. En este caso se preselecciona el reductor de velocidad NemaTRUE NT23-005:

$$J_{Red/Motor} = 0,025 \text{ Kg. cm}^2 = 0,0000025 \text{ Kg. m}^2$$

- Total:

Se tendrá entonces un momento de inercia total respecto del motor de:

$$J_T = J_{Carga/Motor} + J_{Correa/Motor} + 2 \cdot J_{Pol/Pol} + J_{Motor} + J_{Red/Motor}$$

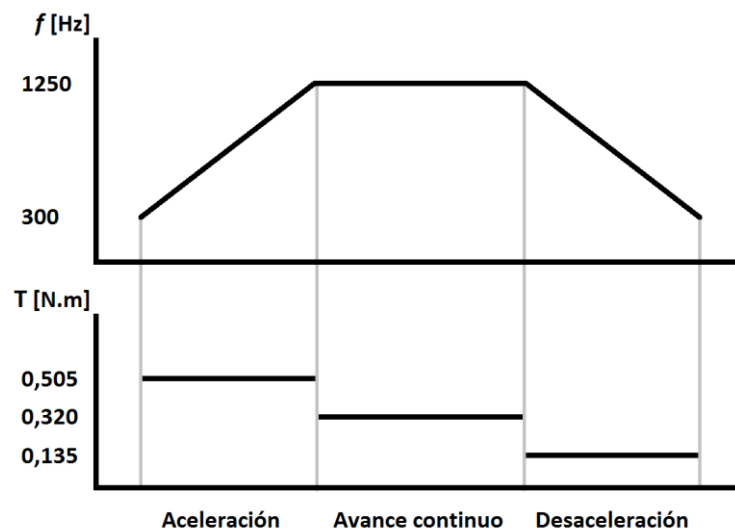
$$J_T = 0,0001306 \text{ Kg. m}^2$$

Luego, se tendrá un torque inercial de:

$$T_{Inercial} = \frac{2 \cdot \pi / p_m}{n} \cdot J_T \cdot \frac{df}{dt}$$

$$T_{Inercial} = \frac{2 \cdot \pi / 200}{2} \cdot 0,0001306 \text{ Kg. m}^2 \cdot 90000 \frac{1}{\text{Seg}^2} = 0,185 \text{ N. m}$$

Perfil de frecuencia del tren de pulsos y toques en cada etapa del movimiento.



TORQUE DEL MOTOR T_{Motor} :

Considerando:

$F_S = 2$ Factor de seguridad

$\eta_T = 97\%$ Rendimiento de la transmisión por correa síncrona

$\eta_R = 93\%$ (De catálogo calculado para el 100% del torque)

$\eta_M = 1$ Rendimiento del motor (aproximación)

Se obtiene así un torque motor aproximado de:

$$T_{Mot} = (0,32 \text{ N.m} + 0,185 \text{ Nm}) \cdot \frac{2}{0,97 \cdot 0,93 \cdot 1} = 1,12 \text{ N.m} = 112 \text{ N.cm}$$

6) Verificación del motor pre seleccionado:

A partir del nuevo valor de torque calculado y encontrándose este lo suficientemente alejado al valor del torque entregado por el motor obtenido de la curva Torque/Velocidad del mismo (mostrado en el punto 3), se verifica el motor preseleccionado de $2,1 \text{ N.m}$ marca Leadshine modelo D57CM21.

7) Nueva selección de correas y poleas:

Utilizando el software Design Flex Pro con los mismos datos utilizados para la preselección del punto 4) pero incluyendo la eficiencia del reductor del 93% se obtuvo entre las alternativas la misma transmisión preseleccionada, solo que en este caso la fuerza de tracción bajó de 720 N a 670 N y el sobredimensionamiento subió de $1,5$ a $1,7$. Por lo tanto se verificó la correa preseleccionada.

Las poleas se mantuvieron sin variación y por lo tanto también se verificaron las ya preseleccionadas.

Las características del motor y transmisión seleccionados para el eje vertical se adjuntan en los anexos A.2, A.3 y A.18.

8) Cálculo del torque de retención necesario:

Cuando el motor se encuentre detenido el par de retención deberá soportar solamente el peso propio del portainstrumento, el cual generará un torque cuyo valor será de:

$$T = (F_P + (F_{MN} + F_{ARR} + 2 \cdot F_{Ex}) \cdot \mu) \cdot \frac{D_{prim.}}{2} \cdot \frac{1}{i}$$

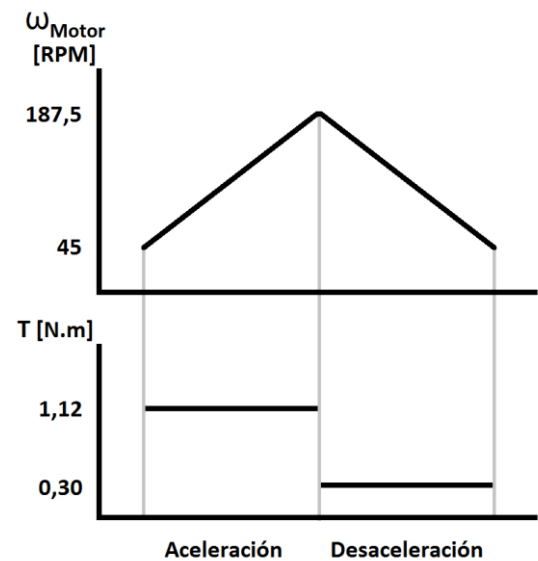
$$T = (44,2 \text{ N} + (0 + 0 + 0) \cdot 0,54) \cdot \frac{0,0382 \text{ m}}{2} \cdot \frac{1}{5} = 0,16 \text{ N.m}$$

Luego, teniendo en cuenta el factor de seguridad y los rendimientos este torque necesario en la retención tendrá un valor de 0,374 Nm.

9) Verificación del reductor de velocidad:

La verificación se realiza siguiendo el método exacto de selección indicado por el fabricante adjunto en el anexo A.1. Aquí el funcionamiento más desfavorable se producirá cuando cada desplazamiento esté compuesto solamente por las rampas de aceleración y desaceleración, sin intervalo de desplazamiento a velocidad constante como se muestra en la siguiente imagen:

En estos casos el torque motor T_{Mot} en las rampas de aceleración y desaceleración serán los obtenidos en el punto 5) habiéndoles aplicado el factor de seguridad y los rendimientos de la transmisión y el reductor.



- Velocidad media de entrada n_m :

En este caso las velocidades n_1 y n_2 serán:

$$n_1 = n_2 = \frac{(187,5 + 45) \text{ RPM}}{2} = 116,25 \text{ RPM}$$

Luego

$$n_m = \frac{n_1 \cdot t_1 + n_2 \cdot t_2}{t_1 + t_2} = \frac{n_1 \cdot t + n_2 \cdot t}{t + t} = n_1 = n_2 = 116,25 \text{ RPM}$$

- Torque equivalente T_{EQ} :

$$T_{EQ} = \sqrt[8,7]{\frac{n_1 \cdot t_1}{n_m \cdot (t_1 + t_2)} \cdot T_1^{8,7} + \frac{n_2 \cdot t_2}{n_m \cdot (t_1 + t_2)} \cdot T_2^{8,7}} = \sqrt[8,7]{\frac{T_1^{8,7}}{2} + \frac{T_2^{8,7}}{2}}$$

$$T_{EQ} = \sqrt[8,7]{\frac{(1,12 \text{ Nm})^{8,7}}{2} + \frac{(0,30 \text{ Nm})^{8,7}}{2}} = 1,03 \text{ N.m}$$

- Torque equivalente modificado T_{EQm} :

Considerando entre 1000 y 2500 ciclos por hora $Q = 0,9$. Aquí se agrega como extra el factor de seguridad 2 utilizado en los cálculos para contemplar posibles inexactitudes entre los cálculos y la construcción real del posicionador. Por lo que el torque equivalente modificado valdrá:

$$T_{EQm} = \frac{T_{EQ}}{Q} \cdot F_s = \frac{1,03 \text{ N.m}}{0,9} \cdot 2 = 2,30 \text{ N.m}$$

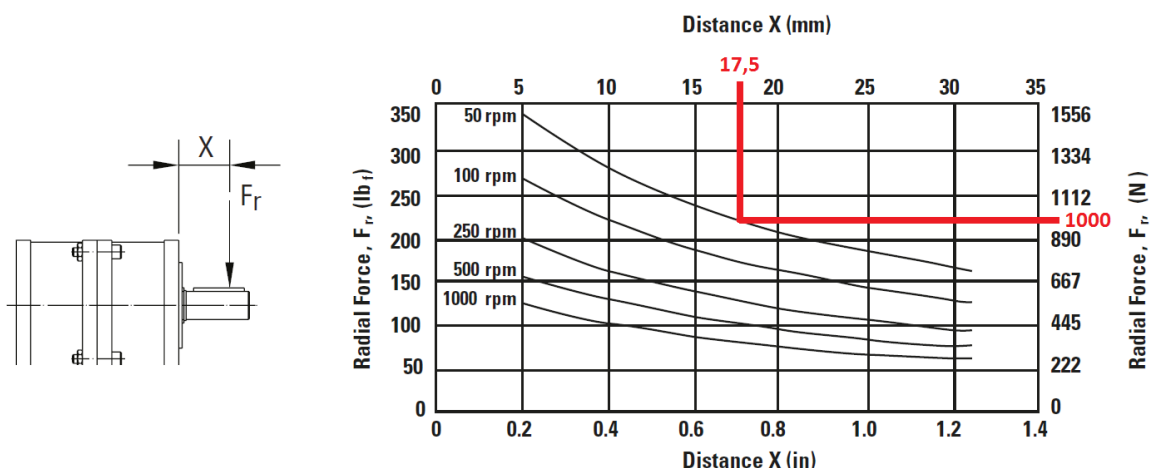
- Verificación de T_{EQm} respecto de T_r del reductor:

Para el modelo NT23-005 y una velocidad inferior a las 1000 RPM, $T_r = 11,9 \text{ N.m} > T_{EQm}$, por lo que el reductor verificará según el torque equivalente.

- Verificación de la fuerza radial:

Según las dimensiones de la polea, el centro de la correa se ubicará a una distancia:

$$X = 28 \text{ mm} - \frac{20,5}{2} \text{ mm} = 17,75 \text{ mm}$$



La polea girará a 37,5 RPM a la cual el reductor, según el grafico, soportaría una carga radial de aproximadamente 1000 N, mayor que la fuerza de tracción de 720 N ejercida por la correa y calculada en el punto 4. Por lo tanto la polea podría montarse directamente sobre el eje del reductor.

4.9 EJES HORIZONTALES – DIMENSIONAMIENTO DE LOS MOTORES Y LAS TRANSMISIONES

1) Adopción del diámetro de las poleas y cálculo del avance por paso y tiempo de desplazamiento:

Se inicia el proceso de cálculo adoptando inicialmente el mismo diámetro de poleas de los ya calculados para el eje vertical y el mismo reductor de velocidad.

- Polea de diámetro primitivo $D_{Prim} = 38,2 \text{ mm}$.
- Motor típico de 200 pasos por revolución ($p_m = 200$)
- Configuración de micropasos como medios pasos ($n = 2$)
- Reductor relación $i = 5$

Se tendría entonces un avance por paso de:

$$Av_p = \frac{\pi \cdot 38,2 \text{ mm}}{200 \cdot 2 \cdot 5} \approx 0,06 \text{ mm}$$

Con lo que se obtiene un error de avance de 0,03 mm.

Utilizando para este caso el mismo tren de pulsos que para el eje vertical con una frecuencia $f_s = 1250 \text{ Hz}$ ($\Delta t = 0,8 \text{ mSeg}$) se tendrá entonces la misma velocidad angular en la polea motora con un valor de 37,5 RPM y un tiempo aproximado para recorrer la Distancia Útil Horizontal, estimada en 2110 mm, de:

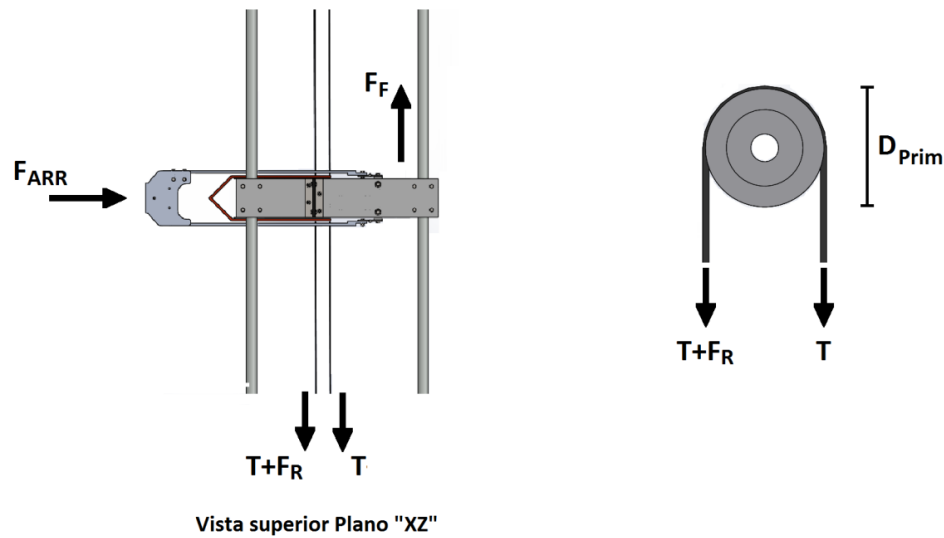
$$t_{Despl.} = \frac{\text{Distancia util}}{Av_p} \cdot \Delta t = \frac{2110 \text{ mm}}{0,06 \text{ mm}} \cdot 0,8 \text{ mSeg} \approx 28 \text{ Seg}$$

2) Determinación del torque del motor.

El eje vertical estará soportado por los ejes guía de los ejes horizontales superior e inferior. Luego, para el desplazamiento se utilizarán dos motores ubicados uno en cada eje, los cuales funcionarán en conjunto de manera que el eje vertical se desplace en forma paralela.

El torque total necesario para el desplazamiento horizontal estará repartido entonces entre ambos motores y, si el montaje del posicionador se realizara de manera exacta, este torque debería repartirse en partes iguales entre ambos motores. Contemplando que la exactitud en el montaje es poco probable, se dimensiona cada motor para que aporte 2/3 del torque total necesario para el desplazamiento horizontal. Para ello se calcula el torque total necesario para mover el eje vertical y luego se determina el torque necesario de cada motor como 2/3 de este torque total.

TORQUE CONSTANTE DE LA CARGA T_{carga} :



- Fuerza paralela a los ejes F_p

En este caso las fuerzas másicas no tendrán componente paralela a los ejes debido a la perpendicularidad entre la dirección de ambas. Se tendrá por lo tanto:

$$F_p = 0$$

- Fuerza de fricción F_f

Esta fuerza será generada por la fricción entre los bujes y los ejes verticales debido a las fuerzas normales actuantes sobre estos últimos. Como se explicó anteriormente estas fuerzas normales estarán compuestas por:

- Fuerzas másicas normales a los ejes F_{MN} :

Debido a que la dirección de la fuerza de gravedad será normal a la dirección de los ejes horizontales que soportarán al eje vertical, las fuerzas másicas estarán soportadas totalmente por los ejes horizontales guía. Esta fuerza estará formada por el peso propio de todos los elementos que componen el eje vertical (chapa de protección vertical, ejes verticales, porta instrumento, motor vertical, reductor, etc), más los elementos del eje horizontal que sirven de soporte del eje vertical y permiten su desplazamiento (soportes centrales horizontales, bujes, cadena portacable horizontal, etc). En otras palabras, esta fuerza estará compuesta por la suma de las masas de todos los elementos que se desplazan durante el movimiento horizontal.

A partir de la modelación en SolidWorks y de la estimación del peso de los diversos accesorios utilizados, se tiene entonces una masa total de los elementos a desplazar de 30 Kg. Luego, la fuerza másica normal a los ejes será de:

$$F_{MN} = m_{ed} \cdot g = 30 \text{ Kg} \cdot g = 295 \text{ N}$$

II. Fuerza de arrastre del viento F_{ARR} :

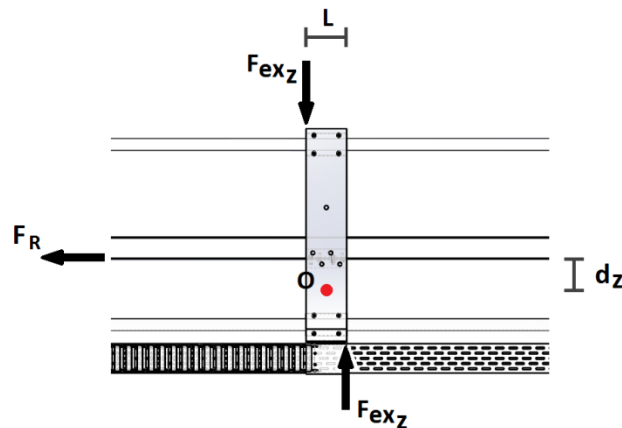
La fuerza de arrastre producido por el viento sobre la parte móvil sería aproximadamente la fuerza de arrastre sobre la chapa de protección vertical, ya que esta cubrirá casi la totalidad de la parte móvil. Los soportes centrales de cada eje horizontal se suponen parcialmente cubiertos de la acción del viento mediante las chapas de protección horizontal.

La fuerza de arrastre sobre la chapa de protección vertical podría aproximarse mediante cálculo analítico aplicando la teoría de flujos externos indicado anteriormente. En este caso el área proyectada sería de 10,3 cm x 157 cm y el coeficiente de arrastre podría aproximarse utilizando el del cilindro equilátero, el cual posee un C_d de 1,4¹. De esta manera para una velocidad de viento $V = 25 \text{ m/seg}$ y una densidad $\rho = 1,247 \text{ kg/m}^3$ se tendría una fuerza de arrastre de:

$$F_{ARR} = \frac{1}{2} \cdot 1,4 \cdot 1,247 \frac{\text{kg}}{\text{m}^3} \cdot \left(25 \frac{\text{m}}{\text{seg}}\right)^2 \cdot (1,57 \cdot 0,103) \text{m}^2 = 88,3 \text{ N}$$

III. Fuerzas debido a la excentricidad entre el centro de gravedad de los elementos móviles y el punto de sujeción de la correa F_{Ex} :

Debido a que el eje vertical se soportará en sus dos extremos (superior e inferior) y desde cada uno de ellos será accionado mediante correas, la excentricidad se producirá solamente en el eje Z, tendiendo a rotar al eje vertical en el plano "XZ" como se aprecia en el siguiente esquema:



Vista superior Plano "XZ"

Si se toma como referencia el punto O el cual representa la línea vertical (dirección Y) que pasa por el centro de masa, se observa como la fuerza resistente F_R tenderá a hacer rotar al porta instrumento en sentido anti horario. Esta rotación será impedida por el par ejercido por las fuerzas F_{Ex} sobre los bujes. Para determinar la magnitud de estas fuerzas se planea sumatoria de momentos igual a cero respecto del punto O. Se tiene entonces:

$$\text{Plano "XZ"} \quad \sum M = F_R \cdot d_Z = F_{ex} \cdot L = 0$$

Siendo entonces:

$$F_{ex} = \frac{F_R \cdot d_Z}{L}$$

¹ Mecánica de los Fluidos, M. Potter y D. Wiggert, 3ra Ed. - Capitulo 8.3.1



Dónde:

F_R : Fuerza resistente. Suma de F_P y F_F

d_Z : Excentricidad en dirección Z

L : Longitud de los bujes

Teniendo en cuenta que F_R está compuesto por F_F el cual incluye a F_{ex} :

$$F_R = F_P + F_F = 0 + F_F = 0 + (F_{MN} + F_{ARR} + 2 \cdot F_{Ex}) \cdot \mu$$

$$F_{ex} = \frac{(F_{MN} + F_{ARR} + 2 \cdot F_{Ex}) \cdot \mu \cdot d_Z}{L}$$

$$F_{Ex} = \frac{(F_{MN} + F_{ARR}) \cdot \mu \cdot d_Z}{(L - 2 \cdot \mu \cdot d_Z)}$$

Los valores de F_{MN} y F_{ARR} ya fueron calculados, mientras que la excentricidad d_Z se obtiene mediante el modelado en SolidWorks dando un valor de 10 mm. Luego:

$$F_{Ex} = \frac{(295 \text{ N} + 88,3 \text{ N}) \cdot 0,54 \cdot 10 \text{ mm}}{(85 \text{ mm} - 2 \cdot 0,54 \cdot 10 \text{ mm})} = 27,9 \text{ N}$$

Por lo tanto el torque constante de la carga T_{Carga} será entonces:

$$T_{Carga} = (F_P + (F_{MN} + F_{ARR} + 2 \cdot F_{Ex}) \cdot \mu) \cdot \frac{D_{prim.}}{2} \cdot \frac{1}{i}$$

$$T_{Carga} = (0 + (295 \text{ N} + 88,3 \text{ N} + 2 \cdot 27,9 \text{ N}) \cdot 0,54) \cdot \frac{0,0382 \text{ m}}{2} \cdot \frac{1}{5} = 0,91 \text{ N.m}$$

Siendo $\mu = 0,54$ el coeficiente de rozamiento entre los ejes y bujes de grilon.

TORQUE INERCIAL $T_{Inercial}$:

En un primer momento se desprecia la inercia de la correa y el motor y se adopta las mismas poleas y el reductor seleccionados para el eje vertical.

▪ Carga:

Como la fuerza para moverla se transmitirá por las correas y a su vez estas se acoplarán al motor mediante reductores de relación i , el valor del momento de inercia total del eje vertical respecto de los ejes de los motores será de:

$$J_{Carga/Motor} = masa \cdot \left(\frac{D_{prim.}}{2} \right)^2 \cdot \frac{1}{i^2} = 30 \text{ Kg} \cdot \left(\frac{0,0382 \text{ m}}{2} \right)^2 \cdot \frac{1}{5^2} = 0,0004378 \text{ Kg.m}^2$$



▪ Poleas:

Por tratarse de las mismas poleas seleccionadas para el eje vertical y la misma relación de transmisión i del reductor a utilizar, su momento de inercia ya fue calculado anteriormente.

$$J_{Pol/Motor} = 0,00000131 \text{ Kg.m}^2$$

▪ Reductor:

Preseleccionando los mismos reductores que el usado en el eje vertical, NemaTRUE NT23-005:

$$J_{Red/Motor} = 0,025 \text{ Kg.cm}^2 = 0,0000025 \text{ Kg.m}^2$$

▪ Total:

Se tendrá entonces un momento de inercia total respecto de los motores de:

$$J_T = J_{Carga/Motor} + 4 \cdot J_{Pol/Pol} + 2 \cdot J_{Red/Motor}$$
$$J_T = 0,0004481 \text{ Kg.m}^2$$

Luego, se tendrá un torque inercial de:

$$T_{Inercial} = \frac{2 \cdot \pi / p_m}{n} \cdot J_T \cdot \frac{df}{dt}$$
$$T_{Inercial} = \frac{2 \cdot \pi / 200}{2} \cdot 0,0004481 \text{ Kg.m}^2 \cdot 90000 \frac{1}{\text{Seg}^2} = 0,64 \text{ N.m}$$

TORQUE DEL MOTOR T_{Motor} :

Considerando:

$F_S = 2$ Factor de seguridad

$\eta_T = 97\%$ Rendimiento de la transmisión por correa síncrona

$\eta_R = 93\%$ (De catálogo calculado para el 100% del torque) Rendimiento del reductor (aproximación)

$\eta_M = 1$ Rendimiento del motor (aproximación)

Se obtiene un torque aproximado por cada motor de:

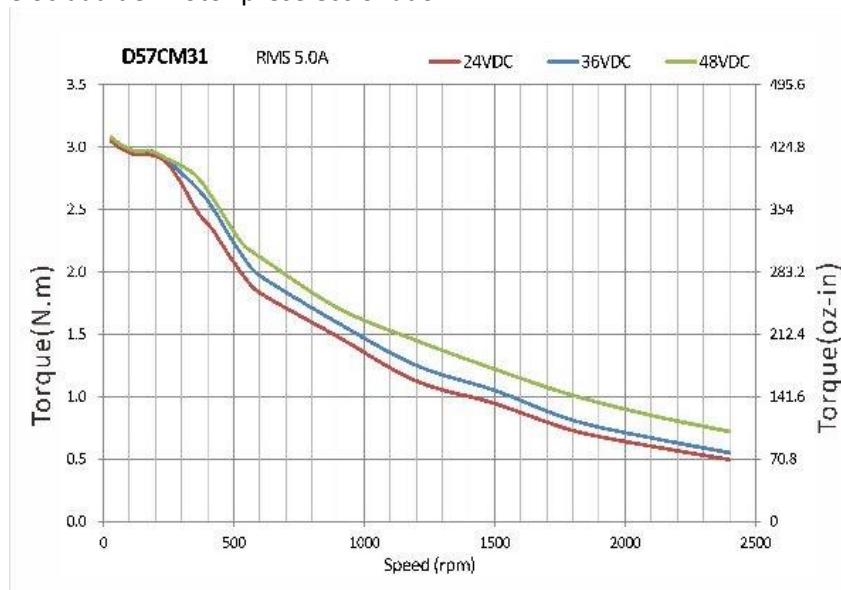
$$T_{Mot} = \frac{2}{3} \cdot (0,91 \text{ N.m} + 0,64 \text{ Nm}) \cdot \frac{2}{0,97 \cdot 0,93 \cdot 1} = 2,29 \text{ N.m} = 229 \text{ N.cm}$$

3) Preselección del motor

A partir del torque requerido calculado y considerando que no se tuvo en cuenta la inercia del motor, el cual posee un valor considerable, se decide preseleccionar un motor Nema 23 de 3,1 Nm. Se preselecciona:

- Motor paso a paso marca Leadshine modelo D57CM31
- Tamaño de brida: NEMA 23
- Nº de cables 4
- Holding Torque: 3,1 N.m
- Corriente: 6 A/fase
- Tension: 1,56 V
- Momento de inercia: 0,84 Kg.cm²
- Diámetro de los ejes: 8 mm
- Long. eje principal: 19,4 mm
- Peso: 1,4 Kg

Curva Torque/Velocidad del motor preseleccionado:



En este caso el fabricante tampoco indica la configuración de micropasos utilizada en el gráfico, por lo que se considera realizado con pasos completos siguiendo el mismo razonamiento indicado en la preselección del motor del eje vertical.

Para este caso el motor girará a 187,5 RPM en configuración de medios pasos, por lo que, a partir del gráfico, el motor entregaría menos de 2,9 Nm de torque. Este valor se encuentra lo suficientemente por encima de los 2,24 Nm para asegurar que el torque entregado por el motor superaría al torque requerido.

4) Preselección de las correas y poleas

Utilizando el software Design Flex Pro con los siguientes datos:

- Correas evaluadas: PowerGrip GT3, GT4, GTX y HTD
- Torque: 3,1 N.m
- Velocidad: 187,5 RPM
- Potencia del motor: 61 W (A partir de torque y la velocidad de rotación)
- Eficiencia del motor: 1



- Relación del reductor de velocidad: 5
- Eficiencia del reductor de velocidad: 93%
- Factor de servicio: 1,3
- Distancia entre ejes: 2250 mm $\pm$ 25%

Con esto se obtienen las siguientes alternativas:

Línea de prod.	Correa	Motriz (Dientes)	Conducida (Dientes)	RPM conducida	Δ RPM	sobredimens.	Coste relativo Δ	Entre centros (mm)	Δ dist. entre ejes (mm)	F. Tracc (kgf)	Ancho (mm)
PowerGrip GT4 4400-8MGT4-20		22	22	37,5	0	2,4	1,0	2.112	-138	68	40
PowerGrip GT4 4400-8MGT4-20		24	24	37,5	0	3,0	1,0	2.104	-146	62	40
PowerGrip GT4 4400-8MGT4-20		26	26	37,5	0	3,5	1,0	2.096	-154	57	40
PowerGrip GT4 4400-8MGT4-20		28	28	37,5	0	4,0	1,0	2.088	-162	53	40
PowerGrip GT4 4400-8MGT4-20		30	30	37,5	0	4,6	1,0	2.080	-170	50	40

En este punto todas las alternativas utilizan correa tipo GT4 de 8mm de paso y 20 mm de ancho. Se tiene en cuenta entonces el diámetro de las poleas a utilizar, siendo la menor de esta de 22 dientes con un diámetro primitivo de 56,02 mm. Según el catálogo de Gates estas correas se montan sobre poleas de perfil HTD.

Correa preseleccionada:

- Correa: PowerGrip GT4 – 8MGT
- Ancho: 20 mm
- Paso: 8 mm
- Longitud: 4708 mm
- Fuerza de tracción: 663 N
- Sobredimensionamiento: 2,4
- Cuerdas: Fibra de vidrio
- Peso específico: 97 gr/m
- Peso total: 457 gr

La fuerza de tensionado se toma solo como referencia, ya que al poseer la transmisión real una distancia entre ejes aproximadamente 138 mm mayor, puede ser necesario una mayor fuerza de tensado. Eso se comprobará en el momento de construcción de la maquina según su funcionamiento.

Poleas preseleccionadas:

- Modelo: HTD 22-8M-20-5F
- Nº de dientes: 22
- Diámetro primitivo: 56,02 mm
- Diámetro correa: 54,65 mm
- Diámetro extensión: - mm
- Ancho correa: 28 mm
- Ancho total: - mm
- Pestañas: 2
- Material: Acero
- Peso: 240 gr
- Casquillo cónico 1008



5) Nueva determinación del torque del motor T_{Motor} :

NOTA: Se realizaron varias modificaciones e iteraciones variando el torque del motor, relación de transmisión, diámetro y tipo de poleas llegando a la solución aquí presentada la cual se calcula a modo de verificación con los siguientes parámetros:

- Motor preseleccionado.
- Polea preseleccionada ($D_{Prim} = 56,02 \text{ mm}$).
- Correa preseleccionada.
- Reductor relación $i = 10$

En este caso debe tenerse en cuenta que la polea preseleccionada es de diámetro primitivo 38,2 mm, mayor a la adoptada en el paso 1 por lo tanto se modificará el torque necesario.

Se tendría entonces un avance por paso de:

$$Av_p = \frac{\pi \cdot 56,02 \text{ mm}}{200 \cdot 2 \cdot 10} \approx 0,044 \text{ mm}$$

Con lo que se obtiene una precisión de 0,022 mm lo cual se considera una precisión elevada y por lo tanto aceptable.

Al cambiar la relación de transmisión i , la velocidad angular en la polea motora será ahora de:

$$\omega = \frac{1}{n \cdot p_m \cdot i} \cdot \frac{60000}{\Delta t} = \frac{1}{2 \cdot 200 \cdot 10} \cdot \frac{60000}{0,8 \text{ mSeg}} = 18,75 \text{ RPM}$$

Y el tiempo aproximado para recorrer la Distancia Útil Vertical, estimada en 2110 mm, será de:

$$t_{Dist \text{ Util}} = \frac{Distancia \text{ util}}{Av_p} \cdot \Delta t = \frac{2110 \text{ mm}}{0,044 \text{ mm}} \cdot 0,8 \text{ mSeg} \approx 39 \text{ Seg}$$

TORQUE CONSTANTE DE LA CARGA T_{Carga} :

- FUERZA PARALELA A LOS EJES F_p

El valor de esta fuerza no dependerá del motor ni del sistema de transmisión, por lo tanto no se modifica:

$$F_p = 0$$

- FUERZA DE FRICCIÓN F_f

I. Fuerzas máxicas normales a los ejes F_{MN} :



El valor de esta fuerza no dependerá del motor ni del sistema de transmisión, por lo tanto no se modifica:

$$F_{MN} = 295 \text{ N}$$

II. Fuerza de arrastre del viento F_{ARR} :

El valor de esta fuerza no dependerá del motor ni del sistema de transmisión, por lo tanto no se modifica:

$$F_{ARR} = 88,3 \text{ N}$$

III. Fuerzas debido a la excentricidad entre el centro de gravedad de los elementos móviles y el punto de sujeción de la correa F_{Ex} :

El valor de esta fuerza no dependerá del motor ni del sistema de transmisión, por lo tanto no se modifica:

$$F_{ex} = 27,9 \text{ N}$$

Luego, el torque constante de la carga T_{Carga} se modificará por modificarse el diámetro primitivo de la polea y la relación de transmisión. Será entonces:

$$T_{Carga} = (F_P + (F_{MN} + F_{ARR} + 2 \cdot F_{Ex}) \cdot \mu) \cdot \frac{D_{prim.}}{2} \cdot \frac{1}{i}$$

$$T_{Carga} = (0 + (295 \text{ N} + 88,3 \text{ N} + 2 \cdot 27,9 \text{ N}) \cdot 0,54) \cdot \frac{0,05602 \text{ mm}}{2} \cdot \frac{1}{10} = 0,67 \text{ N.m}$$

TORQUE INERCIAL $T_{Inercial}$:

▪ Carga:

Como la fuerza para moverla se transmitirá por las correas y a su vez estas se acoplarán al motor mediante reductores de relación i , el valor del momento de inercia total del eje vertical respecto de los ejes de los motores será de:

$$J_{Carga/Motor} = masa \cdot \left(\frac{D_{prim.}}{2} \right)^2 \cdot \frac{1}{i^2} = 30 \text{ Kg} \cdot \left(\frac{0,05602 \text{ m}}{2} \right)^2 \cdot \frac{1}{10^2} = 0,0002354 \text{ Kg.m}^2$$

▪ Correa:

$$J_{Correa/Motor} = masa \cdot \left(\frac{D_{prim.}}{2} \right)^2 \cdot \frac{1}{i^2} = 0,097 \frac{\text{Kg}}{\text{m}} \cdot (4,708 \text{ m}) \cdot \left(\frac{0,05602 \text{ m}}{2} \right)^2 \cdot \frac{1}{10^2} = 0,0000036 \text{ Kg.m}^2$$

▪ Poleas:



Para el caso de poleas con casquillo cónico, como simplificación se consideran construidas de una sola pieza maciza. Para poleas de acero ($\rho = 7800 \text{ Kg/m}^3$) la inercia de cada una de ellas respecto del motor será:

$$J_{Pol/Motor} = \frac{\pi \cdot \rho}{32} \cdot D^4 \cdot L \cdot \frac{1}{i^2} = \frac{\pi \cdot 7800 \text{ kg/m}^3}{32} \cdot (0,05465 \text{ m})^4 \cdot 0,028 \text{ m} \cdot \frac{1}{10^2}$$

$$J_{Pol/Motor} = 0,0000019 \text{ Kg} \cdot \text{m}^2$$

- Motor:

Según catalogo:

$$J_{Motor} = 0,84 \text{ Kg} \cdot \text{cm}^2 = 0,000084 \text{ Kg} \cdot \text{m}^2$$

- Reductor:

Se selecciona el reductor de velocidad en función de tamaño de brida del motor. En este caso se selecciona el reductor de velocidad NemaTRUE NT23-010:

$$J_{Red/Motor} = 0,016 \text{ Kg} \cdot \text{cm}^2 = 0,0000016 \text{ Kg} \cdot \text{m}^2$$

- Total:

Se tendrá entonces un momento de inercia total respecto del motor de:

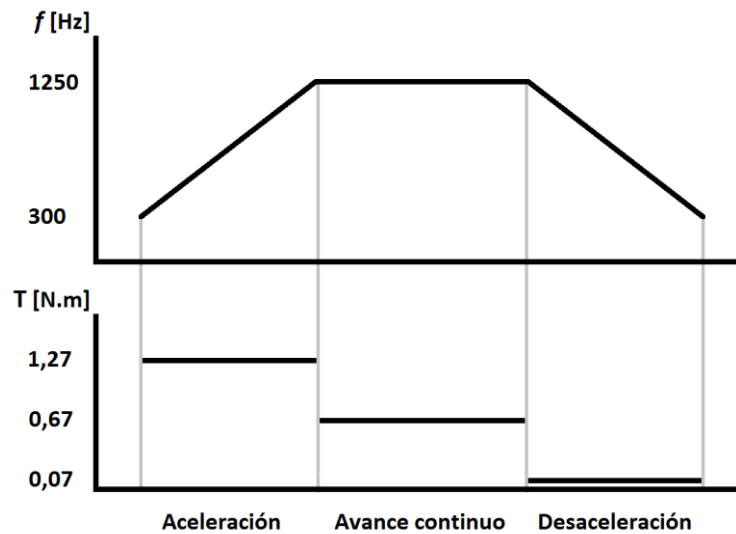
$$J_T = J_{Carga/Motor} + 2 \cdot J_{Correa/Motor} + 4 \cdot J_{Pol/Pol} + 2 \cdot J_{Motor} + 2 \cdot J_{Red/Motor}$$
$$J_T = 0,0004214 \text{ Kg} \cdot \text{m}^2$$

Luego, se tiene un torque inercial de:

$$T_{Inercial} = \frac{2 \cdot \pi \cdot p_m}{n} \cdot J_T \cdot \frac{df}{dt}$$

$$T_{Inercial} = \frac{2 \cdot \pi \cdot 200}{2} \cdot 0,0004214 \text{ Kg} \cdot \text{m}^2 \cdot 90000 \frac{1}{\text{Seg}^2} = 0,60 \text{ N} \cdot \text{m}$$

Perfil de frecuencia del tren de pulsos y toques en cada etapa del movimiento.



TORQUE DEL MOTOR T_{Motor} :

Considerando:

$F_S = 2$ Factor de seguridad

$\eta_T = 97\%$ Rendimiento de la transmisión por correa síncrona

$\eta_R = 93\%$ (De catálogo calculado para el 100% del torque)

$\eta_M = 1$ Rendimiento del motor (aproximación)

Se obtiene un torque aproximado por cada motor de:

$$T_{Mot} = \frac{2}{3} \cdot (0,67 \text{ N.m} + 0,60 \text{ Nm}) \cdot \frac{2}{0,97 \cdot 0,93 \cdot 1} = 1,88 \text{ N.m} = 188 \text{ N.cm}$$

6) Verificación del motor preseleccionado:

A partir del nuevo valor de torque calculado y encontrándose este lo suficientemente alejado al valor del torque entregado por el motor obtenido de la curva $T - \omega$ del mismo (mostrada en el punto 3), se verifica el motor preseleccionado de $3,1 \text{ N.m}$ marca Leadshine modelo D57CM31.

7) Nueva selección de correas y poleas:

Utilizando el software Design Flex Pro con los mismos datos utilizados para la preselección del punto 4) pero cambiando la relación de transmisión de 5 a 10 se obtienen las siguientes alternativas:



Línea de prod.	Correa	Motriz (Dientes)	Conducida (Dientes)	RPM conducida	Δ RPM	sobredimens.	Coste relativo Δ	Entre centros (mm)	Δ dist. entre ejes (mm)	F. Tracc (kgf)	Ancho (mm)
PowerGrip GT4	4400-8MGT4-20	22	22	18,8	0	1,2	1,0	2.112	-138	135	40
PowerGrip GT4	4400-8MGT4-20	24	24	18,8	0	1,5	1,0	2.104	-146	124	40
PowerGrip GT4	4400-8MGT4-20	26	26	18,8	0	1,7	1,0	2.096	-154	114	40
PowerGrip GT4	4400-8MGT4-20	28	28	18,8	0	2,0	1,0	2.088	-162	106	40
PowerGrip GT4	4400-8MGT4-20	30	30	18,8	0	2,3	1,0	2.080	-170	99	40
PowerGrip GT4	4400-8MGT4-20	32	32	18,8	0	2,5	1,0	2.072	-178	93	40
PowerGrip GTX	4400-8MX-20	22	22	18,8	0	3,1	1,3	2.112	-138	135	40
PowerGrip GTX	4400-8MX-20	24	24	18,8	0	3,6	1,3	2.104	-146	124	40
PowerGrip GTX	4400-8MX-20	26	26	18,8	0	4,0	1,3	2.096	-154	114	40
PowerGrip GTX	4400-8MX-20	28	28	18,8	0	4,5	1,3	2.088	-162	106	40
PowerGrip GTX	4400-8MX-20	30	30	18,8	0	4,9	1,3	2.080	-170	99	40

Puede verse que la primera alternativa es la transmisión ya preseleccionada y por lo tanto se verifica su selección, solo que en este caso la fuerza de tracción subirá de 663 N a 1326 N y el sobredimensionamiento bajará de 2,4 a 1,2.

Las poleas se mantienen sin variación y por lo tanto se verifican las ya preseleccionadas.

Las características de los motores y transmisiones seleccionados para los ejes horizontales se adjuntan en los anexos A.4, A.5 y A.18.

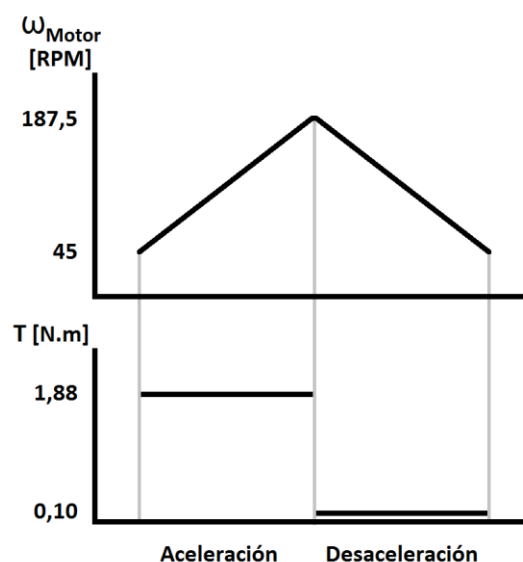
8) Cálculo del torque de retención necesario:

En este caso, en principio, no se requerirá de un par de retención para mantener el eje vertical en su posición cuando el mismo no se esté desplazando, ya que las fuerzas de gravedad y arrastre del viento serán contrarrestadas directamente por los ejes. De todas formas, el viento podría ejercer fuerzas de arrastre paralelas a los ejes horizontales cuando el eje vertical se encuentre desplazado del centro del túnel, ya que a ambos lados de la chapa de protección vertical el viento aplicará diferentes cargas. Esta diferencia de cargas puede no ser suficiente para desplazar el eje vertical pero si para generar vibraciones que puedan lentamente modificar la posición del eje si el mismo se encontrara libre. Por ello es necesario un par de retención cuando los motores se encuentren detenidos, aunque su valor puede ser considerablemente menor al par necesario para el desplazamiento.

9) Verificación del reductor de velocidad:

Se verifica siguiendo el método exacto se selección indicado por el fabricante en el catálogo adjunto en el anexo A.1. Aquí el funcionamiento más desfavorable se producirá cuando cada desplazamiento esté compuesto solamente por las rampas de aceleración y desaceleración, sin intervalo de desplazamiento a velocidad constante como se muestra en la siguiente imagen:

En estos casos el torque motor T_{Mot} en las rampas de aceleración y desaceleración serán los obtenidos en el punto 5) habiéndoles aplicado el factor de seguridad y los rendimientos de la transmisión y el reductor.



- Velocidad media de entrada n_m :

Como se vio para el caso del eje vertical, las velocidades n_1 , n_2 y n_m serán:

$$n_1 = n_2 = n_m = \frac{(187,5 + 45) \text{ RPM}}{2} = 116,25 \text{ RPM}$$

- Torque equivalente T_{EQ} :

$$T_{EQ} = \sqrt[8,7]{\frac{(1,88 \text{ Nm})^{8,7}}{2} + \frac{(0,10 \text{ Nm})^{8,7}}{2}} = 1,74 \text{ N.m}$$

- Torque equivalente modificado T_{EQm} :

Considerando entre 1000 y 2500 ciclos por hora $Q = 0,9$. Se agrega aquí también el factor de seguridad 2 para contemplar posibles inexactitudes entre los cálculos y la construcción real del posicionador. Por lo que el torque equivalente modificado valdrá:

$$T_{EQm} = \frac{T_{EQ}}{Q} \cdot F_s = \frac{1,74 \text{ N.m}}{0,9} \cdot 2 = 3,86 \text{ N.m}$$

- Verificación de T_{EQm} respecto de T_r del reductor:

Para el modelo NT23-010 y una velocidad inferior a las 1000 RPM, $T_r = 7,1 \text{ N.m} > T_{EQm}$, por lo que el reductor verificará según el torque equivalente.

- Verificación de la fuerza radial:

En este caso por cuestiones de espacio la polea no podrá montarse directamente sobre el eje del reductor debiendo poseer un soporte independiente, por lo que no se transmitirá esfuerzo radial al reductor.

4.10 RODAMIENTOS

Para el dimensionamiento se siguió el procedimiento de selección indicado en el catálogo de rodamientos SKF. A partir del mismo se determinó la capacidad de carga dinámica C y estática C_0 mediante los cuales se seleccionaron los rodamientos.

Se dimensionó a partir de la polea motora utilizando luego el mismo tipo de rodamiento en la polea de retorno.

▪ POLEAS HORIZONTALES

Capacidad de carga dinámica C :

Como orientación, según catálogo SKF sección B.3 para las máquinas utilizadas intermitentemente o durante breves periodos se estima una expectativa de vida de los rodamientos entre 3000 y 8000 horas. Se adoptó para el cálculo el máximo valor del intervalo.

La vida nominal básica en millones de revoluciones con una confiabilidad del 90% L_{10} se obtiene mediante las horas de expectativa de vida y la velocidad de rotación mediante la expresión:

$$L_{10} = L_{10H} \cdot \frac{60 \cdot n}{10^6}$$

Dónde:

L_{10H} : Vida nominal básica en horas de funcionamiento con una confiabilidad del 90% [hs]

n : Velocidad de giro [RPM]

Entonces:

$$L_{10} = 8000 \text{ hs.} \cdot \frac{60 \cdot 18,75 \text{ RPM}}{10^6} = 9 \text{ mill}$$

Luego, la capacidad de carga dinámica básica C del rodamiento se obtienen mediante:

$$C = P \cdot L_{10}^{1/p}$$

Dónde:

P : Carga dinámica equivalente del rodamiento [KN]

$p = 3$ para rodamiento de bolas

Siendo P función de las cargas reales radiales F_r y axiales F_a aplicadas al rodamiento. La misma se obtiene mediante la expresión:

$$P = X \cdot F_r + Y \cdot F_a$$

Según catálogo de SKF se hacen las siguientes consideraciones:

- Por considerarse despreciables las posibles cargas axiales comparadas con la carga radial, para rodamientos rígidos de una hilera de bolas será $P = F_r$.
- Para transmisión por correa dentada debe afectarse el valor de la fuerza de tracción de la correa en un factor 1,3 por considerarse una tensión elevada en la correa (catálogo SKF sección B.3).

La fuerza de tracción de la correa horizontal será de 1326 N, la carga radial F_r en cada rodamiento será entonces de 0,663 KN. Luego:

$$C = (1,3 \cdot 0,663 \text{ KN}) \cdot (9 \text{ mill})^{1/3} = 1,8 \text{ KN}$$

Capacidad de carga estática C_0 :

La capacidad de carga estática básica requerida C_0 del rodamiento se obtienen mediante:



$$C_0 = S_0 \cdot P_0$$

Dónde:

P_0 : Carga estática equivalente del rodamiento.

S_0 : Factor de seguridad estática.

La carga estática equivalente P_0 se obtiene mediante la expresión:

$$P_0 = X_0 \cdot F_r + Y_0 \cdot F_a$$

Luego, para rodamientos de una hilera de bolas será:

$$P_0 = 0,6 \cdot F_r + 0,5 \cdot F_a$$

En este caso por ser $F_a = 0$

$$P_0 = 0,6 \cdot F_r < F_r \Rightarrow P_0 = F_r = 0,663 \text{ KN}$$

El factor de seguridad estática por tratarse de movimiento poco frecuente con certeza baja en cuanto al nivel de carga se adoptó $S_0 = 1$.

La capacidad de carga estática básica requerida C_0 deberá poseer entonces un valor mínimo de:

$$C_0 = S_0 \cdot P_0 = 0,663 \text{ KN}$$

Selección de rodamiento:

Teniendo en cuenta el momento del ensamblaje, el diámetro interno del rodamiento deberá ser mayor que el diámetro del extremo del eje, el cual se acoplará con la salida del reductor. El extremo del eje a acoplar será de 8 mm por lo que se adopta un diámetro interno del rodamiento de 9 mm. Luego, el rodamiento deberá poseer:

- C (mínimo) = 1,8 KN
- C_0 (mínimo) = 0,663 KN
- d (mínimo) = 9 mm

Se tuvo en consideración solamente rodamientos indicados como “Producto popular” y con sellos en ambos lados. Considerando luego los de menor dimensión en cuanto a diámetro exterior y espesor del rodamiento se selecciona entonces cualquiera de los siguientes modelos:

- 609-2RSH
- 609-2RSL
- 609-2Z

Los cuales poseen $C = 3,9 \text{ KN}$, $C_0 = 1,66 \text{ KN}$, $D_{ext} = 24 \text{ mm}$, $D_{int} = 9 \text{ mm}$ y espesor = 7 mm.

Las demás dimensiones necesarias para el dimensionamiento del eje y el soporte que lo contendrá pueden observarse en el anexo A.6.



▪ POLEAS VERTICALES

Capacidad de carga dinámica C :

La vida nominal básica en millones de revoluciones con una confiabilidad del 90% será igual a la calculada para los rodamientos horizontales, por lo que $L_{10} = 9 \text{ mill}$.

Siguiendo las mismas consideraciones que para los rodamientos horizontales $P = F_r$. En este caso la fuerza de tracción de la correa vertical será de 670 N, por lo que la carga radial F_r en cada rodamiento será entonces de 0,335 KN.

Luego, la capacidad de carga dinámica básica C del rodamiento se obtiene mediante:

$$C = (1,3 \cdot 0,335 \text{ KN}) \cdot (9 \text{ mill})^{1/3} = 0,91 \text{ KN}$$

Capacidad de carga estática C_0 :

Nuevamente partiendo de las mismas consideraciones que para el caso horizontal

$$P_0 = 0,6 \cdot F_r < F_r \Rightarrow P_0 = F_r = 0,335 \text{ KN}$$

Utilizando el mismo factor de seguridad estática $S_0 = 1$ la capacidad de carga estática básica requerida C_0 deberá poseer entonces un valor mínimo de:

$$C_0 = S_0 \cdot P_0 = 0,335 \text{ KN}$$

Selección de rodamiento:

En este caso se utilizarán poleas macizas, la motora se montará directamente sobre el eje del reductor planetario mientras que la de retorno girará sobre un eje mediante dos rodamientos. El diámetro externo de los rodamientos estará condicionado entonces por el máximo taladro que admiten estas poleas. Para las poleas 24-5MR-15 el taladro máximo será de 16mm. Luego, el rodamiento debe poseer:

- C (mínimo) = 0,91 KN
- C_0 (mínimo) = 0,335 KN
- D (máximo) = 16 mm

Se tuvo en consideración solamente rodamientos indicados como "Producto popular" y con sellos en ambos lados. Buscando que los valores de C y C_0 no se encuentren muy próximos a los mínimos requeridos, se selecciona entonces cualquiera de los siguientes modelos:

- 628/8-2RS1
- 628/8-2Z

Los cuales poseen $C = 1,33 \text{ KN}$, $C_0 = 0,57 \text{ KN}$, $D_{ext} = 16 \text{ mm}$, $D_{int} = 8 \text{ mm}$ y espesor = 5mm

Las demás dimensiones necesarias para el dimensionamiento del eje que los montará pueden observarse en el anexo A.7.



4.11 RESUMEN DE LAS TRANSMISIONES

	CARACTERÍSTICA	EJE VERTICAL	EJE HORIZONTAL (x2)
MOTOR	Marca y Modelo	LEADSHINE D57CM21	LEADSHINE D57CM31
	Torque detenido del motor [Nm]	2,1	3,1
	Torque requerido [Nm]	1,12	1,88
	Brida	Nema 23	Nema 23
	Nº de cables	4	4
	I en conexión Bipolar [A/fase]	5	6
	Tensión [V]	1,26	1,56
	Inercia del rotor [Kg.cm2]	0,57	0,84
	Diámetro del eje [mm]	8	8
	Longitud del eje [mm]	19,4	19,4
	Peso [Kg]	1,2	1,4
REDUCTOR	Marca	Micron	Micron
	Modelo	NemaTRUE NT23-005	NemaTRUE NT23-010
	Tipo y relación	Planetario (1:5)	Planetario (1:10)
CORREA	Marca	Gates	Gates
	Modelo	Power Grip GT3 – 5MGT	Power Grip GT4 – 8MGT
	Ancho [mm]	15	20
	Paso [mm]	5	8
	Longitud [mm]	3120	4708
	Fuerza de tracción [N]	670	1326
	Sobredimensionamiento	1,7	1,2
	Cuerdas	Fibra de vidrio	Fibra de vidrio
	Peso específico [gr/m]	61,5	97
	Peso total [gr]	192	457
POLEAS	Modelo	GT3 24-5MR-15-6F	HTD 22-8M-20-5F
	Nº de dientes	24	22
	Diámetro primitivo [mm]	38,2	56,02
	Pestañas	2	2
	Material	Acero	Acero
	Peso [gr]	180	240
	Tipo de buje	Polea maciza	Casquillo cónico 1008-12
	Rodamientos	628/8-2Z	609-2Z
PRECISIÓN Y TIEMPO DE DESPLAZ.	Avance por paso [mm]	0,06	0,044
	Error de posicionamiento [mm]	0,03	0,022
	Rango de detección sensor [mm]	0,2	0,2
	Tiempo de desplaz. total [Seg]	20	39

5. DIMENSIONAMIENTO DE EJES GUIA

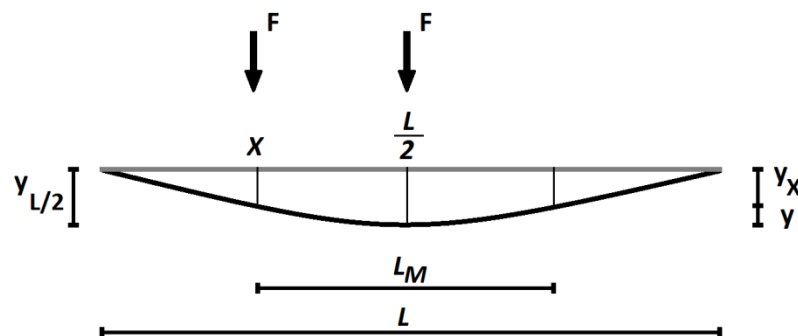
5.1 PROCEDIMIENTO PARA DIMENSIONAMIENTO DE LOS EJES

Ya que la finalidad del posicionador será la de ubicar con precisión al instrumento de medición, debe esperarse un mínimo de deformación de los ejes, ya que de otra manera sería inútil utilizar mecanismos de precisión como motores paso a paso cuando la misma estructura con su deformación alejaría grandemente el punto de medición real del punto deseado. Esta consideración se realiza especialmente para los ejes horizontales ya que la deformación de estos será transversal al túnel, lo cual introduciría mayor error que las deformaciones axiales producidas en el eje vertical.

Por lo tanto la selección del diámetro de los ejes se realizó en función a la deformación, verificándose luego la resistencia mecánica.

▪ DETERMINACIÓN DE LA DEFORMACIÓN Y DIMENSIONAMIENTO DE EJES

En este caso lo importante es la diferencia de deflexión entre los puntos de medición de un determinado ensayo. Como puede observarse en la figura siguiente, al desplazarse el instrumento de medición (y por ello la carga puntual F) dentro de un intervalo de medición L_M en un ensayo, la deflexión pasará del valor y_x cuando el posicionador se encuentre en el extremo del intervalo de medición, al valor $y_{L/2}$ cuando se encuentre en el centro del eje. Luego la deflexión de interés será la diferencia y entre las anteriores ya que esta determinará el error de posición por deflexión del ensayo.



Por lo tanto, considerando un intervalo de medición L_M debe determinarse la diferencia de deflexión cuando la carga puntual F se encuentre en el extremo del intervalo a una distancia $x = (L - L_M)/2$ del extremo del eje y cuando la carga se encuentre en el centro del eje a la distancia $L/2$ del extremo del mismo. Debe tenerse también en cuenta para el cálculo de las deformaciones a las cargas distribuidas aplicadas sobre el eje las cuales no dependen de la posición del posicionador.

Para la determinación de las deflexiones debe tenerse en cuenta que se tratará de un caso de flexión compuesta producto de la combinación de una fuerza axial de compresión S , fuerzas transversales puntuales F y/o distribuidas P y Q . En este caso, debe determinarse las deflexiones considerando únicamente las cargas transversales, para luego, afectar los valores calculados por un coeficiente que tiene en cuenta la acción de la carga axial S . De esta manera, las deflexiones en los puntos x y $L/2$ se determinarán mediante:

$$y_x = y_{0x} \cdot C$$

$$y_{L/2} = y_{0L/2} \cdot C$$



$$y = (y_{0L/2} - y_{0x}) \cdot C$$

Dónde:

y_{0x} e $y_{0L/2}$: Deflexiones producidas únicamente por las cargas transversales.

C : Coeficiente de corrección que refleja la influencia de la carga de compresión axial.

Por el principio de superposición lineal las deflexiones y_{0x} e $y_{0L/2}$ estarán compuestas por la suma de las deflexiones y_F debido a la carga puntual F , la deflexión y_P debido a la carga distribuida total P producto del empuje ejercido por el viento sobre el eje y la deflexión y_Q debido a la carga distribuida total Q producto del peso propio del eje. Luego:

$$y_{0x} = y_{Fx} + y_{Px} + y_{Qx}$$

$$y_{0L/2} = y_{FL/2} + y_{PL/2} + y_{QL/2}$$

Así, las deflexiones debidas a cada tipo de carga considerando los extremos empotrados vienen dadas por las expresiones:

- Carga puntual F aplicada a una distancia x del extremo del eje:

$$y_{Fx} = \frac{F \cdot (L - x)^3 \cdot x^3}{3 \cdot E \cdot J \cdot L^3} \text{ (Deflexion en el punto } x)$$

$$y_{FL/2} = \frac{F \cdot L^3}{192 \cdot E \cdot J} \text{ (Deflexion en el punto } x = L/2)$$

- Carga total P distribuida uniformemente:

$$y_{Px} = \frac{P \cdot x^2}{24 \cdot E \cdot J \cdot L} (L - x)^2 \text{ (Deflexion en el punto } x)$$

$$y_{PL/2} = \frac{P \cdot L^3}{384 \cdot E \cdot J} \text{ (Deflexion en el punto } x = L/2)$$

- Carga total Q distribuida uniformemente:

$$y_{Qx} = \frac{Q \cdot x^2}{24 \cdot E \cdot J \cdot L} (L - x)^2 \text{ (Deflexion en el punto } x)$$

$$y_{QL/2} = \frac{Q \cdot L^3}{384 \cdot E \cdot J} \text{ (Deflexion en el punto } x = L/2)$$

Dónde:

E : Módulo de elasticidad.

J : Momento de inercia mínimo de la sección del eje.

L : Longitud del eje.



NOTA: Si bien los extremos de los ejes podrían considerarse empotrados por el tipo de sujeción de los mismos, los soportes de estos también sufrirán deformación por lo que el estado de empotramiento no se cumplirá en realidad. Aun así la deformación real sería muy próxima a esta, por lo tanto se considerará este tipo de sujeción en los extremos a fin de obtener las deformaciones y tensiones aproximadas.

Por su parte el coeficiente C se obtiene de la expresión:

$$C = \frac{1}{1 - \frac{S}{S_E}}$$

Siendo S la fuerza axial aplicada al eje y S_E la fuerza de Euler determinada mediante:

$$S_E = \frac{\pi^2 \cdot E \cdot J}{(\mu \cdot l)^2}$$

Siendo μ el coeficiente de reducción de la longitud, parámetro que depende del tipo de apoyo de los extremos.

Se determinó así la deformación que sufrirían los ejes para distintos diámetros considerando como intervalo de medición L_M a la Distancia Útil de desplazamiento en cada dirección. A partir de ello se seleccionó los diámetros de eje a utilizar.

▪ VERIFICACIÓN DE LA RESISTENCIA MECÁNICA

En este caso el momento flector máximo aplicado en los soportes (sección más solicitada) se determina por la fórmula:

$$M_{Max} = M_F + M_P + M_Q + S \cdot y_{L/2}$$

Dónde:

$$M_F = \frac{F \cdot L}{8} \text{ (Carga puntual F en su centro)}$$

$$M_P = \frac{P \cdot L}{12} \text{ (Carga total P distribuida uniformemente)}$$

$$M_Q = \frac{Q \cdot L}{12} \text{ (Carga total Q distribuida uniformemente)}$$

Luego las tensiones normales máximas vienen dadas por:

$$|\sigma_{max}| = \frac{S}{A} + \frac{M_{Max}}{W}$$

Siendo:

A : Área de la sección transversal de la barra

W : Modulo axial de la sección

$$|\sigma_{max}| = \frac{S}{A} + \frac{M_F}{W} + \frac{M_P}{W} + \frac{M_Q}{W} + \frac{S \cdot y_{L/2}}{W \cdot \left(1 - \frac{S}{S_E}\right)}$$

Esta expresión indica la no linealidad de la carga y las tensiones, ya que las tensiones crecen más rápidamente que las cargas. Por ello la resistencia mecánica no podrá verificarse mediante el cálculo de tensiones admisibles si no por el cálculo de cargas admisibles. Para ello se obtiene el factor de seguridad efectivo F_{SE} el cual indicará cuantas veces debería aumentarse las cargas para que las tensiones normales alcancen el límite de fluencia. Por ello, en el momento en el que σ_{max} alcance el valor de σ_{fl} la expresión anterior tendrá la forma:

$$|\sigma_{fl}| = \frac{S \cdot F_{SE}}{A} + \frac{M_F \cdot F_{SE}}{W} + \frac{M_P \cdot F_{SE}}{W} + \frac{M_Q \cdot F_{SE}}{W} + \frac{(S \cdot F_{SE}) \cdot (y_{0L/2} \cdot F_{SE})}{W \cdot \left(1 - \frac{S \cdot F_{SE}}{S_E}\right)}$$

Luego para determinar el factor de seguridad efectivo tiene que resolverse la ecuación cuadrática anterior.

5.2 EJES VERTICALES

5.2.1 DETERMINACIÓN DE LA SECCIÓN POR DEFORMACIÓN

▪ ESTADO DE CARGAS

En este caso las cargas serán diferentes en cada eje, por ello se analizó por separado la deformación y la resistencia mecánica en cada uno de ellos a fin de determinar cuál sería el más afectado. Para esto se hicieron las siguientes suposiciones:

- La fuerza de arrastre ejercida por el viento sobre el porta instrumento se transmitirá a los ejes verticales a través de los bujes. Si bien el porta instrumento se sujetará a los dos ejes verticales, según cual sea la precisión con que se construyan las piezas de la máquina y el ajuste dado a estos bujes, la fuerza de arrastre podría no transmitirse de forma pareja entre ambos ejes, y en caso extremo incluso transmitirse a solo uno de ellos. Por lo tanto se dimensionó considerando esta fuerza aplicada en su totalidad en cada uno de los ejes verticales.
- Uno de los ejes verticales se encontrará protegido de la acción del viento al estar ubicado dentro de la chapa de protección vertical, en cambio el otro se encontrará expuesto a la acción del viento. Si bien este segundo eje se ubicará detrás de la chapa de protección, se encontrará lo suficientemente apartado de esta como para considerarse protegido en su totalidad, por lo tanto el viento incidirá en mayor o menor medida sobre el mismo y por ello esta carga debe tenerse en cuenta en el cálculo. Por no conocer la intensidad que tendrá el viento sobre este eje se consideró como si el mismo estuviera completamente desprotegido, es decir, como si estuviera expuesto a la acción directa del viento.
- Debido a la posición vertical de los ejes el peso propio de los mismos no incidirá en la deformación transversal y por lo tanto no se lo tuvo en cuenta.
- La carga axial S será debida a la tensión de la correa y el porcentaje de esta tensión que recaerá sobre cada eje dependerá de la distancia de estos al punto de sujeción de la polea de retorno.



A partir de las consideraciones anteriores se entiende que la deformación calculada en cada eje será en exceso y supondrá un valor máximo que en la práctica no se llegaría a alcanzar.

Llamando 1er eje al eje al que se encuentra primero en la dirección del viento y 2do eje al que se encuentra después, se tendrán los siguientes estados de carga:

1er eje:

- Carga puntual $F \neq 0$: Valor igual a la fuerza de arrastre sobre el porta instrumentos.
- Carga distribuida $P = 0$: Por encontrarse protegido por la chapa de protección vertical.
- Carga distribuida $Q = 0$: Por posición vertical del eje.
- Carga axial $S \neq 0$: Valor que depende de la distancia del eje al punto de sujeción de la correa.

2do eje:

- Carga puntual $F \neq 0$: Valor igual a la fuerza de arrastre sobre el porta instrumentos.
- Carga distribuida $P \neq 0$: Valor igual a la fuerza de arrastre sobre el eje.
- Carga distribuida $Q = 0$: Por posición vertical del eje.
- Carga axial $S \neq 0$: Valor que depende de la distancia del eje al punto de sujeción de la correa.

Carga puntual F :

La carga puntual F será la fuerza de arrastre sobre el porta instrumento y que se transmitirá a los ejes. Esta carga ya se determinó en el capítulo 4.8 y poseerá un valor de:

$$F = F_{Arr\ Vert} = 4,8\text{ N}$$

Carga distribuida P :

La carga distribuida P será la fuerza de arrastre que ejerce el viento sobre el 2do eje. Como se indicó en el capítulo 4.8 esta fuerza puede determinarse mediante la expresión:

$$P = \frac{1}{2} \cdot C_D \cdot \rho \cdot V^2 \cdot A$$

Dónde:

P : Fuerza de arrastre sobre el cuerpo en [N]

C_D : Coeficiente de arrastre adimensional que depende de la geometría del cuerpo y el número de Reynolds.

ρ : Densidad del fluido en [Kg/m³]

V : Velocidad del fluido en [m/seg]

A : Área del cuerpo proyectada sobre un plano normal a la dirección del flujo en [m²]

El coeficiente de arrastre C_D sobre un cilindro se obtuvo mediante la Figura 8.9 y la Tabla 8.1 del libro Mecánica de los Fluidos, M. Potter y D. Wiggert, 3ra Ed en función del número de Reynolds, el diámetro y longitud del cilindro. El número de Reynolds para un cilindro se obtiene como:

$$R = \frac{V \cdot D}{\nu}$$

Dónde:

ν : Viscosidad cinemática del fluido en [m²/seg]

D : Diámetro del cilindro en [m]

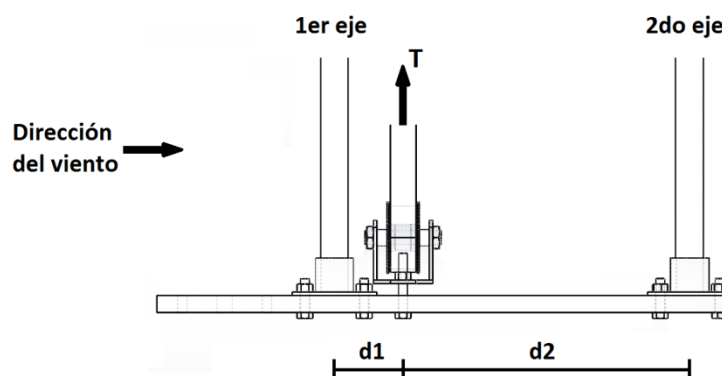
De la figura 8.9 del libro para un rango de Re entre $1,5E-4$ y $5E-4$ corresponde un valor constante de $C_{D\infty} = 1,2$. Este es el coeficiente de arrastre para un cilindro de longitud infinita. Para una longitud finita el mismo debe corregirse según la tabla 8.1, donde el factor de corrección depende del cociente L/D . Ahora bien, por encontrarse los extremos de los ejes sujetos a una superficie sólida debe utilizarse como longitud el doble de la longitud real de la barra, por lo que el coeficiente de corrección dependerá del cociente $2.L/D$. Para una longitud aproximada de $L = 1630$ mm con ejes de 16 mm se tendría un cociente $2.L/D = 200$ con lo que, según la tabla, podrían considerarse factores de corrección de 1.

Si utilizamos la densidad del aire $\rho = 1,247$ kg/m³ a 10°C, su viscosidad cinemática a la misma temperatura $\nu = 1,42E - 5$ m²/seg, la velocidad máxima del viento dentro del túnel $V = 25$ m/seg y la longitud del eje vertical $L = 163$ cm, la fuerza de arrastre total P sobre ejes de diámetros 12, 15 y 16 mm será:

D [mm]	V [m/seg]	Visc Cin. [m ² /seg]	Re	Cd	Densidad [Kg/m ³]	L [m]	P [N]
12	25	1.42E-05	2.1E+04	1.2	1.247	1.63	9.1
15	25	1.42E-05	2.6E+04	1.2	1.247	1.63	11.4
16	25	1.42E-05	2.8E+04	1.2	1.247	1.63	12.2

Carga axial S :

En este caso la carga axial será producida por la fuerza de tensionado T de la correa vertical la cual se determinó en el capítulo 4.8 y poseerá un valor total de 670 N. Esta fuerza se distribuirá entre ambos ejes en función de la distancia de estos al punto de sujeción de la correa.



A partir de la figura y teniendo en cuenta las distancias $d_1 = 40,5$ mm y $d_2 = 169,5$ mm, la fuerza axial S aplicada en cada eje valdrá:

$$S_{1er\ eje} = \frac{d_2}{d_1 + d_2} * T = 540,8\ N$$

$$S_{2do\ eje} = \frac{d_1}{d_1 + d_2} * T = 129,2\ N$$



▪ DETERMINACIÓN DE LA SECCIÓN

En este caso se determinó la deformación para tres valores de diámetros de eje, 12, 15 y 16 mm. Las expresiones para determinar las deflexiones en la barra debido a cada tipo de carga fueron indicadas en el capítulo 5.1. Para la aplicación de las mismas se tuvieron en cuenta los siguientes parámetros:

- $L = 1630 \text{ mm}$ Longitud de cada eje.
- $L_M = 1435 \text{ mm} \Rightarrow x = 97,5 \text{ mm}$
- $E = 210 \text{ Gpa}$ Módulo de elasticidad del acero.
- $J = \pi \cdot D^4 / 64$ Momento de inercia de una sección circular.
- $\mu = 1$ Coeficiente de reducción de la longitud considerando extremos articulados¹.

El resumen de cargas y las deformaciones máximas en cada eje se indica en los cuadros siguientes.

1er EJE:

D [mm]	S [N]	S_E [N]	C []	F [N]	$y_{F x}$ [mm]	$y_{F L/2}$ [mm]	P [N]	$y_{P x}$ [mm]	$y_{P L/2}$ [mm]	Q [N]	$y_{Q x}$ [mm]	$y_{Q L/2}$ [mm]	y [mm]
12	540,8	794	3,14	4,8	0,006	0,507	0	0	0	0	0	0	1,570
15	540,8	1939	1,39	4,8	0,002	0,207	0	0	0	0	0	0	0,284
16	540,8	2510	1,27	4,8	0,002	0,160	0	0	0	0	0	0	0,202

2do EJE:

D [mm]	S [N]	S_E [N]	C []	F [N]	$y_{F x}$ [mm]	$y_{F L/2}$ [mm]	P [N]	$y_{P x}$ [mm]	$y_{P L/2}$ [mm]	Q [N]	$y_{Q x}$ [mm]	$y_{Q L/2}$ [mm]	y [mm]
12	129,2	794	1,19	4,8	0,006	0,507	9,1	0,024	0,483	0	0	0	1,145
15	129,2	1939	1,07	4,8	0,002	0,207	11,4	0,013	0,247	0	0	0	0,471
16	129,2	2510	1,05	4,8	0,002	0,160	12,2	0,010	0,204	0	0	0	0,371

Se observa así que el eje más deformado sería el segundo. Se seleccionan entonces para los ejes verticales diámetros de 16 mm.

5.2.2 VERIFICACIÓN DE LA RESISTENCIA MECÁNICA

Las expresiones para determinar los momentos y tensiones en los extremos de los ejes debido a cada tipo de carga fueron indicadas en el capítulo 5.1. A partir de ellas se obtuvieron los siguientes valores utilizando ejes de 16mm según lo seleccionado en el punto anterior:

¹ “Muy a menudo, incluso cuando los extremos de la barra están empotrados rígidamente a otros elementos, el cálculo se realiza considerando que los extremos están articulados, lo que va a favor de la reserva de resistencia.” Consideración extraída del libro Resistencia de Materiales – P. A. Stiopin - Capítulo 87.



$$M_F = \frac{F \cdot l}{8} = 0,987 \text{ Nm}$$

$$M_P = \frac{P \cdot l}{12} = 1,66 \text{ Nm}$$

$$M_Q = \frac{Q \cdot l}{12} = 0$$

$$M_S = S \cdot y_{L/2} \text{ (S e } y_{L/2} \text{ segun el eje analizado)}$$

$$M_{Total} = M_F + M_P + M_Q + S \cdot y_{L/2}$$

$$J = \frac{\pi \cdot D^4}{64} = 3,217 \text{ E} - 9 \text{ m}^4$$

$$W = \frac{\pi \cdot D^3}{32} = 4,02 \text{ E} - 7 \text{ m}^3$$

Considerando ejes de acero al carbono 1010 trefilado $\sigma_{fl} = 300 \text{ Mpa}$. Luego, las tensiones producidas por cada tipo de carga, el valor de la tensión máxima $\sigma_{Max} [\text{Mpa}]$ y el factor de seguridad efectivo F_{SE} en cada eje se resumen en la siguiente tabla:

	S [N]	M_F [Nm]	M_P [Nm]	M_Q [Nm]	M_S [Nm]	σ_S [Mpa]	σ_{MF} [Mpa]	σ_{MP} [Mpa]	σ_{MQ} [Mpa]	σ_{MS} [Mpa]	σ_{Max} [Mpa]	F_{SE}
Eje 1	540,8	0,978	0	0	0,11	2,69	2,43	0	0	0,27	5,40	4,56
Eje 2	129,2	0,978	1,66	0	0,05	0,64	2,43	4,12	0	0,12	7,32	16,18

Se observa así que los ejes, considerando un diámetro de 16 mm, verificarían a la resistencia mecánica.

Puede verse aquí la no linealidad entre las cargas y las tensiones totales en los ejes. Por ej. en el primer eje la tensión máxima sería 55 veces menor a la tensión de fluencia, sin embargo si las cargas aumentaran solo 4,56 veces, las tensiones en este eje alcanzarían el límite de fluencia.

5.3 EJES HORIZONTALES

5.3.1 DETERMINACIÓN DE LA SECCIÓN POR DEFORMACIÓN

■ ESTADO DE CARGAS

Aquí las cargas también serán diferentes en cada eje, por ello se analizaron por separado la deformación y la resistencia mecánica en cada uno de ellos a fin de determinar cuál sería el más afectado. Para esto se hicieron las siguientes suposiciones:

- La fuerza de arrastre ejercida por el viento sobre el eje vertical se transmitirá a los ejes horizontales mediante los bujes. Si bien el eje vertical se sujetará a los cuatro ejes guía horizontales (un par en el



eje superior y otro par en el inferior), según cual sea la precisión con que se construyan las piezas de la máquina y el ajuste dado a estos bujes, la fuerza de arrastre podría no transmitirse de forma pareja. Por ello como seguridad se consideró que la fuerza de arrastre sobre el eje vertical se distribuirá uniformemente solamente entre dos de los ejes, uno de la parte superior y otro de la inferior.

- Se desprecia la fuerza de arrastre ejercida por el viento sobre los ejes horizontales. En el caso de los primeros ejes horizontales por encontrarse protegidos por las chapas de protección horizontales y en el caso de los segundos ejes por encontrarse próximos al piso o techo del túnel donde por efecto de la viscosidad la velocidad del viento disminuye.
- Tal como se hizo para el dimensionamiento de la transmisión horizontal, se considera que el peso del eje vertical no se distribuirá equitativamente sobre el par de ejes superior e inferior, si no que un par de ejes soportará $2/3$ del peso total. Luego, la porción de esta carga que recaerá sobre el primer y el segundo eje dependerá de la distancia de estos al centro de gravedad del eje vertical.
- Debido a la posición horizontal de los ejes debe tenerse en cuenta el peso propio de los mismos.
- La carga axial S será debida a la tensión de la correa y el porcentaje de esta tensión que recaerá sobre cada eje dependerá de la distancia de estos al punto de sujeción de la polea de retorno.

A partir de las consideraciones anteriores se entiende que la deformación calculada en cada eje será en exceso y supondrá un valor máximo que en la práctica no se llegaría a alcanzar.

Llamando 1er eje al eje al que se encuentra primero en la dirección del viento y 2do eje al que se encuentra después, se tendrá para ambos ejes los siguientes estados de carga:

- Carga puntual $F \neq 0$: Producto de la fuerza de arrastre sobre el eje vertical y el peso propio del eje vertical. El valor aplicado sobre cada eje dependerá de la distancia de estos al centro de gravedad del eje vertical.
- Carga distribuida $P = 0$: Por encontrarse protegidos por la chapa de protección vertical y proximidad al techo y piso.
- Carga distribuida $Q \neq 0$: Por posición horizontal de los ejes.
- Carga axial $S \neq 0$: El valor aplicado sobre cada eje dependerá de la distancia de estos al punto de sujeción de la correa.

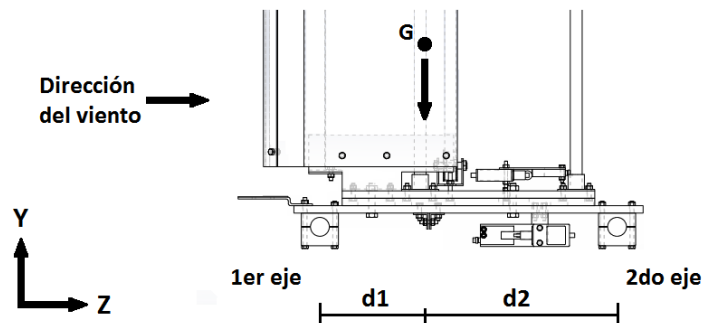
Cargas puntuales F_Z y F_Y :

La carga puntual poseerá dos componentes, F_Z producto del arrastre del viento la cual tendrá dirección axial respecto del túnel y F_Y producto del peso propio del eje vertical y los soportes centrales de cada eje horizontal.

La fuerza total de arrastre sobre el eje vertical fue determinada en el capítulo 4.9 y poseerá un valor de 88,3 N. Suponiendo que esta fuerza será resistida solamente por uno de los ejes horizontales superiores y uno de los inferiores, y teniendo en cuenta que por estabilidad la misma debería repartirse uniformemente entre ambos ejes, se tendrá entonces:

$$F_Z = \frac{88,3 \text{ N}}{2} = 44,15 \text{ N}$$

Por su parte la fuerza F_Y será debido al peso de todos los elementos móviles. Como se indicó en el capítulo 4.9 este peso será de aproximadamente 295 N. Como ya se mencionó, por seguridad se considera que 2/3 de este peso (197 N) se aplicará solo en un par de ejes (superior o inferior). El porcentaje de este peso que recaiga sobre cada eje de este par vendrá determinado por la distancia en dirección Z de los ejes al centro de gravedad del eje vertical.



En el grafico anterior $d_1 = 141 \text{ mm}$ y $d_2 = 259 \text{ mm}$. Luego:

$$F_{Y \text{ 1er eje}} = \frac{d_2}{d_1 + d_2} * 197 \text{ N} = 127,5 \text{ N}$$

$$F_{Y \text{ 2do eje}} = \frac{d_1}{d_1 + d_2} * 197 \text{ N} = 69,4 \text{ N}$$

Carga distribuida Q:

Esta carga será el peso del eje y por lo tanto dependerá del diámetro del mismo a utilizar. Para diámetros de 25, 27 y 28 mm se tienen los siguientes valores, considerando ejes de acero de 2350 mm de longitud.

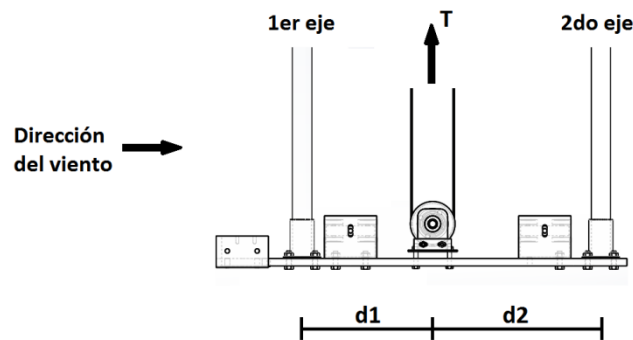
$$D = 25\text{mm} \Rightarrow Q = 88,8 \text{ N}$$

$$D = 27\text{mm} \Rightarrow Q = 103,6 \text{ N}$$

$$D = 28\text{mm} \Rightarrow Q = 111,4 \text{ N}$$

Carga axial S:

La carga axial será producida por la fuerza de tensionado T de las correas horizontales las cuales se determinaron en el capítulo 4.9 y poseerán un valor total de 1326 N. Estas fuerzas se distribuirán entre los ejes de cada par en función de la distancia de estos al punto de sujeción de la correa.



A partir de la figura y teniendo en cuenta las distancias $d_1 = 175 \text{ mm}$ y $d_2 = 225 \text{ mm}$, la fuerza axial S aplicada en cada eje valdrá:

$$S_{1er \text{ eje}} = \frac{d_2}{d_1 + d_2} * T = 745,9 \text{ N}$$

$$S_{2do \text{ eje}} = \frac{d_1}{d_1 + d_2} * T = 580,1 \text{ N}$$

▪ DETERMINACIÓN DE LA SECCIÓN

Se determinará la deformación en el 1er eje ya que este sería el más solicitado y por lo tanto presentaría la mayor deformación. En este caso se calculará por separado la deformación del eje en sentido vertical y horizontal (transversal y axial al túnel) para tres valores de diámetro, 25, 27 y 28 mm.

Las expresiones para determinar las deflexiones del eje debido a cada tipo de carga fueron indicadas en el capítulo 5.1. Para la aplicación de las mismas se tuvo en cuenta los siguientes parámetros:

- $L = 2350 \text{ mm}$ Longitud de cada eje.
- $L_M = 2110 \text{ mm} \Rightarrow x = 120 \text{ mm}$
- $E = 210 \text{ Gpa}$ E Modulo de elasticidad del acero.
- $J = \pi \cdot D^4 / 64$ Momento de inercia de una sección circular.
- $\mu = 1$ Coeficiente de reducción de la longitud considerando extremos articulados.

La deformación máxima en cada dirección y para los distintos diámetros se muestra en las siguientes tablas.

Deformación horizontal (Axial al túnel):

D [mm]	S [N]	S_E [N]	C []	F [N]	y_{Fx} [mm]	$y_{FL/2}$ [mm]	P [N]	y_{Px} [mm]	$y_{PL/2}$ [mm]	Q [N]	y_{Qx} [mm]	$y_{QL/2}$ [mm]	y [mm]
25	745,9	7196	1,12	44,15	0,005	0,741	0	0	0	0	0	0	0,821
27	745,9	9791	1,08	44,15	0,004	0,545	0	0	0	0	0	0	0,585
28	745,9	11324	1,07	44,15	0,003	0,471	0	0	0	0	0	0	0,501

Deformación vertical (Perpendicular al túnel):

D [mm]	S [N]	S_E [N]	C []	F [N]	$y_{F x}$ [mm]	$y_{F L/2}$ [mm]	P [N]	$y_{P x}$ [mm]	$y_{P L/2}$ [mm]	Q [N]	$y_{Q x}$ [mm]	$y_{Q L/2}$ [mm]	y [mm]
25	745,9	7196	1,12	127,5	0,016	2,140	0	0	0	88,8	0,028	0,746	3,171
27	745,9	9791	1,08	127,5	0,011	1,573	0	0	0	103,6	0,024	0,639	2,356
28	745,9	11324	1,07	127,5	0,010	1,360	0	0	0	111,4	0,022	0,594	2,058

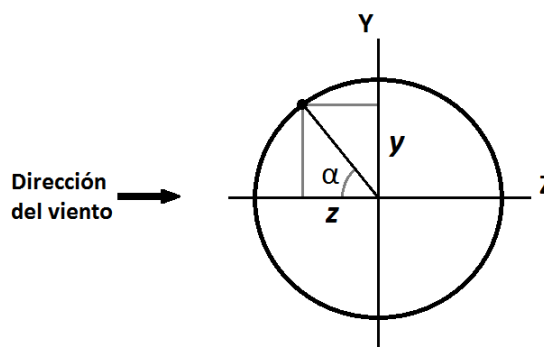
Se selecciona entonces para los ejes horizontales diámetros de 28 mm.

5.3.2 VERIFICACIÓN DE LA RESISTENCIA MECÁNICA

En este caso se tendrán tensiones normales en dos direcciones perpendiculares por lo que primeramente se determinó los momentos aplicados en cada dirección utilizando para ello las expresiones indicadas en el capítulo 5.1. Los valores obtenidos se indican en la siguiente tabla:

	F [N]	M_F [Nm]	P [N]	M_P [Nm]	Q [N]	M_Q [Nm]	S [N]	M_S [Nm]	M_{TOT} [Nm]
Horiz.	44,15	12,97	0	0	0	0	745,9	0,38	13,35
Vert.	127,5	37,45	0	0	111,4	21,82	745,9	1,56	60,84

En este caso las tensiones normales producto de cada momento flector y de la fuerza de compresión S se supondrán. Si tenemos en cuenta la figura siguiente, el momento flector vertical provocará tensiones normales de tracción por debajo del eje Z y de compresión por encima de este. Por su parte el momento flector horizontal generará tracción del lado derecho del eje Y y compresión del lado izquierdo. Por ello, si tenemos en cuenta también la fuerza de compresión S , el cuadrante más solicitado será el superior izquierdo ya que aquí todas las tensiones serán de compresión.



Las tensiones normales σ producto de los momentos flectores en cada dirección dependerán de la distancia del punto analizado respecto del eje neutro de cada caso. Considerando un punto sobre la superficie del eje las tensiones normales σ_{MH} producto del momento horizontal y σ_{MV} del momento vertical actuantes sobre este serán:



$$\sigma_{MH} = \frac{M_{TOT H}}{J} \cdot z = \frac{M_{TOT H}}{J} \cdot \frac{D}{2} \cdot \cos \alpha$$
$$\sigma_{MV} = \frac{M_{TOT V}}{J} \cdot y = \frac{M_{TOT V}}{J} \cdot \frac{D}{2} \cdot \sin \alpha$$

Luego, la tensión de compresión máxima en cada punto del cuadrante superior izquierdo será:

$$|\sigma| = \sigma_{MH} + \sigma_{MV} + \sigma_S$$

$$\sigma = \frac{D}{2 \cdot J} (M_{TOT H} \cdot \cos \alpha + M_{TOT V} \cdot \sin \alpha) + \frac{S}{Area}$$

Modificando el valor del ángulo α se obtienen las tensiones en los distintos puntos de la superficie siendo su valor máximo de $\sigma_{Max} = 30,11 \text{ Mpa}$ cuando $\alpha = 78^\circ$ considerando el eje de 28 mm seleccionado.

Si se utilizan ejes de acero al carbono 1010 trefilado $\sigma_{fl} = 300 \text{ Mpa}$ se tendrá un factor de seguridad efectivo $F_{SE} = 7,54$. Aquí puede verse nuevamente la no linealidad entre las cargas y las tensiones ya que la tensión de fluencia será casi 10 veces mayor que la tensión máxima aplicada.



6. AJUSTE Y TOLERANCIA DE EJES Y BUJES

En el sistema eje-buje el elemento base será el eje, ya que reducir el diámetro de estos sería impracticable. Por lo tanto el sistema será de eje único.

Según las normas ISO para este tipo de aplicaciones se recomienda utilizar ajuste corriente con calidad h9 para el eje y H8 para el agujero. Ahora bien, según el catálogo de Acindar, las barras trefiladas fabricadas por ellos poseen tolerancia ISO estándar h11. Por lo tanto el sistema se realizará con calidad ISO H8/h11.

▪ EJES VERTICALES

EJE h11 y D = 16 mm:

$$\text{Ajuste h (eje único)} \quad DS = 0$$

$$\begin{array}{ll} \text{Calidad 11} & T = 100 \cdot i = 100 (0,45 \sqrt[3]{16\text{mm}} + 0,001 \cdot 16\text{mm}) = 115 \mu = \\ 0,115 \text{ mm} & \end{array}$$

$$D_{\max} = D - DS = 16 - 0 = 16 \text{ mm}$$

$$D_{\min} = D - DS - T = 16 - 0 - 0,115 = 15,885 \text{ mm}$$

AGUJERO H8 y D = 16 mm:

$$\text{Ajuste H} \quad DI = 0$$

$$\text{Calidad 8} \quad T = 25 i = 25 (0,45 \sqrt[3]{16\text{mm}} + 0,001 \cdot 16\text{mm}) = 29 \mu = 0,029 \text{ mm}$$

$$D_{\max} = D + DI + T = 16 + 0 + 0,029 = 16,029 \text{ mm}$$

$$D_{\min} = D + DI = 16 + 0 = 16 \text{ mm}$$

▪ EJES HORIZONTALES

EJE h11 y D = 28 mm:

$$\text{Ajuste h (eje único)} \quad DS = 0$$

$$\begin{array}{ll} \text{Calidad 11} & T = 100 \cdot i = 100 (0,45 \sqrt[3]{28\text{mm}} + 0,001 \cdot 28\text{mm}) = 139 \mu = \\ 0,139 \text{ mm} & \end{array}$$

$$D_{\max} = D - DS = 28 - 0 = 28 \text{ mm}$$

$$D_{\min} = D - DS - T = 28 - 0 - 0,139 = 27,861 \text{ mm}$$

AGUJERO H8 y D = 28 mm:

$$\text{Ajuste H} \quad DI = 0$$

$$\text{Calidad 8} \quad T = 25 i = 25 (0,45 \sqrt[3]{28\text{mm}} + 0,001 \cdot 28\text{mm}) = 35 \mu = 0,035 \text{ mm}$$

$$D_{\max} = D + DI + T = 28 + 0 + 0,035 = 28,035 \text{ mm}$$

$$D_{\min} = D + DI = 28 + 0 = 28 \text{ mm}$$

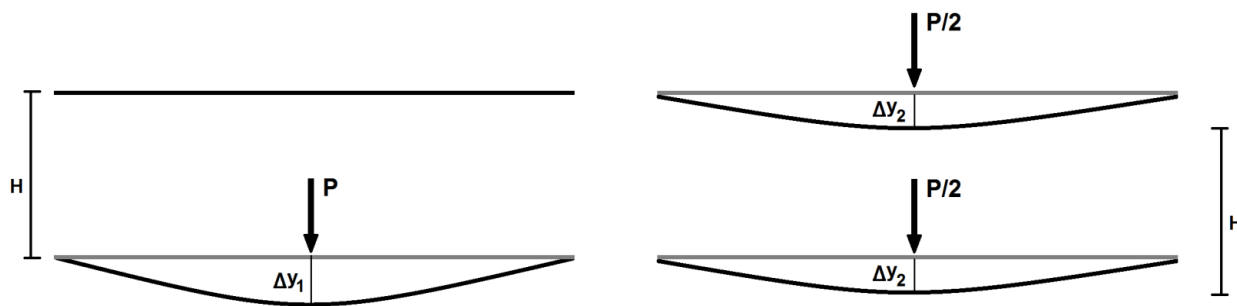
7. DISTRIBUCIÓN DE CARGA DEL EJE VERTICAL SOBRE LOS EJES HORIZONTALES

Para el correcto funcionamiento del posicionador y evitar que los motores carezcan del torque necesario, el peso del eje vertical con todos sus accesorios deberá dividirse lo más uniformemente entre los pares de ejes horizontales. Si el armado no se realiza de la forma correcta toda la carga del eje vertical podría descansar sobre un solo par de ejes horizontales estando el otro descargado. O peor aún, un par de ejes podría sobrecargar al otro, el cual, además de soportar todo el peso del eje vertical, debería soportar también la carga extra transmitida por el otro par.

Se debe recurrir entonces a algún método simple que permita asegurarse de que cada eje horizontal esté soportando la carga que debe. Para ello el armado debe realizarse de la siguiente manera:

- Apoyar por completo el eje vertical sobre el par de ejes horizontales inferior.
- Ajustar los 2 tornillos de sujeción inferior del eje vertical.
- Mover el eje vertical hasta ubicarlo en el centro del eje horizontal (punto de mayor deflexión)
- Colocar los 3 tornillos de sujeción superior del eje vertical y ajustar a mano hasta hacer tope (sin utilizar herramientas).
- Ajustar los tornillos de sujeción superior del eje vertical con alguna herramienta, haciéndolos girar solo la cantidad indicada. De esta manera se descargará el eje horizontal inferior y se empezará a cargar el superior.

Para determinar la cantidad que deben girarse los tornillos de sujeción superior del eje vertical se parte de la siguiente figura:



En la figura izquierda se indica el momento inicial del armado del posicionador. En este momento todo el peso P del eje vertical descansará solo sobre el par de ejes horizontales inferior, y si la carga se posicionara en el centro de los mismos se producirá una deflexión máxima Δy_1 en estos. En este punto si se ajustaran los tornillos superiores solo con la mano hasta hacer tope el eje vertical tendría una longitud $H + \Delta y_1$. Si ahora en lugar de soportar el par de ejes inferiores todo el peso, este se repartiera en partes iguales $P/2$ entre ambos pares de ejes, en cada uno de estos se produciría la misma deflexión Δy_2 y por lo tanto la distancia entre el punto inferior de cada par sería la misma distancia inicial H de los ejes descargados.

Por lo tanto, para que ambos pares de ejes horizontales se repartan por igual la carga del eje vertical, este último deberá tener una longitud H . Para ello, una vez ajustado los tornillos de sujeción superior con la mano, estos deberán girarse de manera de acortar el eje vertical una longitud Δy_1 haciendo así que su longitud final sea H .

Para determinar el valor de la deflexión Δy_1 se procederá como ya se indicó en el capítulo 5.1 pero teniendo en cuenta en este caso solamente la acción de la carga puntual en dirección vertical y sin considerar el peso propio de los ejes ya que estos serán iguales en todos los ejes guía horizontales.

Cabe aclarar en este punto que aquí se debe considerar que el peso del eje vertical se distribuye uniformemente sobre cada par de ejes horizontal ya que ese es el objetivo, a diferencia del cálculo realizado en el capítulo 5.3.1 donde, como medida de seguridad contemplando posibles errores en el ensamblaje, se consideró que 2/3 del peso total descansa sobre cada par de ejes.

Como se indicó en el capítulo 4.9 el peso del eje vertical será de aproximadamente 295 N por lo que, sobre cada par de ejes horizontales, recaerá una carga puntual de 148 N. Considerando luego que la porción de este peso que recaerá sobre cada eje dependerá de la posición de estos respecto del centro de gravedad, teniendo en cuenta la primer figura mostrada en el capítulo 5.3.1 y las dimensiones allí indicadas $d_1 = 141 \text{ mm}$ y $d_2 = 259 \text{ mm}$ se tendrá:

$$F_{Y \text{ 1er eje}} = \frac{d_2}{d_1 + d_2} * 148 \text{ N} = 95,8 \text{ N}$$

$$F_{Y \text{ 2do eje}} = \frac{d_1}{d_1 + d_2} * 148 \text{ N} = 52,2 \text{ N}$$

Así la deformación y sobre cada eje cuando la carga se encuentre en el centro del mismo se muestra en la siguiente tabla:

	S [N]	S_E [N]	C []	F [N]	$y_{F \text{ L/2}}$ [mm]	P [N]	$y_{P \text{ L/2}}$ [mm]	Q [N]	$y_{Q \text{ L/2}}$ [mm]	y [mm]
1er eje	745,9	7196	1,07	95,8	1,022	0	0	0	0	1,699
2do eje	580,1	9791	1,05	52,2	0,557	0	0	0	0	1,185

Luego, utilizando para la sujeción superior del eje vertical tornillos métricos M8 de paso grueso, estos poseen un paso de 1,25 mm, por lo que para eliminar la deflexión Δy_1 en los ejes se deberán girar los tornillos:

- Tornillo único próximo al 1er eje

$$Giro = \frac{\Delta y_1}{Paso_{M8}} = \frac{1,699 \text{ mm}}{1,25 \text{ mm}} \approx 1 \text{ y } 1/3 \text{ vueltas}$$

- Par de tornillos próximos al 2do eje

$$Giro = \frac{\Delta y_1}{Paso_{M8}} = \frac{1,185 \text{ mm}}{1,25 \text{ mm}} \approx 15/16 \text{ vueltas}$$



8. ACCESORIOS

8.1 CADENAS PORTA CABLES

Las mismas se seleccionaron en función de los elementos que deberán contener (cables y/o mangueras) y el radio de curvatura máximo de las mismas, lo que será limitado por el espacio dentro del cual estas deberán moverse. Luego de seleccionadas se debe verificar el límite de auto-sustentación.

Para su selección se utilizará la línea liviana de la marca FESMA, cuyo catálogo se adjunta en el anexo A.8.

▪ CADENA PARA EJE VERTICAL

Por la disposición de los distintos elementos que compondrán al eje vertical, el espacio dentro del cual la cadena se moverá será de 90 mm, por lo que la altura total de la cadena doblada deberá ser menor que este valor. Esta altura total se indica en el catalogo como H, y varia dentro de una misma serie según el radio de curvatura R de la cadena. Los modelos que cumplen con este requisito son los de la Serie 10 en su totalidad, los de la Serie 12 para radios de curvatura 18 y 28 y la Serie 18 para radio de curvatura 28.

Luego, la cadena deberá contener 2 mangueras de aire provenientes de los instrumentos de medición. Si estas se suponen de diámetro exterior 6 mm y se deja además un espacio de reserva para elementos extra como ser más mangueras, mangueras de mayor diámetro o cables, se requeriría de una cadena de al menos 20 mm de ancho útil. Ahora bien, todas las cadenas de las series 10 y 12 poseen un ancho único de 10 y 15 mm respectivamente. Por su parte las de Serie 18 se proveen en anchos de 15, 18, 26 y 37 mm, pudiéndose entonces adoptar únicamente una de estas dos últimas. Se selecciona en este caso la de 37 mm de ancho útil cuyo código es FB.18.37.28.

En este caso se instalará la cadena en forma vertical con el punto fijo en la parte central del desplazamiento, por lo que la longitud total de la cadena vendrá de la formula $L = C/2 + E$, donde C será la longitud del desplazamiento del porta instrumento y E la longitud de la curva de la cadena, la cual viene indicado en el catalogo para el radio de curvatura mínimo de la misma. En este caso C será aproximadamente 1435 mm y para la cadena Serie 18 con radio de curvatura 28 el valor de E es de 152 mm. Por lo tanto la cadena deberá poseer una longitud mínima de 870 mm. Teniendo en cuenta que el valor real de E será un poco mayor al utilizado en el cálculo y previendo otros inconvenientes de instalación se considera una cadena de 1 m de longitud. La misma según catalogo estará compuesta por 32 eslabones y 2 terminales de fijación, pesando 384 gr en total.

Como la cadena será instalada en posición vertical con curvatura hacia abajo no se requiere verificación del límite de auto-sustentación de la misma.

Se selecciona entonces como cadena porta cables vertical:

- Marca: Fesma Serie 18
- Código: FB.18.37.28 (FB = Con abertura superior)
- Radio de curvatura: 28 mm
- Sección útil: 37 x 18 mm
- Sección total: 51 x 23 mm
- Longitud: 1 m
- N° de eslabones: 32 + 2
- Peso total: 384 gr



▪ CADENA PARA EJE HORIZONTAL

En este caso el espacio dentro del cual la cadena se moverá será de 100 mm.

Mediante el mismo criterio utilizado para la selección de la cadena vertical se determinó que podrían utilizarse únicamente las cadenas Serie 18 de radio de curvatura 28 y 38 y ancho útil de 37 mm. En este caso la cadena contendrá cables de señal y de muy baja tensión, por lo que se prefirió utilizar la cadena de mayor radio de curvatura para reducir el daño a los cables.

Al igual que la cadena vertical esta también tendrá su punto fijo en la parte central del desplazamiento, por lo que la longitud de la misma se determina mediante la formula $L = C/2 + E$. Aquí la longitud del desplazamiento del eje vertical será de 2110 mm mientras que el valor de E en este caso será de 183 mm, por lo que se requerirá una cadena de 1,30 m, la cual según catalogo estará compuesta por 42 eslabones y 2 terminales de fijación, pesando 500 gr en total.

En este caso por la disposición horizontal de la cadena debe verificarse el límite de autosustentación de la misma. Para este caso $C/2 = 1,055$ m por lo que según el grafico indicado en el catálogo de la cadena, esta puede soportar hasta 0,5 Kg/m, lo cual es superior a la carga estimada y por lo tanto se verifica la autosustentación.

Se selecciona entonces como cadena portacables Horizontal:

- Marca: Fesma Serie 18
- Codigo: FB.18.37.38 (FB = Con abertura superior)
- Radio de curvatura: 38 mm
- Sección útil: 37 x 18 mm
- Sección total: 51 x 23 mm
- Longitud: 1,3 m
- N° de eslabones: 42 + 2
- Peso total: 500 gr

8.2 ACOPLAMIENTO ENTRE EJES Y REDUCTORES HORIZONTALES

Estos accesorios se utilizarán para acoplar los ejes de salida de los reductores planetarios de los motores horizontales con los ejes que soportarán a las poleas motoras horizontales. Los mismos no serán necesarios en el caso del eje vertical ya que en ese caso la polea motora se montará directamente sobre el eje de salida del reductor planetario.

Se utilizarán acoplamientos flexibles para así absorber posibles desalineaciones entre los ejes de las poleas y los de los reductores. En este caso se selecciona acoplamientos en estrella sin juego modelo GAS/SG-ST de marca Comintec, los cuales según catalogo son ideales para maquinaria CNC y mecánica de precisión. Las características de los mismos se adjuntan en el anexo A.17.

En este caso ya que el diámetro del eje del reductor (9,51 mm) y el diámetro en el extremo del eje de la polea (8 mm) serán diferentes, se selecciona el tipo de bloqueo B1: Con mordaza de una parte en orificio H7 con hueco, con el cual se pueden utilizar ejes de distinto diámetros siempre que estos se encuentren dentro de un determinado rango de valores.

El par nominal del acoplamiento deberá ser superior al par motor en el eje multiplicado por un factor de servicio F, por lo que el mismo deberá poseer un valor superior a:



$$C_{nom} > T_{Eje} \cdot F = T_{Mot} \cdot i \cdot F$$

Dónde:

C_{nom} : Par nominal del acoplamiento en [Nm].

T_{Eje} : Par motor en el eje de salida del reductor de velocidad [Nm]

T_{Mot} : Par motor entregado por el motor [Nm]

i : Relación de transmisión del reductor de velocidad [RPM]

Luego, según lo calculado en el capítulo 4.9, el par necesario en cada motor horizontal será de 1,88 Nm, si la relación del reductor de velocidad es $i = 10$ y considerando un factor de servicio $F = 2$, se requerirá un acoplamiento con par nominal superior a:

$$C_{nom} > 1,88 \text{ Nm} \cdot 10 \cdot 2 = 37,6 \text{ Nm}$$

En este caso, utilizando ejes de 8 mm y 9,51 mm, el tamaño de mordaza B que permitirá la transmisión de más de 37,6 Nm será la de tamaño 00(19), las cuales pueden transmitir 45 Nm y 47 Nm para ejes de 8 mm y 10 mm respectivamente.

Como elemento elástico deberá utilizarse poliuretano de dureza 64 Sh-D, el cual para el tamaño 00(19) soporta un par máximo de 42 Nm.

Por tanto los dos acoplamientos a utilizarse tendrán el código: GAS/SG ST 00 M1-M1 d1=8 H7 B1 d2=10 H7 B1 – Estrella Verde SG 64 Sh-D

8.3 MANGUERAS DE PVC

Se usa como guía para la selección de las mangueras el catálogo de Espiroflex adjuntado en el anexo A.9. De allí se obtiene para las mangueras de PVC transparente de diámetro exterior 6 mm e interior de 3 mm un peso de 26 gr por metro y un radio de curvatura de 12 mm, menor al radio de curvatura de la cadena vertical.

Luego, siendo la cadena de 1 m de longitud y dejando 1 m extra para las conexiones de las mismas al instrumento de medición y a los sensores, se obtienen 4 metros de manguera con un peso de 104 gr.

8.4 INSTRUMENTO DE MEDICIÓN

El tubo de Pitot con el que cuenta el laboratorio de aerodinámica posee 1 cm de espesor y una longitud total aproximada de 65 cm, estimándose para el mismo un peso de 350 gr y un total de 500 gr incluyendo el soporte para el mismo.

Si bien el posicionador está pensado para ser utilizados con tubos de Pitot Prandtl y anemómetros, en el mismo pueden colocarse cualquier otro tipo de instrumentos. Por ello para los cálculos se considera que el conjunto instrumento más porta instrumento posee un peso de 600 gr.

8.5 CARRO Y RIEL PORTA CARRO HORIZONTAL

Los carros soportarán a los ejes horizontales, para ello se utilizarán dos carros en cada extremo de estos. Para determinar el modelo de los carros se debe calcular el peso que deberá soportar cada uno de ellos.

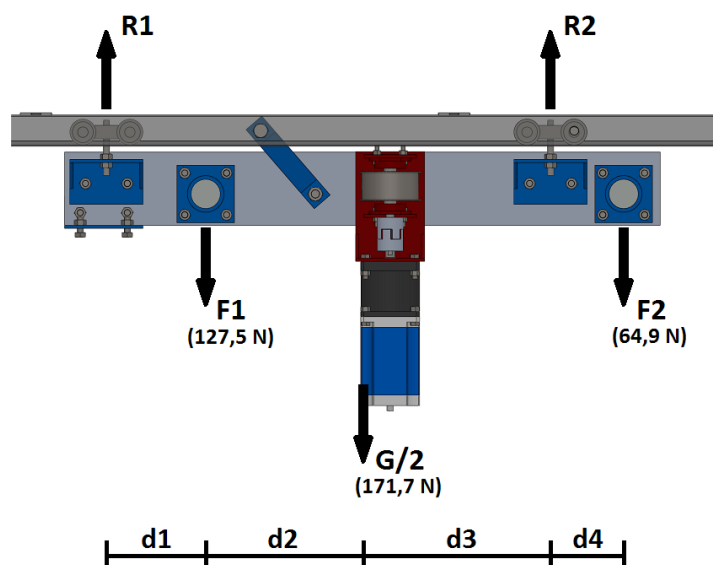
La carga máxima sobre los carros se producirá cuando el eje vertical se ubique en un extremo de los ejes horizontales, en ese momento podría considerarse que todo el peso del eje vertical recaerá solamente sobre ese extremo, repartiéndose el peso entre el eje horizontal superior e inferior. Como se indicó en el capítulo 4.9 el peso de todos los elementos móviles (eje vertical más los soportes centrales de cada eje horizontal) será de aproximadamente 295 N. Luego, por seguridad, a la hora de calcular se consideró que cada eje horizontal soportará $2/3$ de este peso (197 N). Como se determinó en el capítulo 5.3.1, esta fracción del peso se distribuirá entre los ejes horizontales guía de cada eje, recayendo 127,5 N en el primer eje y 69,4 N en el segundo eje.

Por otra parte debe considerarse el peso propio de cada eje horizontal (sin considerar los soportes centrales ya considerados en los elementos móviles). Este peso se distribuirá de forma pareja entre ambos extremos de cada eje debido a la simetría de los mismos. El lugar de aplicación de estos pesos serán los centros de masa de cada eje, los cuales se determinan a partir de la modelación en SolidWorks.

Los cálculos se realizarán del lado de la seguridad ya que el eje vertical repartirá siempre su peso entre ambos extremos de cada eje y no por completo sobre un solo extremo. Además, como se indica en el capítulo 7, durante el ensamblaje se buscará que el peso del eje vertical se reparta en forma pareja entre ambos ejes horizontales en lugar de los $2/3$ utilizados en el cálculo.

▪ EJE HORIZONTAL SUPERIOR

En este caso el peso del eje horizontal superior será de aproximadamente 35 Kg, por lo que cada extremo soportará 17,5 Kg (171,7 N). Luego, los pesos R1 y R2 que deberá soportar cada carro dependerán de las cargas y las ubicaciones de estas, como se muestra en la siguiente figura:



Siendo $d1 = 9,50 \text{ cm}$; $d2 = 16,21 \text{ cm}$; $d3 = 16,79 \text{ cm}$; $d4 = 7 \text{ cm}$.

Luego las cargas aplicadas a cada carro serán:

$$R1 = 156,1 \text{ N} = 15,9 \text{ Kg}$$

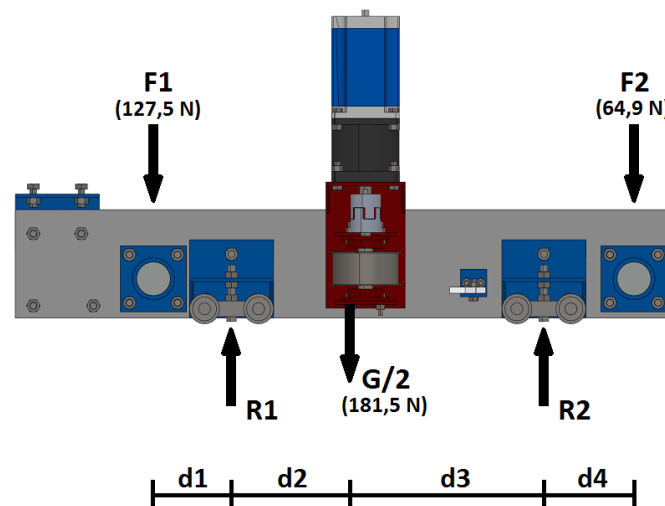
$$R2 = 208 \text{ N} = 21,2 \text{ Kg}$$

Según catálogo de Herrajes Roma adjunto en anexo A.10 podrían utilizarse los carros 168-A que admiten 60 Kg por par (30 Kg c/u) o los carros 164-A que admiten 80 Kg por par (40 Kg c/u).

Se selecciona para su uso los carros 164-A con rieles planos 164, trabajando así con una seguridad de 1,88 según el segundo carro.

▪ EJE HORIZONTAL INFERIOR

En este caso el peso del eje horizontal inferior será de aproximadamente 37 Kg, por lo que cada extremo soportará 18,5 Kg (181,5 N). Luego, los pesos R1 y R2 que deberá soportar cada carro dependerán de las cargas y las ubicaciones de estas, como se muestra en la siguiente figura:



Siendo $d1 = 6,50 \text{ cm}$; $d2 = 9,71 \text{ cm}$; $d3 = 16,29 \text{ cm}$; $d4 = 7,5 \text{ cm}$.

Luego las cargas aplicadas a cada carro serán:

$$R1 = 254,4 \text{ N} = 25,9 \text{ Kg}$$

$$R2 = 119,5 \text{ N} = 12,2 \text{ Kg}$$

Según catálogo de Herrajes Roma podrían utilizarse los carros 168-A que admiten 60 Kg por par (30 Kg c/u) o los carros 164-A que admiten 80 Kg por par (40 Kg c/u).

Se selecciona para su uso los carros 164-A con rieles planos 164, trabajando así con una seguridad de 1,56 según el primer carro.



2DA PARTE

SISTEMA DE CONTROL



9. DESCRIPCIÓN DEL SISTEMA DE CONTROL

El cerebro principal del sistema de control estará formado por una placa de Arduino modelo Mega 2560 R3. La misma contendrá la programación principal del sistema, es decir, la programación de las diferentes funciones. Esta a su vez se comunicará con las distintas interfaces HMI, comandará a los drivers de los motores y leerá las señales provenientes de los sensores y pulsadores.

Poseerá 3 interfaces Hombre-Máquina (HMI). La interfaz principal será una aplicación creada mediante LabVIEW para su instalación en PC donde la transferencia de datos entre esta y el Arduino se realizará mediante conexión USB. Con esta interfaz principal se tendrá el manejo completo del posicionador con todas sus funciones y se visualizará en tiempo real la posición del mismo.

Por otra parte se tendrá una interfaz para celular mediante una APP la cual se conectará al Arduino a través de Bluetooth y por último una interfaz tipo botonera conectada a los pines digitales del Arduino. Estas dos últimas interfaces permitirán un manejo reducido del posicionador ya que algunas funciones no podrán ejecutarse mediante estas ya que requerirán el ingreso de datos numéricos por parte del usuario, lo cual únicamente podrá realizarse mediante la aplicación de LabVIEW. La descripción detallada y programación de las diferentes HMI se indican en la tercera parte de este informe.

Por su parte el Arduino mediante salidas digitales indicará a los drivers de los motores el instante y sentido de giro en que cada motor debe realizar un paso. En este caso se trabajará como sistema abierto sin realimentación debido a la muy buena repetitividad de los motores paso a paso.

Se tendrá también conectado a los pines de entrada digitales del Arduino dos sensores de posición ubicados en el posicionador, los cuales servirán de referencia para determinar la posición exacta del posicionador cuando esto sea necesario.

El posicionador poseerá dos sistemas de protección para evitar que el mismo colisione con la estructura durante su desplazamiento. Uno de ellos será mediante la programación, la cual cuando determine que el posicionador se encuentre próximo a la estructura detendrá el avance del mismo. La otra se realizará mediante fines de carrera que actuarán sobre un circuito eléctrico independiente cortando la alimentación eléctrica a los drivers de los motores.

10. ARDUINO

Arduino es una plataforma electrónica o microcontrolador que permite realizar de forma sencilla todo tipo de proyectos interactivos. El mismo contiene circuitos electrónicos con entradas y salidas analógicas y digitales que conectan el mundo físico con el virtual. Gracias a su facilidad y versatilidad estas placas pueden utilizarse para desarrollar de manera sencilla prototipos de automatización, control, muestreo de datos, etc. y todo tipo de proyectos interactivos autónomos o comunicados a una PC u otro dispositivo.

El mismo incluye tanto el software como el hardware, y la documentación de estos se encuentra en formato open-source o de código abierto, por lo que cualquiera puede usarlos y hacer cambios en ellos.

▪ HARDWARE ARDUINO

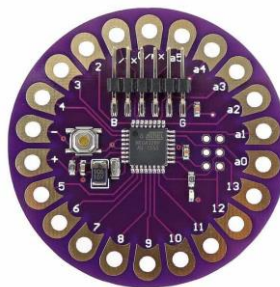
El hardware de arduino es básicamente un microprocesador y los bloques funcionales necesarios para su funcionamiento.

Las placas se pueden ensamblar a mano o adquirirlas pre ensambladas. Los diseños de referencias del hardware están disponibles bajo licencia open-source, por lo que cada usuario puede adaptarlas a sus necesidades personales.

Existen gran variedad de placas, diferenciándose principalmente por el microprocesador, número de entradas/salidas, memoria, velocidad de reloj y tensión de alimentación.



Arduino Uno R3



Arduino Lilypad



Arduino Nano

▪ ARDUINO MEGA 2560 R3

De las diversas placas existentes se seleccionó trabajar con el modelo MEGA 2560 R3 debido a su gran número de entradas/salidas digitales. Las características del hardware de esta placa son:

- | | |
|------------------------------------|---------------|
| - Microcontrolador: | ATmega2560 |
| - Tensión de operación: | 5 V |
| - Tensión de entrada (recomendada) | 7 – 12 V |
| - Tensión de entrada (limite) | 6 – 20 V |
| - Pines I/O digitales | 54 (15 pueden |
| utilizarse como salidas PWM) | |



- Pines entrada analógica	16
- Corriente en pines I/O digitales	20 mA
- Corriente en pin 3,3V	50 mA
- Memoria flash	256 KB (8 KB utilizados por el bootloader)
- SRAM	8 KB
- EEPROM	4 KB
- Velocidad del reloj	16 MHz

▪ SOFTWARE ARDUINO

El software de Arduino es un entorno de desarrollo integrado (IDE) de código abierto que implementa el lenguaje de programación Arduino y el bootloader ejecutado en la placa. El mismo es un entorno de programación que ha sido empaquetado como un programa de aplicación. Este consta de un editor de código, un compilador, un depurador, un constructor de interfaz gráfica y las herramientas necesarias para cargar el programa compilado en la memoria flash del hardware.

La principal característica del software y del lenguaje de programación es su sencillez y facilidad de uso.

▪ COMUNIDAD ARDUINO

Otro factor importante de Arduino es la comunidad que apoya este desarrollo, comparte conocimiento, elabora librerías para facilitar el uso de Arduino y publica proyectos para que puedan ser replicados o tomados como base para otros proyectos similares. Por ello, en internet se encuentran fácilmente cursos, tutoriales, foros de consulta, proyectos, etc. que ayudan al usuario a aprender fácilmente su uso y solucionar problemas.

▪ MÓDULOS Y DRIVERS

A su vez, existen numerosos módulos, drivers, sensores, etc. para trabajar fácilmente con las placas de Arduino, lo que permite realizar una gran variedad de proyectos de manera sencilla. Entre ellos se encuentran por ej módulos Bluetooth, módulos Wi-Fi, drivers para control de motores paso a paso¹ y de CC, sensores de distancia por ultrasonido o infrarrojos, relés optoacopladores, pantallas LCD, sensores de movimiento, sonido, humo, módulos lectores de tarjetas MicroSD, entre muchos otros.



Driver Pololu A4988 para control de motores Paso a Paso o de CC



Módulo Wi-Fi ESP8266



Sensor de distancia por ultrasonido HC-SR04

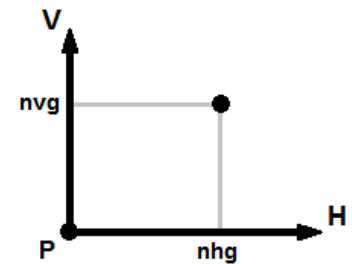
¹ El Driver para control de motores paso a paso Pololu A4988 mostrado en la imagen es una versión reducida (corriente Max 2A) de los drivers seleccionados en el capítulo 22, y su control mediante las salidas digitales del Arduino es idéntica.

11. SISTEMAS DE REFERENCIA.

11.1 VARIABLES nvg y nhg

Como se indicó anteriormente el accionamiento del posicionador se hará mediante motores paso a paso. Estos motores realizan avances discretos y poseen muy buena repetitividad, esto permite trabajar con ellos como sistemas abiertos sin realimentación. Gracias a esto conocer la ubicación del posicionador, es decir, sus coordenadas, se vuelve mucho más simple. Para ello únicamente se debe ir registrando cada paso realizado por los motores. Este registro se efectúa a través de las variables nvg y nhg en las cuales se almacena el número de pasos realizados en dirección vertical y horizontal respectivamente. Estas variables no son más que contadores que por cada paso realizado se incrementan o decrementan en una unidad dependiendo del sentido del desplazamiento.

Si ambas variables nvg y nhg poseen valor 0 en un punto determinado P, y van actualizando sus valores por cada paso realizado por los motores en el desplazamiento del posicionador, los valores de estas variables estarán indicando la cantidad de pasos verticales y horizontales a los cuales se encuentra el posicionador de dicho punto P. Es decir, estas variables en la forma (nhg ; nvg) indicarán las coordenadas “en número de pasos” del posicionador respecto de un sistema de referencias con origen en el punto P. Luego, mediante un factor de conversión, podrán obtenerse también estas coordenadas en unidades de longitud.

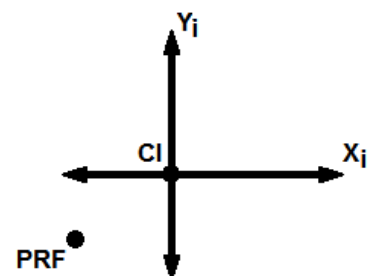


Todo el funcionamiento del sistema se efectuará monitoreando los valores de estas variables, ya sea para cálculos internos de la programación de cada función, memorizar una posición, detectar si el posicionador sale fuera de la zona de trabajo o para indicar al usuario las coordenadas en tiempo real a través de la interfaz.

Las variables nvg y nhg serán almacenadas en la memoria volátil SRAM del Arduino. Por lo tanto sus valores se perderán al apagar o resetear el mismo, no pudiendo así guardarse la última posición de la máquina. Debido a esto cada vez que se encienda o resetee el Arduino estas variables se iniciarán con valor 0, y a partir de allí se irán modificando a medida que el posicionador se desplaza.

11.2 SISTEMA DE REFERENCIA INICIAL

Como se indicó, al encender o resetear el Arduino, los contadores nvg y nhg se iniciarán con el valor 0. Esto puede interpretarse como si inicialmente se trabajara con un sistema de referencia cuyo origen sea el punto donde se encuentre el posicionador al momento de encender o resetear el Arduino, ya que este punto tendrá coordenadas (nhg ; nvg) = (0 ; 0). A este sistema de referencia se lo denominará Sistema de Referencia Inicial (SRI) y a su origen CERO INICIAL (CI). El origen de este sistema por lo tanto no será fijo sino variable, ya que podrá cambiar de lugar en cada reinicio del Arduino. En este caso las variables nvg y nhg , y por consiguiente las coordenadas del posicionador a medida que este se desplaza, podrán tomar valores positivos o negativos, ya que dependerá de en qué lugar se encuentre el origen del sistema de referencias.



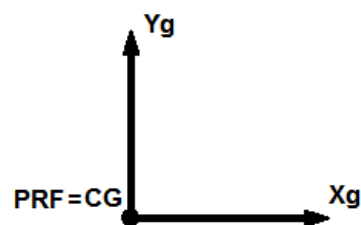
Este sistema no se encontrará relacionado con la estructura. Es decir, a través las coordenadas podrá conocerse la ubicación del posicionador respecto del CI, pero no podrá conocerse su ubicación real respecto de la estructura. Por lo tanto, no podrá saberse si el posicionador se encuentra cerca o lejos de los extremos de la máquina y si pueda colisionar o no con la estructura al desplazarse.

Por otro lado, al no poseer un origen fijo, las coordenadas de cada punto tampoco serán fijas, no pudiéndose volver a un punto en particular luego de apagar o resetear el Arduino ya que el origen CI pudo haber cambiado de posición y por ello también las coordenadas de dicho punto particular respecto del SRI.

11.3 SISTEMA DE REFERENCIA GLOBAL

Para conocer la ubicación real del posicionador respecto de la estructura será necesario un sistema de referencias relacionado a esta. A este sistema se lo denominará Sistema de Referencias Global (SRG), y el origen del mismo recibirá el nombre de CERO GLOBAL (CG).

La relación entre este sistema de referencia y la estructura se conseguirá a través de un punto de referencia físico (PRF). Este PRF será un punto fijo determinado mediante dos sensores de posición. Así, cuando el posicionador se encuentre bloqueando de forma simultánea ambos sensores, en ese momento se conocerá la ubicación exacta de este respecto de la estructura. Luego, esta posición perfectamente determinada será el punto CG, origen del Sistema de Referencias Global.



Pudiendo entonces localizar perfectamente la ubicación del CG respecto de la estructura, solo restaría convertir a dicho punto en origen de este sistema de referencias. Esto se logrará llevando al posicionador hasta este punto y allí restableciendo a 0 el valor de los contadores de pasos. De esta manera ahora las coordenadas del punto CG serán $(nhg ; nvg) = (0 ; 0)$. A partir de esto podrá conocerse la ubicación exacta del posicionador respecto de la estructura simplemente monitoreando los valores de las variables nvg y nhg ya que las mismas estarán referidas a este origen fijo. Gracias a esto podrá realizarse cálculos en la programación para impedir que el posicionador salga fuera de la zona de trabajo y colisione con la estructura durante su desplazamiento. A demás, si se conocen las coordenadas de un punto de interés puede regresarse al mismo punto luego de un reinicio del arduino, ya que la ubicación de este origen del sistema de referencias y por ello las coordenadas del punto de interés respecto de ella serán invariables.

La detección del PRF se realizará mediante dos sensores, uno para el desplazamiento vertical y el otro para el horizontal. Estos se denominarán Sensores de Posición y al estar bloqueados simultáneamente estarían indicando que el posicionador se encuentra en este punto de referencia. Los sensores se encontrarán en el extremo inferior izquierdo del posicionador y luego el CG se ubicará en este punto.

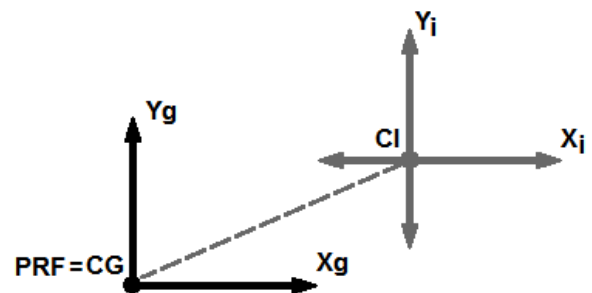
Ya determinada la posición del CERO GLOBAL este se convertirá ahora en el límite inferior izquierdo al cual pueda desplazarse el posicionador ya que la programación impedirá que el mismo se desplace más allá de este punto. En consecuencia, por ser este punto el origen de este sistema de referencia, nvg y nhg siempre tendrán valores positivos y por ello las coordenadas del posicionador también serán siempre positivas.

11.4 CAMBIO DEL SISTEMA DE REFERENCIAS INICIAL AL GLOBAL

Al momento de encender o resetear el Arduino se iniciará con el SRI. Como se explicó, el origen de este sistema será el punto donde se encuentre el posicionador al momento de ser encendido y por lo

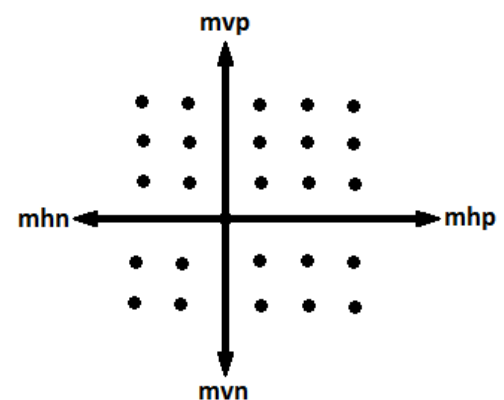
tanto el mismo será aleatorio. Las coordenadas estarán referidas a este punto y podrán tomar valores tanto positivos como negativos, dependiendo de dónde se ubique el origen. Este sistema no estará relacionado con la estructura y por ello no se podrá conocer la ubicación exacta del posicionador respecto de esta. Por lo tanto, no podrá funcionar la protección contra colisiones mediante la programación que se explicará más adelante. Debido a esto, por seguridad algunas funciones permanecerán desactivadas mientras se trabaje con este sistema, y otras podrán utilizarse pero con precaución por parte del usuario.

Para poder trabajar con el SRG será necesario trasladar el origen del sistema de referencias del punto CI al punto CG. Esto se conseguirá llevando al posicionador hasta el PRF para allí restablecer a 0 el valor de los contadores nvg y nhg . Todo este procedimiento se realizará automáticamente mediante una función creada para ello llamada DETERMINAR CERO¹, la cual se ejecutará en el momento que el usuario lo desee. Utilizando este sistema podrá conocerse la ubicación exacta del posicionador respecto de la estructura, ya que el origen del sistema se encontrará relacionado a esta. Gracias a ello podrá entonces activarse la Protección Contra Colisiones Mediante la Programación y con ello las funciones inicialmente desactivadas.



11.5 SISTEMA DE REFERENCIA RELATIVO

Este sistema solamente será utilizado en la función MEDICIÓN AUTOMÁTICA. El mismo tomará como origen al punto donde se encuentre el posicionador al momento de iniciar esta función. Los ejes de este sistema, a diferencia del Inicial y el Global, no medirán pasos si no numero de mediciones. Así, mediante este sistema, el usuario indicará la cantidad de mediciones que desee realizar en cada dirección y sentido, y podrá además visualizar el avance del proceso de medición observando las coordenadas del punto actual de medición respecto de este sistema. Al finalizar la función se regresará al origen del sistema, de manera que el mismo finalice en el mismo punto donde se inició.



¹ La descripción de las funciones se realizan más adelante en el capítulo 14.

12. PROTECCIÓN CONTRA COLISIONES

Para evitar que el posicionador colisione con la estructura en caso de desplazarse en exceso se contará con dos tipos de protecciones:

- Protección mediante la programación
- Protección mediante fines de carrera de protección

12.1 PROTECCIÓN MEDIANTE LA PROGRAMACIÓN

Esta protección requerirá conocer la ubicación del posicionador respecto de la estructura por lo tanto solo se encontrará activa cuando se esté trabajando con el SRG.

La misma consistirá en limitar el movimiento del posicionador a una determinada zona de trabajo. Ya que la posición estará determinada por las variables nvg y nhg , puede limitarse la zona de trabajo limitando el rango de variación de estas. Esto se consigue monitoreando el valor de estas variables y evitando así que tomen valores fuera de los límites máximos y mínimos previamente fijados en la programación. Para esto, antes de cada avance se verificará que estas variables no vayan a superar estos valores límite. El límite inferior será el valor 0 para ambas variables. El límite superior lo determinarán las variables de ajuste $nvgMax$ y $nhgMax$ cuyos valores numéricos deberán ser obtenidos del mismo posicionador una vez construido. La descripción de estas variables y el procedimiento para determinar sus valores se indican en el capítulo 13.2.

La zona de trabajo estará definida entonces por los intervalos:

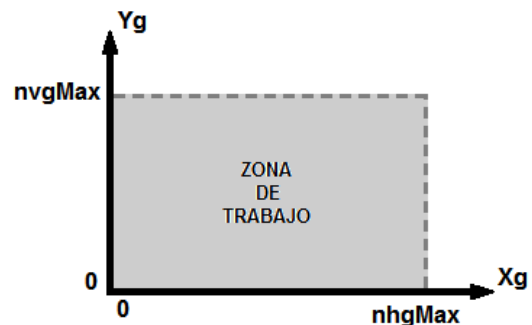
$$0 \leq nhg \leq nhgMax$$

$$0 \leq nvg \leq nvgMax$$

Cuando se esté ejecutando la función “MOVIMIENTO LIBRE” y se ordene avanzar en una dirección, antes y durante el desplazamiento se comprobará el valor de la variable correspondiente a esa dirección. Si se detecta que esta variable alcanzará alguno de los límites, se bloqueará el avance en ese sentido, pero se permitirá el avance en sentido contrario. Es decir, si la variable alcanzara el valor 0, se bloqueará el retroceso en esa dirección y solo se permitirá el avance. De igual manera si la variable alcanzara el valor máximo se bloqueará solo el movimiento de avance. En estos casos tanto en la aplicación de LabVIEW como en la botonera se encenderán señales indicando que se llegó a uno de los extremos.

En la función “IR A COORDENADA” luego de ingresar la coordenada deseada, se comparará el valor de esta con los valores límites de la zona de trabajo. En caso de superarse alguno de estos límites en la aplicación de LabVIEW (esta función solo se podrá ejecutar desde esta interfaz) se desplegará una ventana informando que la coordenada ingresada no es válida y se indicará el rango de valores permitidos para cada una de ellas.

En el caso de la función “MEDICIÓN AUTOMÁTICA” luego de ingresar los valores deseados para la medición (número de mediciones y distancia entre ellos) la programación calculará el rango de valores que deberán tomar nvg y nhg para lograr los desplazamientos necesarios para estas mediciones. Si este rango se encontrara dentro de los límites se podrá iniciar el proceso de medición mediante el botón de





INICIO de la aplicación de LabVIEW. De lo contrario la aplicación indicará mediante señales que existe un desborde en las mediciones y señalará en qué dirección y sentido se produce. A su vez se bloqueará el botón de INICIO del proceso de medición, pudiéndose únicamente reingresar nuevos valores deseados para la medición o cancelar el proceso de medición automática.

Como ya se mencionó esta protección solo funcionará con el SRG ya que requiere conocer la posición exacta del posicionador, lo que se consigue solo con este sistema. Por lo tanto, por seguridad, mientras se esté trabajando con el Sistema de Referencia Inicial las funciones IR A COORDENADA y MEDICIÓN AUTOMÁTICA permanecerán bloqueadas ya que requieren conocer los límites dentro de los cuales moverse. Mientras que la función MOVIMIENTO LIBRE podrá ejecutarse pero sin esta protección, por lo tanto debe tenerse cuidado durante los movimientos en esta función, en especial cuando se trabaje cerca de los extremos de la estructura.

12.2 PROTECCIÓN MEDIANTE FINES DE CARRERA DE PROTECCIÓN

Como se explicó, la protección mediante la programación no estará activa mientras se utilice el SRI. Por esto debe contarse con una protección extra que prevenga colisiones y posibles daños mecánicos mientras se utilice este sistema de referencia.

A demás, esta protección servirá también como segundo mecanismo para evitar colisiones mientras se trabaje en el SRG. Este segundo mecanismo es necesario en caso de no funcionar la Protección Mediante La Programación, por ejemplo, por estar mal determinados los valores $nvgMax$ y $nhgMax$.

Esta protección consiste en colocar fines de carrera en el posicionador los cuales, mediante un circuito eléctrico independiente, cortarán la alimentación eléctrica a los driver de los motores. Estos fines de carrera, normalmente cerrados, se conectarán en serie para alimentar el circuito de mando de los contactores encargados de realizar el cierre o apertura de la alimentación de los drivers.

El circuito lógico así formado se describe en el Capítulo 24.



13. VARIABLES DE AJUSTE

13.1 DESCRIPCIÓN

Las variables de ajuste son variables que dependen de las dimensiones definitivas y componentes del posicionador y por ello solo podrán determinarse correctamente cuando el mismo se encuentre ya montado dentro del túnel. Estas variables pueden variar entre un montaje y otro, en especial si se realizó alguna modificación mecánica como ser por ej cambio de correas o retensado de las mismas, cambio de poleas, motores, o cualquier tipo de modificación estructural. Incluso pueden depender del lugar dentro del túnel en el cual el posicionador sea montado, ya que las dimensiones del túnel a lo largo del mismo presentan pequeñas variaciones.

Por lo tanto, luego de cada nuevo montaje del posicionador el usuario deberá determinar el valor de estas variables e ingresarlos a la programación a través de la interfaz de LabVIEW.

Estas variables son:

- NumPasosXMetroH: Cantidad de pasos de deben realizar los motores encargados del desplazamiento horizontal para que dicho desplazamiento sea de 1 metro [N° de pasos].
- NumPasosXMetroV: Cantidad de pasos de debe realizar el motor encargado del desplazamiento vertical para que dicho desplazamiento sea de 1 metro [N° de pasos].
- nhgMax: Cantidad máxima de pasos que el posicionador puede desplazarse en dirección horizontal a partir del punto CERO GLOBAL [N° de pasos].
- nvgMax: Cantidad máxima de pasos que el posicionador puede desplazarse en dirección vertical a partir del punto CERO GLOBAL [N° de pasos].

NumPasosXMetroH y NumPasosXMetroV se utilizarán como factores de conversión para relacionar número de pasos con desplazamientos en mm del posicionador.

Como se explicó en el capítulo 12.1, nhgMax y nvgMax determinarán la cantidad máxima de pasos que puede realizar el posicionador en cada dirección a partir de CG evitando así que el mismo colisione con la estructura durante su desplazamiento, es decir, delimitarán la zona de trabajo del posicionador.

13.2 DETERMINACIÓN DE LAS VARIABLES DE AJUSTE

IMPORTANTE: Para comprender este capítulo es necesario primeramente leer los capítulos 17.4. Este se ubica aquí solo por cuestión de orden.

Para determinar el valor de estas variables de ajuste se necesitará utilizar la aplicación de Labview. En ella, en el margen izquierdo en la sección VARIABLES DE AJUSTE se encontrarán los CONTROLES/INDICADORES:

- N° Max PH (corresponde a nhgMax),
- N° Max PV (corresponde a nvgMax),
- Pasos / mH (corresponde a NumPasosXMetroH),
- Pasos / mV (corresponde a NumPasosXMetroV),

En primer lugar al ejecutar la aplicación se deberá seleccionar el botón MODIFICAR VARIABLES con lo cual se habilitan los controles para el ingreso de datos por parte del operador. Se deberá asignar luego a estas variables los siguientes valores que permitirán operar el posicionador (el valor aquí asignado a estas variables se explicará más adelante):

- N° Max PH: 999999
- N° Max PV: 999999
- Pasos/mH: Indiferente
- Pasos/mV: Indiferente

A continuación se selecciona CONFIRMAR VALORES y luego el botón de INICIO, habiendo previamente seleccionado el puerto serie a utilizar.

- NumPasosXMetroH y NumPasosXMetroV (Pasos/mH y Pasos/mV)

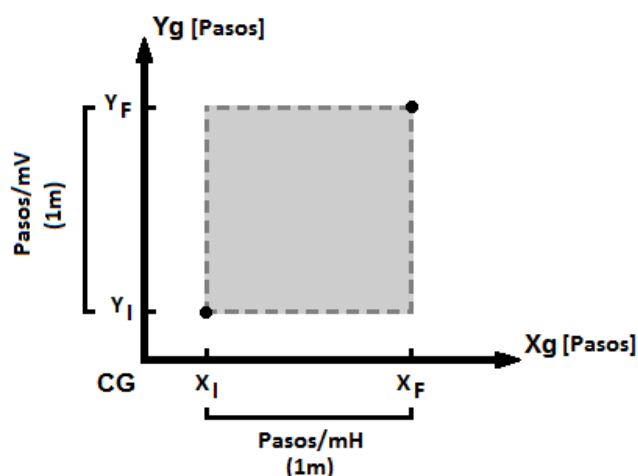
Las variables Pasos/mH y Pasos/mV que como ya se explicó son el número de pasos que deben realizar los motores en cada dirección para conseguir un desplazamiento de un metro.

Para determinar estas variables primeramente se deberá ejecutar la función DETERMINAR CERO. Luego se deberá ejecutar la función MOVIMIENTO LIBRE CONTINUO y ubicar el posicionador en un punto cualquiera denominado INICIAL (no muy lejano al CG) y registrar las coordenadas en pasos (X_I ; Y_I) de este punto. A continuación se hace avanzar el posicionador un metro en cada dirección y se registra las coordenadas en pasos (X_F ; Y_F) de este nuevo punto denominado FINAL. Para medir estos desplazamientos se deberá utilizar cinta métrica o algún otro elemento de medición.

Luego las variables Pasos/mH y Pasos/mV serán simplemente la diferencia entre ambas coordenadas.

$$\text{Pasos/mH} = \text{Coord. Hor. Final} - \text{Coord. Hor. Inicial} = X_F - X_I$$

$$\text{Pasos/mV} = \text{Coord. Vert. Final} - \text{Coord. Vert. Inicial} = Y_F - Y_I$$



Para hacer avanzar el posicionador del punto Inicial al Final se puede utilizar la interfaz para celular de manera que el operario pueda estar dentro del túnel en dicho momento facilitando el trabajo.



- N° Max PH y N° Max PV

Estas variables determinan la zona de trabajo del posicionador y no son otra cosa que las coordenadas máximas en pasos que este puede tomar.

Para determinarlas se deberá ejecutar primeramente la función DETERMINAR CERO. Luego se deberá ejecutar la función MOVIMIENTO LIBRE CONTINUO y simplemente hacer avanzar el posicionador en cada dirección hasta estar próximo a los Fines de Carrera de Protección, o hasta el punto que el operario establezca como punto máximo al cual se pueda avanzar. Luego el valor de la variable N° Max PH será el valor de la coordenada horizontal en pasos de este punto y N° Max PV el de la vertical.

Cabe aclarar entonces que durante este procedimiento, por estar trabajándose con el SRG, la protección contra colisiones mediante la programación estará funcionando, limitándose la zona de trabajo según los valores ingresados inicialmente en las variables N° Max PH y N° Max PV (es decir 99999 en cada dirección). Es por esto que inicialmente se deberá asignar a estas variables valores altos, ya que de lo contrario el posicionador podría detenerse al alcanzar dichos valores antes de poder aproximarse lo suficiente a los fines de carrera de protección.

13.3 ALMACENAMIENTO DE LAS VARIABLES DE AJUSTE

IMPORTANTE: Para comprender este capítulo es necesario primeramente leer los capítulos 17.4. Este se ubica aquí solo por cuestión de orden.

Para que cada vez que se utilice el posicionador (sin haber modificado estructuralmente el mismo o haberlo desmontado y montado nuevamente) no sea necesario ingresar nuevamente el valor de estas variables, las mismas serán almacenadas tanto en la computadora como en el Arduino. De esta manera, el ingreso de las variables se realizará solo una vez luego de montar el posicionador y no será necesario repetirlo hasta después de un nuevo montaje.

El almacenamiento dentro de la computadora se realizará mediante un documento de texto llamado "Variables_de_ajuste.txt" ubicado en la carpeta "data" que se encontrará en la carpeta donde se instaló la aplicación de LabVIEW. Al ejecutar la aplicación de LabVIEW la misma leerá este documento obteniendo así el último valor asignado a cada variable de ajuste y los mostrará en la parte izquierda de la pantalla en la sección VARIABLES DE AJUSTE. Luego, podrá trabajarse con estos mismos valores o modificarlos oprimiendo el botón MODIFICAR VARIABLES. A continuación, al iniciar la aplicación oprimiendo el botón de INICIO, los valores asignados a las variables se enviarán al Arduino al mismo tiempo que se guardarán en un archivo .txt que sobrescribirá al existente, guardándose así estos últimos valores utilizados para la próxima vez que se ejecute la aplicación.

Si bien la HMI principal del posicionador es la aplicación de LabVIEW, el mismo también podrá ser operado solamente a través de la APP para celular y la botonera sin conectarse a la computadora, o conectándose a la misma solo para alimentar al Arduino eléctricamente (sin ejecutar la aplicación). Como recién se explicó, la aplicación de LabVIEW será la encargada de enviarle inicialmente al Arduino el valor de las variables de ajuste, por ello cuando se trabaje sin esta interfaz deberá darse valor a estas variables de otra manera. Es por ello necesario almacenar también en el Arduino los últimos valores asignados a estas variables y de esta manera poder utilizar estos mismos cuando se trabaje solo con la APP y la botonera.

Estos valores se almacenarán en la memoria no volátil EEPROM del Arduino. Cuando se esté utilizando la aplicación de LabVIEW, el Arduino recibirá los valores y los asignará a las variables correspondientes, luego comparará estos valores con los ya almacenados en la memoria EEPROM y en caso de ser distintos sobrescribirá la memoria con los nuevos valores recibidos. Para ello se utilizará la función EEPROM.put la cual reescribe la memoria EEPROM solo cuando lo que se desea escribir es distinto a lo que ya se



encuentra almacenado, y además, esta función puede escribir valores que superen la capacidad de un Byte (valores mayores a 255).

Por su parte la memoria EEPROM del Arduino tiene una vida útil de 100.000 operaciones de escritura y no tiene límites para la operación de lectura. Teniendo en cuenta que la misma solo se escribirá una vez en cada encendido o reseteo del Arduino la vida útil de la misma es enormemente superior a la necesaria, más aun si se tiene en cuenta que no siempre se escribirá en ella, sino solamente cuando se modifiquen los valores ingresados desde LabVIEW.



14. DESCRIPCIÓN DE LAS FUNCIONES

▪ FUNCIÓN 1: DETERMINAR CERO

Esta función realizará el cambio de sistema de referencia del Inicial al Global. De esta manera podrá entonces ejecutarse la Protección Contra Colisiones Mediante la Programación y desbloquearse todas las funciones del posicionador.

Como se explicó anteriormente, al encender o reiniciar la placa Arduino se comenzará utilizando el SRI. El origen de este sistema de referencia no será fijo si no variable, ya que se tomará la ubicación del posicionador al momento de encender la placa como origen de coordenadas. Debido a que este sistema no permitirá conocer la ubicación exacta del posicionador respecto de la estructura no se podrá ejecutar la Protección Contra Colisiones Mediante la Programación, por lo que algunas de las funciones se bloquearán por seguridad.

Para conocer la ubicación exacta del posicionador respecto de la estructura se deberá trabajar con el SRG, ya que este sistema poseerá un origen de coordenadas fijo denominado CERO GLOBAL, el cual se determinará a partir de un punto de referencias físico PRF. El PRF por su parte quedará determinado por dos sensores denominados Sensores de Posición los cuales al ser bloqueados simultáneamente por el posicionador indicarán que este se encuentra sobre el PRF.

La función DETERMINAR CERO se encargará entonces de localizar automáticamente la posición de este CERO GLOBAL para así poder trabajar con el Sistema de Referencias Global y habilitar todas las funciones.

Al ejecutar la función, el posicionador retrocederá horizontal y verticalmente hasta bloquear los Sensores de Posición. Cada movimiento será independiente del otro, y retrocederá hasta bloquear su sensor correspondiente, momento en el cual se detendrá y tomará esta posición como origen del Sistema de Referencia Global o CERO GLOBAL.

En caso de que inicialmente los sensores ya se encuentren bloqueados el posicionador avanzará en sentido positivo hasta que los mismos sean liberados para recién luego iniciar el proceso de retroceso. Si al producirse un avance considerable algún sensor permaneciera bloqueado se considerará una obstrucción o problema en el mismo o el circuito de alimentación de este, por lo que se dará una señal de alarma y finalizará la función.

▪ FUNCIÓN 2: MOVIMIENTO LIBRE CONTINUO

Mediante esta función se podrá hacer avanzar al posicionador de forma continua en cualquiera de las direcciones mientras se mantenga accionado el botón direccional correspondiente.

En esta función, al accionar uno de los botones de avance en alguna interfaz, el posicionador se desplazará permanentemente en el sentido correspondiente hasta que el botón sea liberado.

Si se está trabajando con el Sistema de Referencias Global, esta función se ejecutará con Protección Contra Colisiones Mediante la Programación. Por ello, durante el desplazamiento se comprobará constantemente que el posicionador no haya alcanzado alguno de los extremos de la zona de trabajo. En el caso de haberlo hecho la programación impedirá que se siga avanzando en este sentido e indicará al usuario a través de las interfaces que se ha llegado a uno de los límites y a cuál de ellos ha sido.

Hasta no haberse ejecutado la función DETERMINAR CERO y por ello estarse trabajando con el Sistema de Referencia Inicial, esta función se ejecutará sin la protección contra colisiones mediante la programación recién mencionada. Deberá tenerse entonces cuidado por parte del usuario si se está moviendo el posicionador cerca de los límites de la estructura. De todas maneras, una colisión contra la estructura sería evitada por la Protección Mediante Fines de Carrera de Protección, pero en este caso se



produciría la desconexión de los drivers de los motores y debería procederse a su reconexión para poder continuar trabajando.

Uso de las memorias:

Si el punto donde se encuentre el posicionador se considerara de interés y sea necesario regresar al mismo en el futuro, podrá guardarse esa posición en una de las dos memorias disponibles MEM 1 y MEM2. Luego al ejecutar la función de IR A MEMORIA el posicionador se desplazará hasta la posición del punto guardado en la memoria correspondiente.

Si la memoria donde desea guardarse la posición actual ya almacenara una posición guardada anteriormente, esta será sobrescrita por la nueva.

Las posiciones guardadas en las memorias se perderán al encender o reiniciar el Arduino ya que serán almacenadas en la memoria SRAM del mismo.

Hasta tanto no sean utilizadas las memorias, estas contendrán la posición del origen del sistema de referencia en uso. Es decir, si alguna de las memorias no fue utilizada para almacenar una posición particular, al ser ejecutada la función IR A MEMORIA correspondiente, el posicionador se dirigirá al origen del sistema de referencia con el cual se esté trabajando en ese momento.

En caso de que se guardara una posición estando trabajando con el sistema inicial y luego se empezara a trabajar con el sistema global, la programación realizará la conversión de coordenadas de un sistema al otro, de esta manera las posiciones memorizadas se mantendrán invariables. En particular, si trabajando con el sistema de referencia inicial se guardara una posición que luego se encontrara fuera de los límites del sistema de referencia global, al pasar del sistema inicial al global esta memoria se borrará y en su lugar se guardarán las coordenadas (0 ; 0).

▪ FUNCIÓN 3: MOVIMIENTO LIBRE ESCALONADO

Con esta función al accionar uno de los botones de avance en alguna interfaz el posicionador se desplazará 10 pasos en la dirección correspondiente y se detendrá aun si el botón permaneciera accionado. Para volver a avanzar otros 10 pasos el botón deberá ser liberado y vuelto a accionar.

El funcionamiento de las protecciones contra colisiones según cual sea el sistema de referencia utilizado y el uso de las memorias es idéntico al explicado en la función anterior.

▪ FUNCIÓN 4: IR A MEMORIA 1

Al ejecutarse esta función el posicionado se desplazará hasta las coordenadas del punto guardado en la Memoria 1.

▪ FUNCIÓN 5: IR A MEMORIA 2

Al ejecutarse esta función el posicionado se desplazará hasta las coordenadas del punto guardado en la Memoria 2.

▪ FUNCIÓN 6: MEDICIÓN AUTOMÁTICA

Esta función permitirá realizar múltiples mediciones en forma de cuadrícula de manera automática. Para esto se partirá del Sistema de Referencia Relativo, el cual tomará como origen al punto donde se encuentre el posicionador al momento de iniciar esta función. La unidad de los ejes de este sistema será

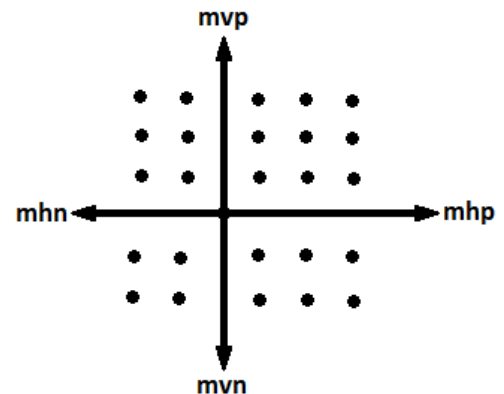


en Número de Mediciones, siendo la distancia entre ellos una variable a definir por el operario. A partir del punto de origen el usuario indicará la cantidad de mediciones que desee realizar en cada dirección y sentido, la distancia entre ellos y los Tiempos de Espera y Muestreo.

Ingreso de parámetros para la medición

En cada dirección deberán indicarse el número de mediciones en uno y otro sentido. La distancia entre mediciones en una misma dirección será siempre la misma. Deberá especificarse entonces:

- Numero de mediciones en el sentido horizontal positivo (mhp).
- Numero de mediciones en el sentido horizontal negativo (mhn).
- Numero de mediciones en el sentido vertical positivo (mvp).
- Numero de mediciones en el sentido vertical negativo (mvn).
- Distancia deseada entre mediciones horizontales en mm (dmh).
- Distancia deseada entre mediciones verticales en mm (dmv).



Los números de mediciones en sentido negativo se ingresarán en la aplicación de LabVIEW como valores positivos ya que solo indican la cantidad de mediciones en ese sentido.

Deberán especificarse también los tiempos:

- Tiempo de Espera en Seg
- Tiempo de Muestreo en mSeg

El Tiempo de Espera será la pausa realizada antes de iniciar el muestreo en cada punto de medición. Esta pausa se realizará debido a que, en el instante en que el posicionador alcanzara un punto de medición, la presión a lo largo de las mangueras que conectan el instrumento con los sensores de presión no será constante. Se presenta en este momento un periodo transitorio que puede durar algunos segundos según el largo de las mangueras, en el cual no pueden realizarse mediciones ya que se producirían importantes errores. Por ello debe esperarse que finalice este transitorio y se estabilicen las condiciones dentro de las mangueras para iniciar el muestreo.

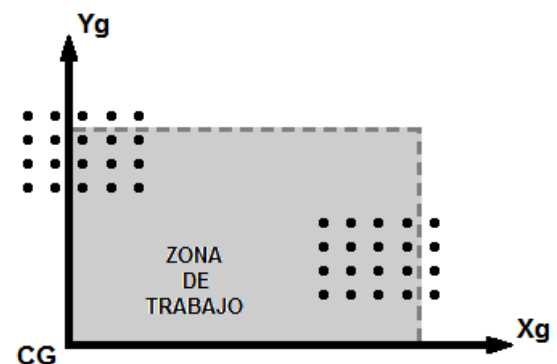
El Tiempo de Muestreo será la duración de la medición. Como se explica en la introducción el posicionador realizará la tarea de ubicar al instrumento en el punto donde se desea realizar un muestreo y luego un Sistema de Adquisición (creado separadamente en función del ensayo) será el encargado de la toma, almacenamiento y análisis de datos provenientes de los sensores. Esta toma de datos por parte del Sistema de Adquisición se deberá realizar en los momentos en que el posicionador se lo indique mediante una Señal de referencia. Esta señal será digital y tomará valor HIGH (5V) en el momento de realizar los muestreos y LOW (0V) cuando esta deba finalizar. El Tiempo de Muestreo será entonces el tiempo en que se encontrara activada esta Señal de Referencia.

Recepción y confirmación de datos

Luego de ingresar los 8 parámetros deseados en la aplicación de LabVIEW, al presionar la tecla ENVIAR VALORES estos se enviarán al Arduino el cual analizará los mismos y los devolverá a la aplicación a modo de confirmación para su visualización en pantalla. Además de esto el Arduino calculará y enviará también la duración aproximada de todo el proceso de medición, la distancia real entre mediciones y la existencia de desbordes.

En los parámetros de la medición se ingresará la distancia deseada entre mediciones en mm. Ahora bien, debido al avance discreto de los motores, no existirá un número de pasos que coincida exactamente con la distancia en mm deseada. La distancia deseada caería entonces dentro del intervalo producido por dos pasos consecutivos (una determinada cantidad de pasos corresponderá a una distancia menor a la deseada y con un paso extra a una distancia mayor). En este caso la programación determinará con cuál de estas dos posiciones se produce un menor error y trabajará con esta. Esta distancia real será informada al usuario para que el mismo la tenga en cuenta durante el análisis de datos ya que, si bien la diferencia entre la distancia real y la deseada podrá ser despreciable en una única medición, al realizar numerosas mediciones consecutivas el error se irá incrementando hasta volverse importante. Así por ejemplo si se desean realizar 150 mediciones cada 10 mm, entre el primer y el último punto de medición deberá haber 1500 mm de distancia. En cambio si la separación real entre mediciones fuera de 10,06 mm, la distancia total real sería de 1509 mm. Se tendría entonces un error de 9 mm, comparable a la separación entre dos mediciones.

Los desbordes se producirán cuando al definir la cantidad y distancia entre los puntos de medición, parte de ellos se ubicaran fuera de la zona de trabajo. En la siguiente figura se muestran dos ejemplos de ello. En un caso se produce solo desborde horizontal positivo y en el otro desborde horizontal negativo y vertical positivo. En estos casos en la aplicación de LabVIEW se indicará las direcciones en las cuales se han producido los desbordes y se impedirá el proceso de medición bloqueando el botón INICIAR MEDICIONES. Luego podrá seleccionarse la opción MODIFICAR VALORES con el cual se reingresarán los valores a fin de corregir los que produjeron desbordes o directamente cancelarse el proceso de medición mediante el botón CANCELAR. Este cálculo de desbordes formará parte de la Protección Contra Colisiones Mediante la Programación. Es por esto que esta función solo podrá utilizarse si se está trabajando con el Sistema de Referencias Global, permaneciendo entonces bloqueada hasta no ejecutar la función DETERMINAR CERO.



Proceso de medición

En caso de no presentarse problemas de desbordes podrá iniciarse el proceso de medición automática pulsando el botón de INICIAR MEDICIONES en la aplicación de LabVIEW. En ese momento el posicionador retrocederá hasta el punto de medición inferior izquierdo de la cuadrícula. Al llegar al mismo comenzará el Tiempo de Espera permitiendo que se establezca la presión dentro de las mangueras. Finalizado este se activará la Señal de Referencia poniéndola en HIGH durante el Tiempo de Muestreo. Finalizado este tiempo la señal de referencia se pondrá en LOW y el posicionado avanzará en sentido horizontal positivo hasta el siguiente punto repitiendo allí el proceso de medición. Ya alcanzado el último punto de la fila el posicionador retrocederá horizontalmente hasta la primera columna y avanzará verticalmente hasta la siguiente fila de mediciones. Este proceso se repetirá hasta censar todos los puntos de medición de la cuadrícula. Al finalizar el posicionador se dirigirá nuevamente al punto de origen del Sistema Relativo y se



finalizará el proceso de medición automática saliendo de esta función en el mismo punto donde se la inició.

Durante el proceso de medición en la interfaz LabVIEW podrá observarse el porcentaje realizado del proceso y las coordenadas relativas del punto en el cual se está realizando la medición.

En cualquier momento mientras las mediciones se estén realizando el proceso podrá ser cancelado. En ese caso se esperará a que finalice la medición en el punto actual y luego el posicionador se dirigirá nuevamente al punto de origen del Sistema Relativo para luego salir de la función.

▪ FUNCIÓN 7: IR A COORDENADAS

Mediante esta función el operador podrá enviar al posicionador directamente al punto que desee ingresando las coordenadas del mismo.

Esta función podrá ejecutarse de dos maneras:

- Coordenadas en pasos
- Coordenadas en mm

De esta manera, luego de seleccionar el tipo de coordenadas y habiendo ingresado los valores de estas, al oprimir el botón IR el posicionador se dirigirá a la coordenada deseada.

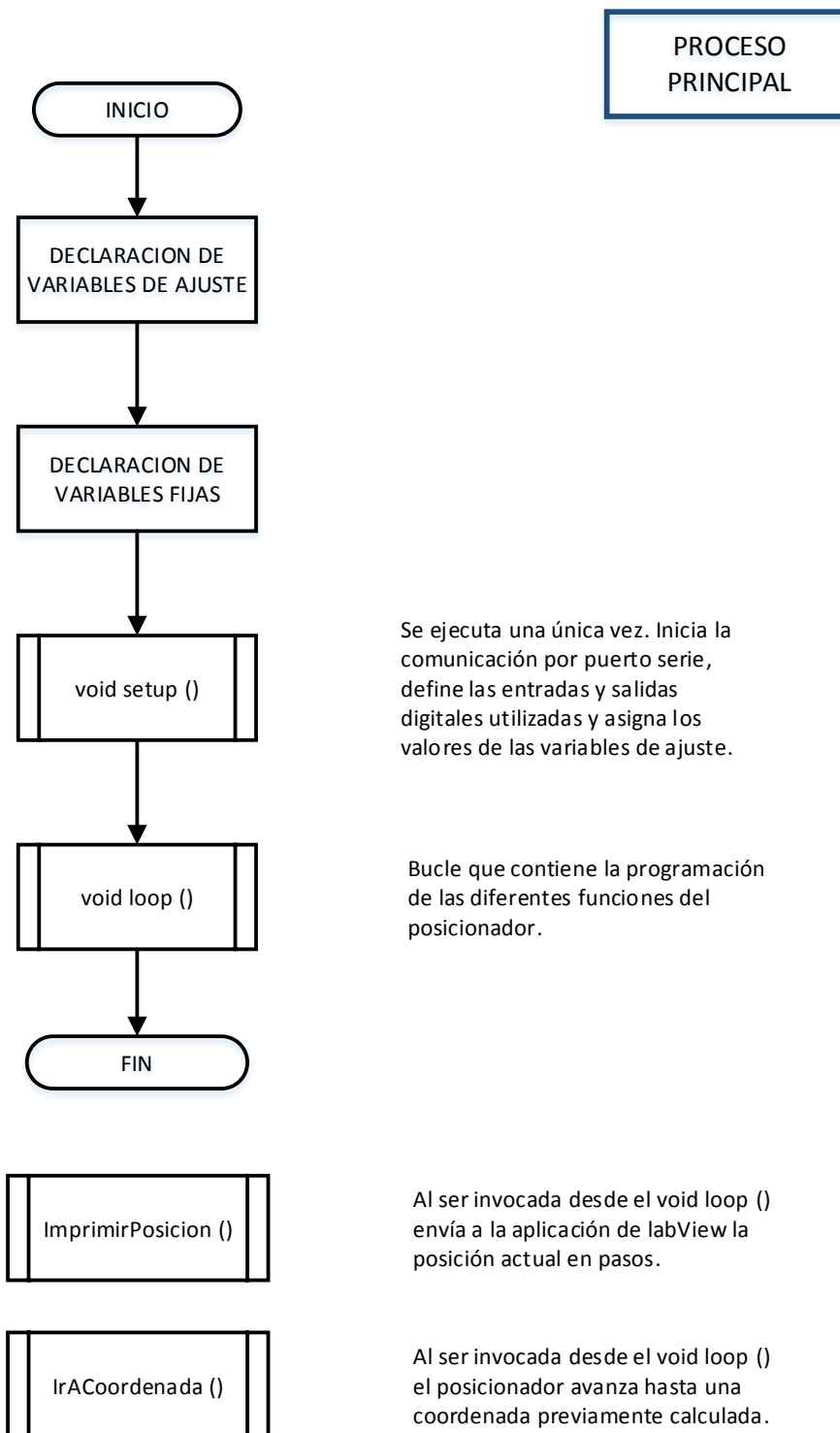
Así como se explicó en la función anterior si se deseara ingresar las coordenadas en mm, debido al avance discreto de los motores, no existirá un número de pasos que coincida exactamente con la posición deseada por lo que la posición que podrá obtenerse será anterior o posterior a este punto. La programación determinará entonces en cuál de estas posiciones se producirá un menor error respecto de la posición deseada y luego se dirigirá a este.

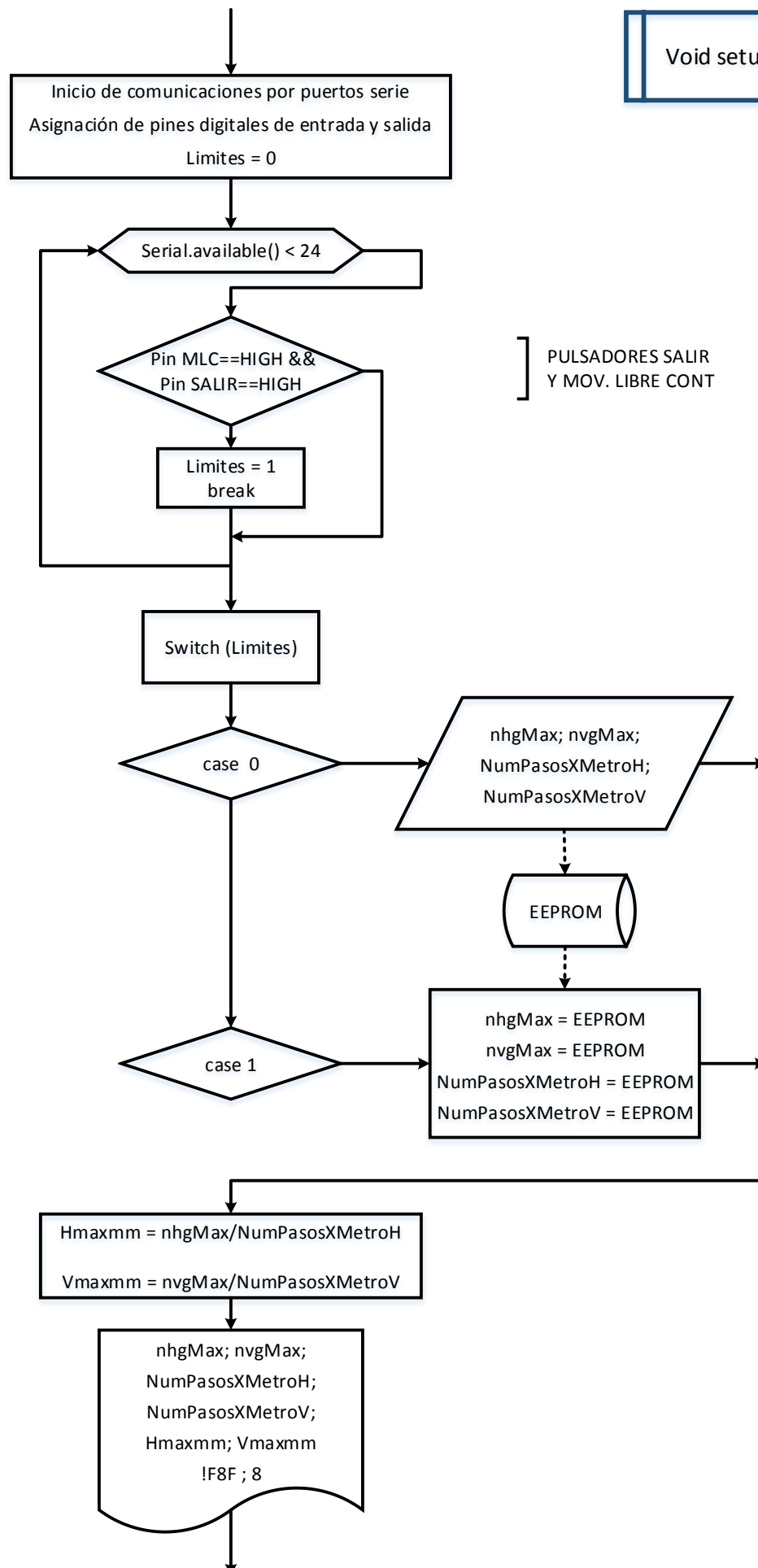
Esta función utiliza la Protección Contra Colisiones Mediante la Programación por lo que solo podrá trabajar con el SRG, permaneciendo la función bloqueada hasta no ejecutar la función DETERMINAR CERO. Por ello los valores de coordenadas ingresados solo podrán estar dentro de la zona de trabajo. En primer lugar como el SRG solo posee coordenadas positivas, los valores de coordenadas ingresados deberán ser mayores o iguales a cero. Por otro lado, si alguno de los valores ingresados superara la zona de trabajo se desplegará una ventana en LabVIEW indicando que las coordenadas ingresadas están fuera de los límites e indicará a su vez el rango de valores que podrían ingresarse.

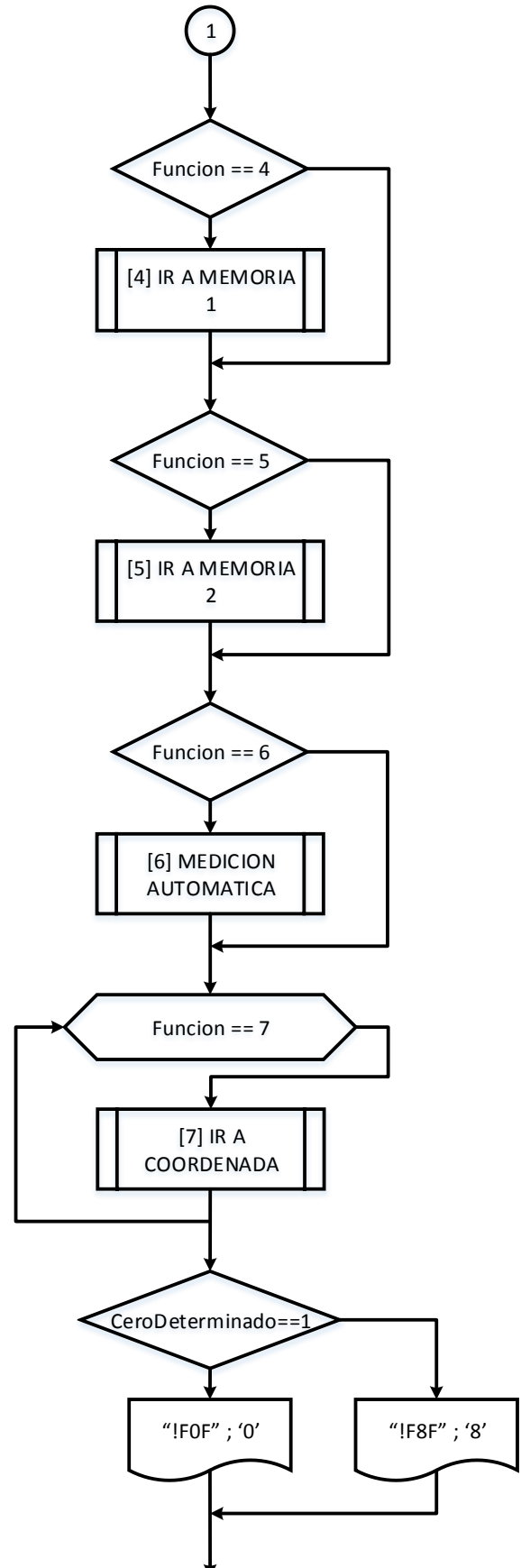
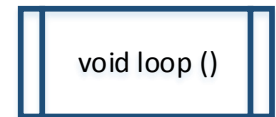
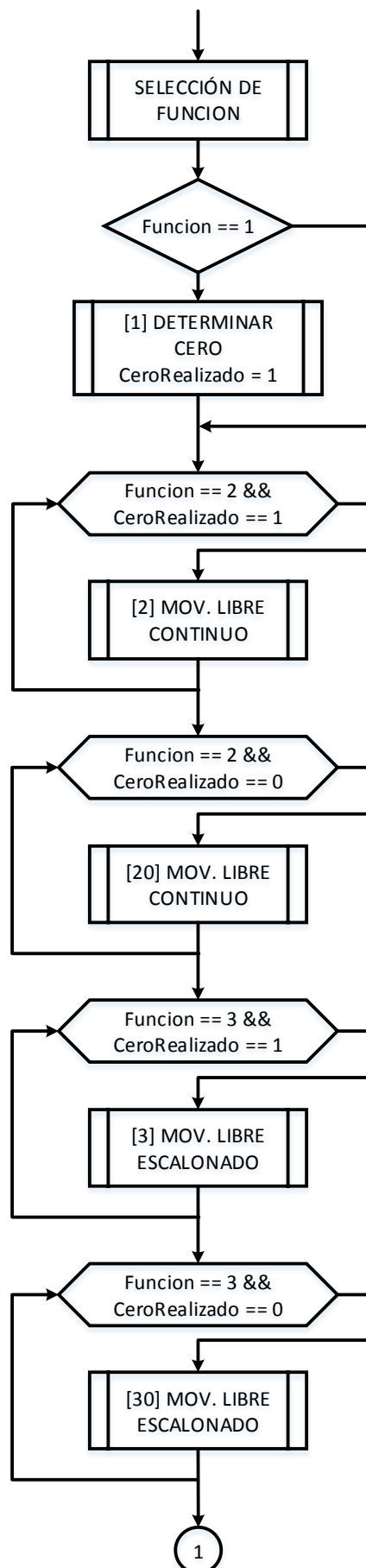


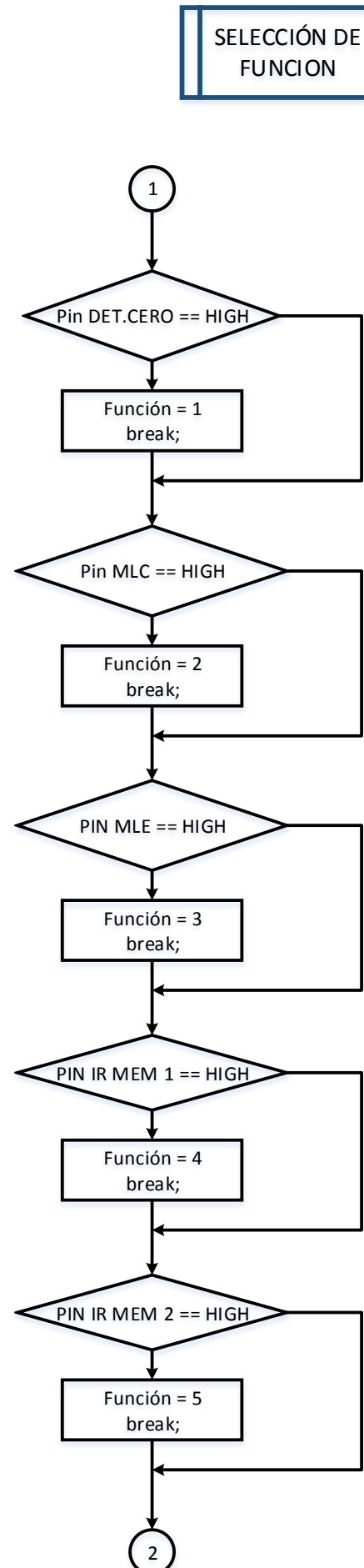
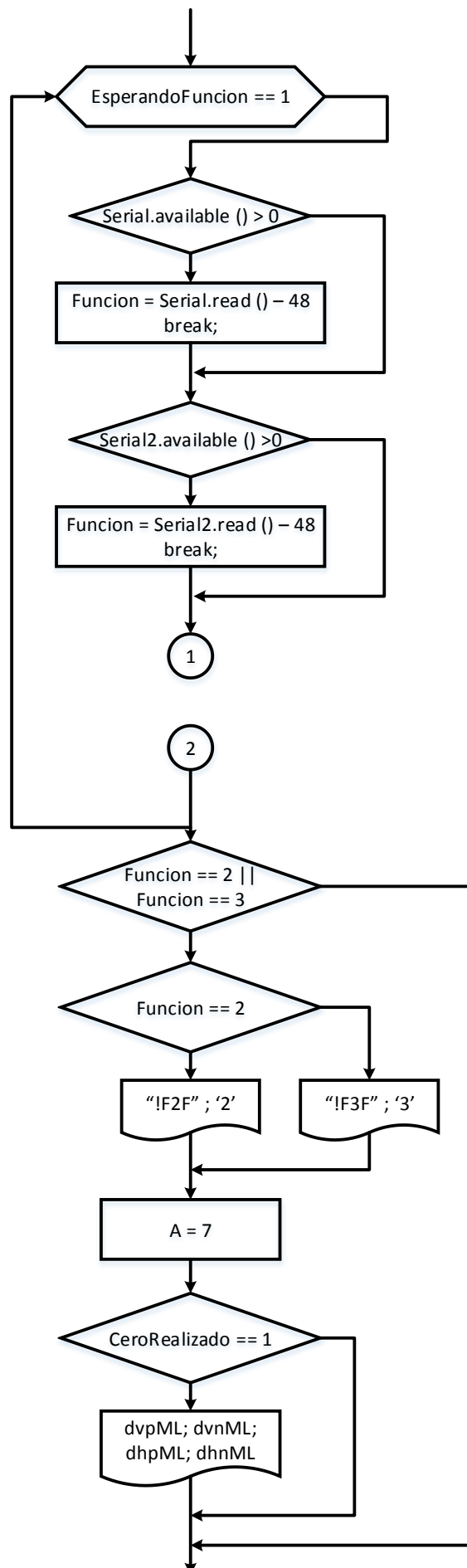
15. PROGRAMACIÓN DEL SISTEMA DE CONTROL – ARDUINO

15.1 DIAGRAMAS DE FLUJO RESUMIDOS DE LA PROGRAMACIÓN

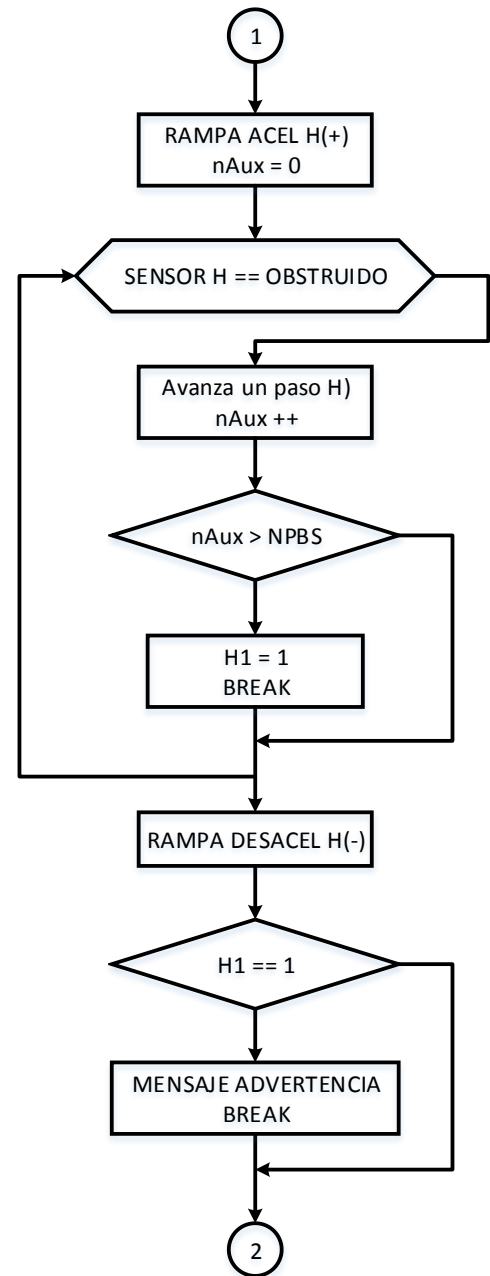
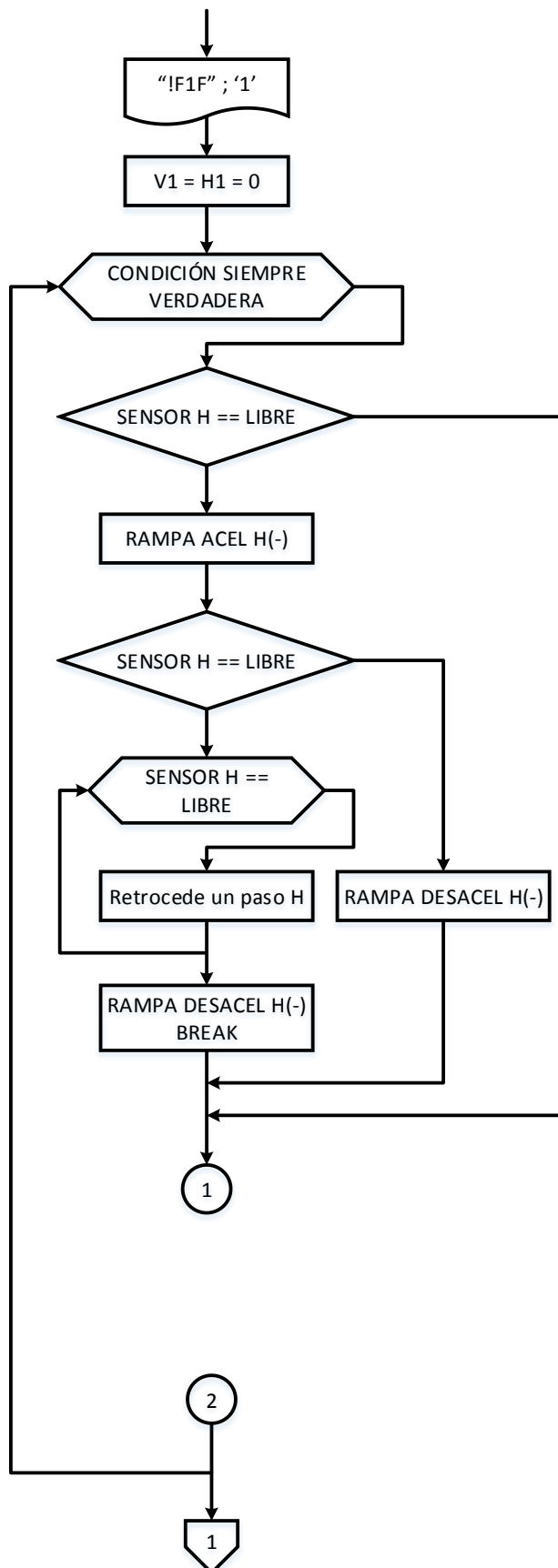




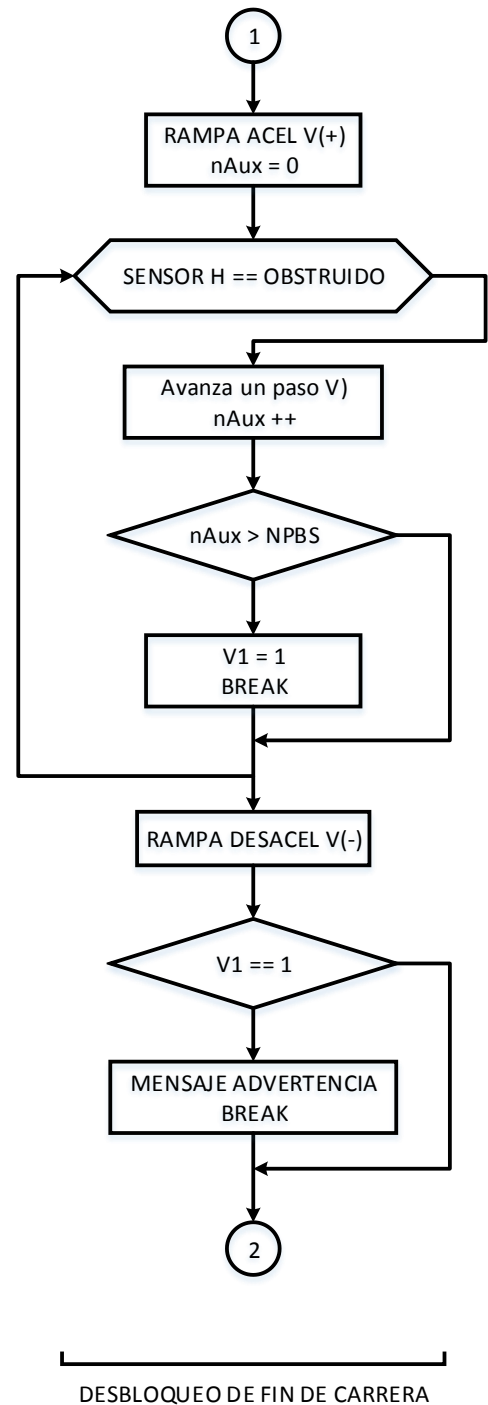
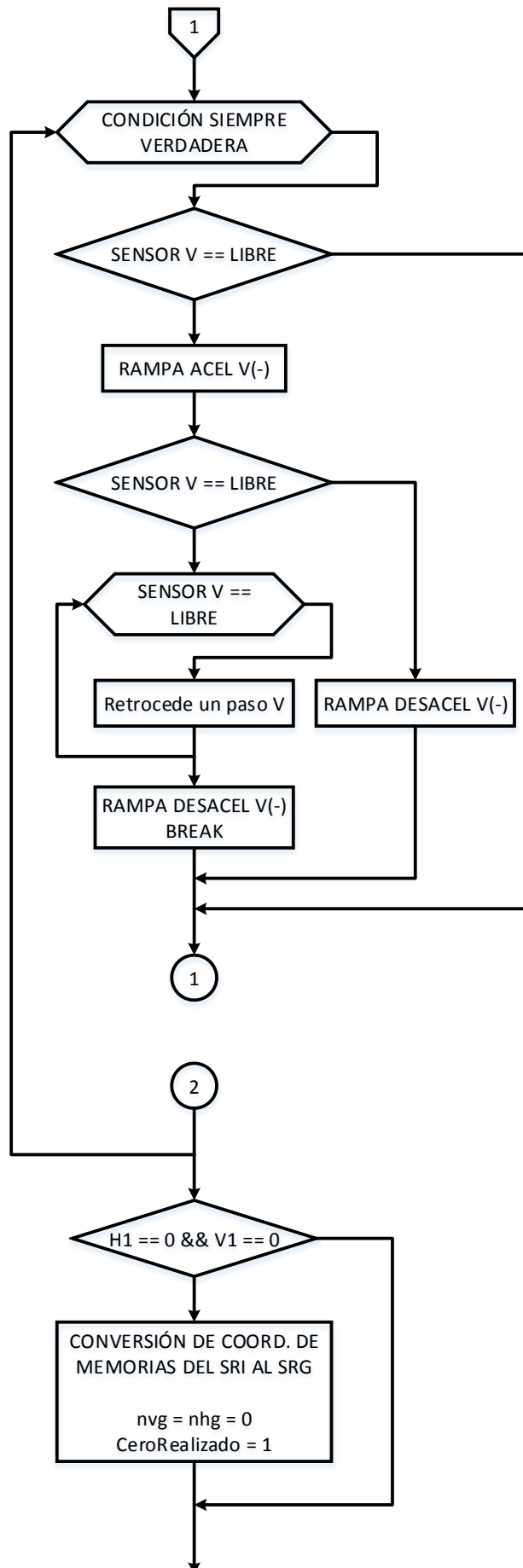




[1] DETERMINAR
CERO



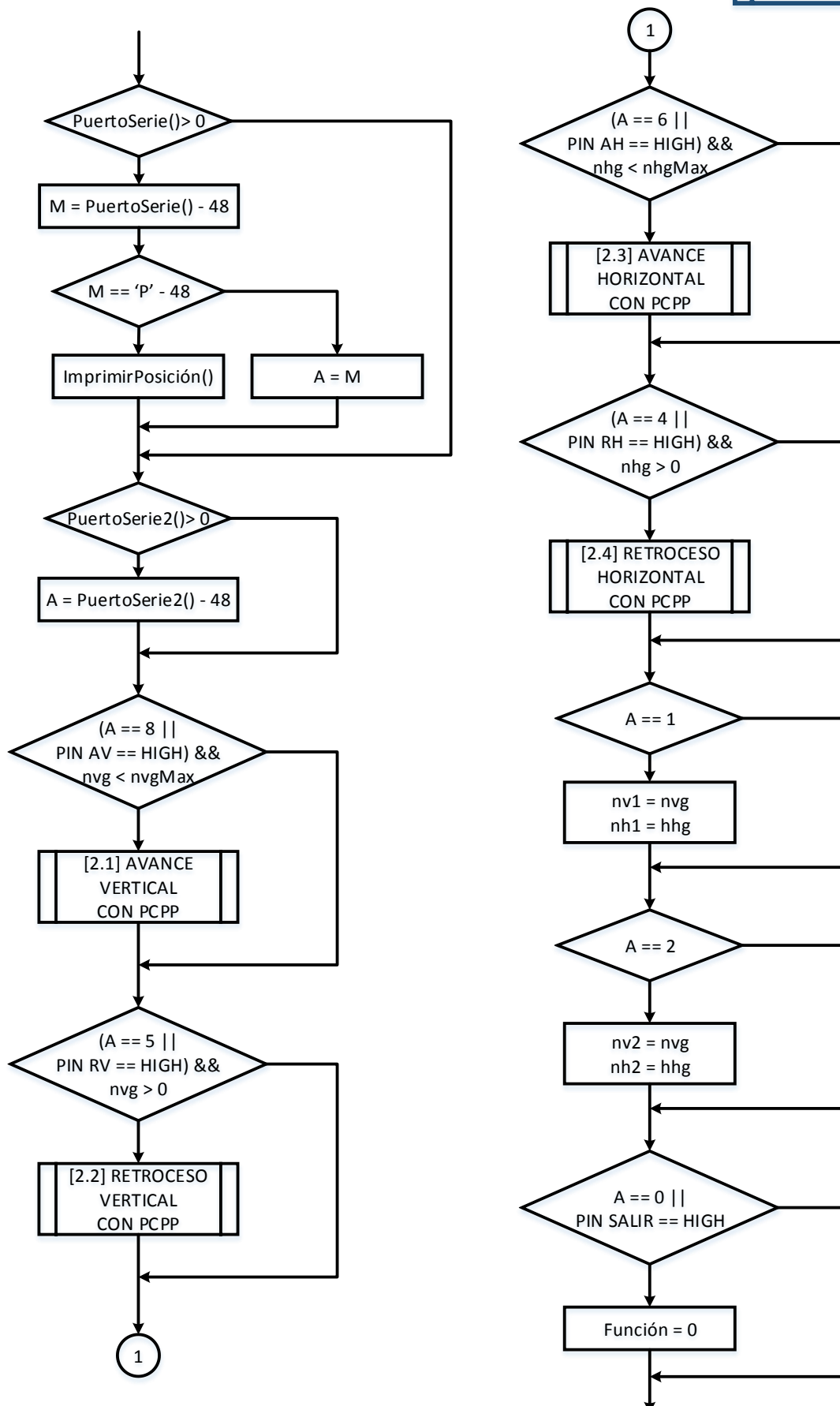
DESBLOQUEO DE FIN DE CARRERA



DESBLOQUEO DE FIN DE CARRERA

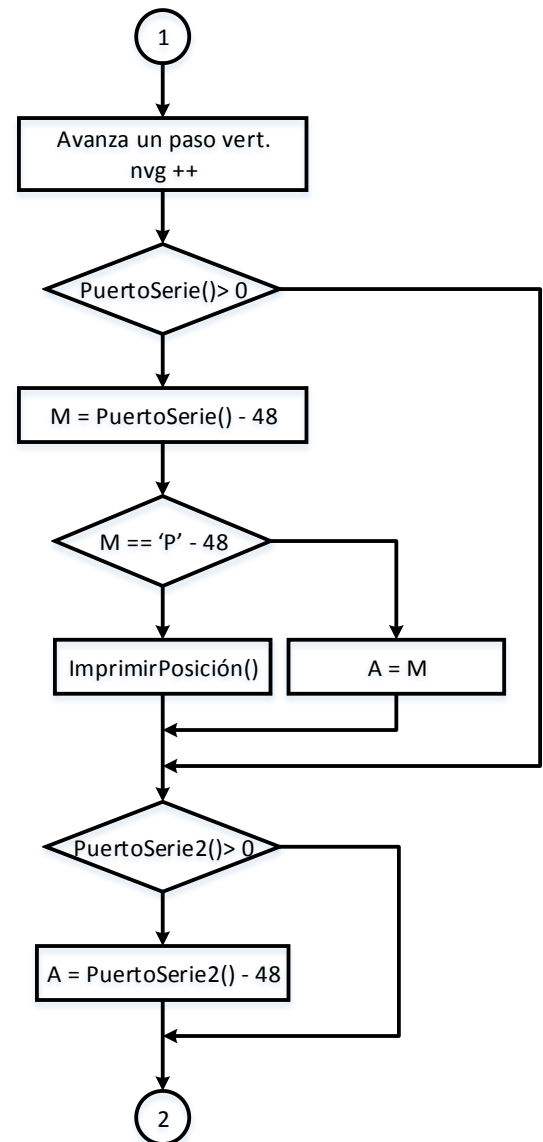
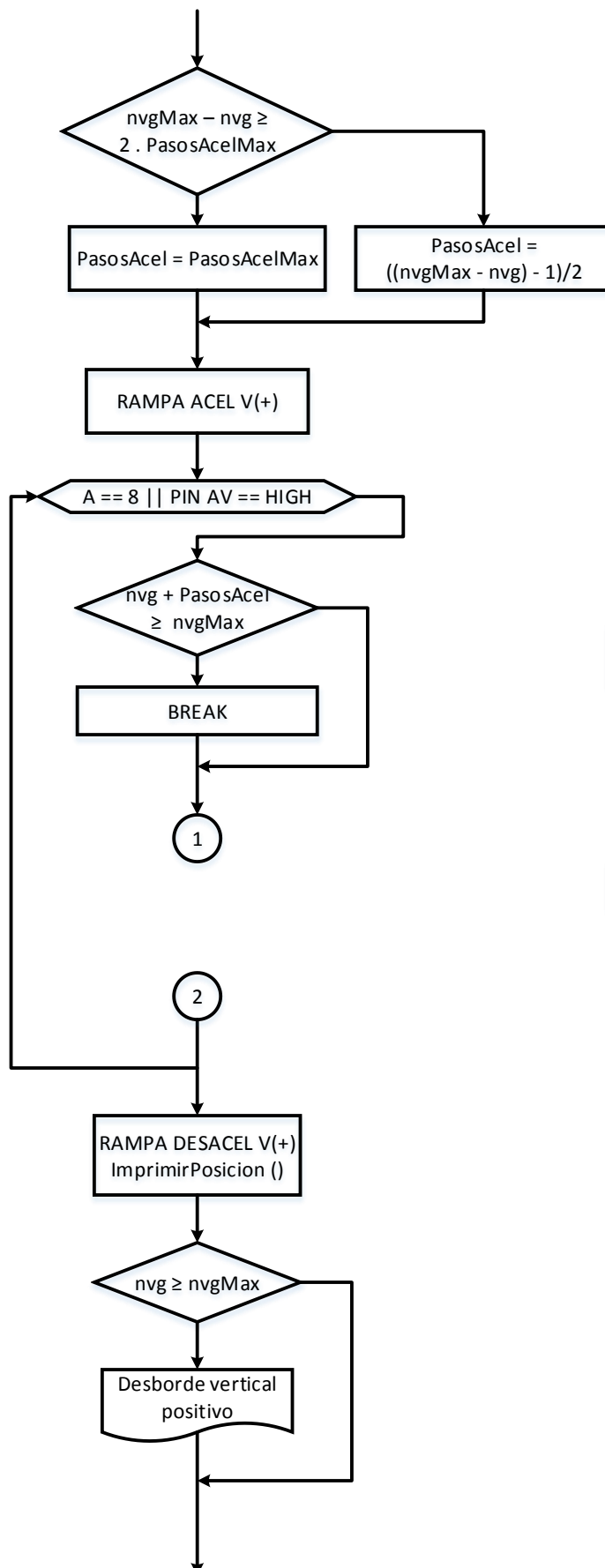


[2] MOV. LIBRE
CONTINUO



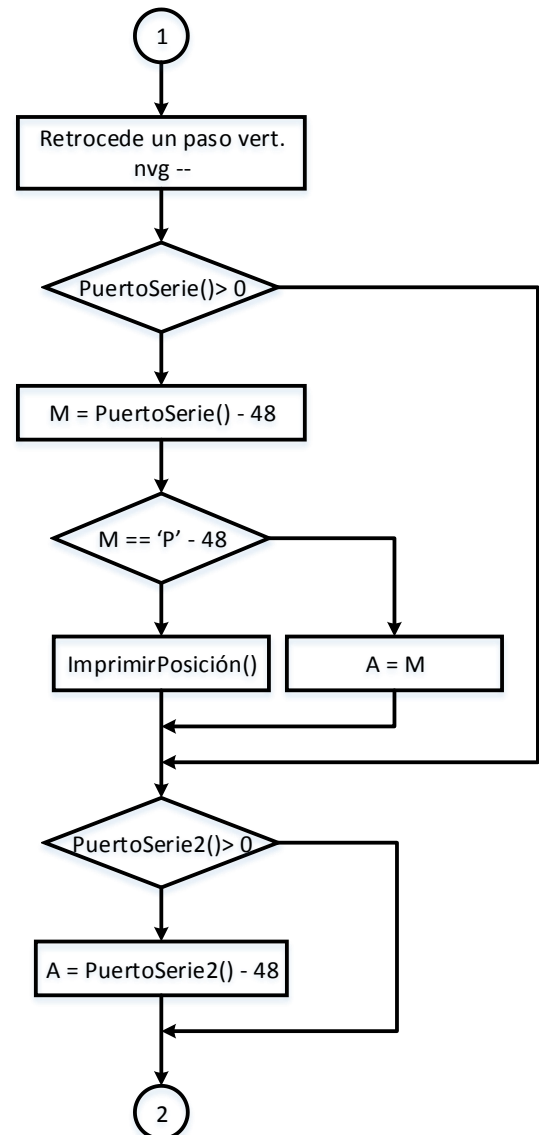
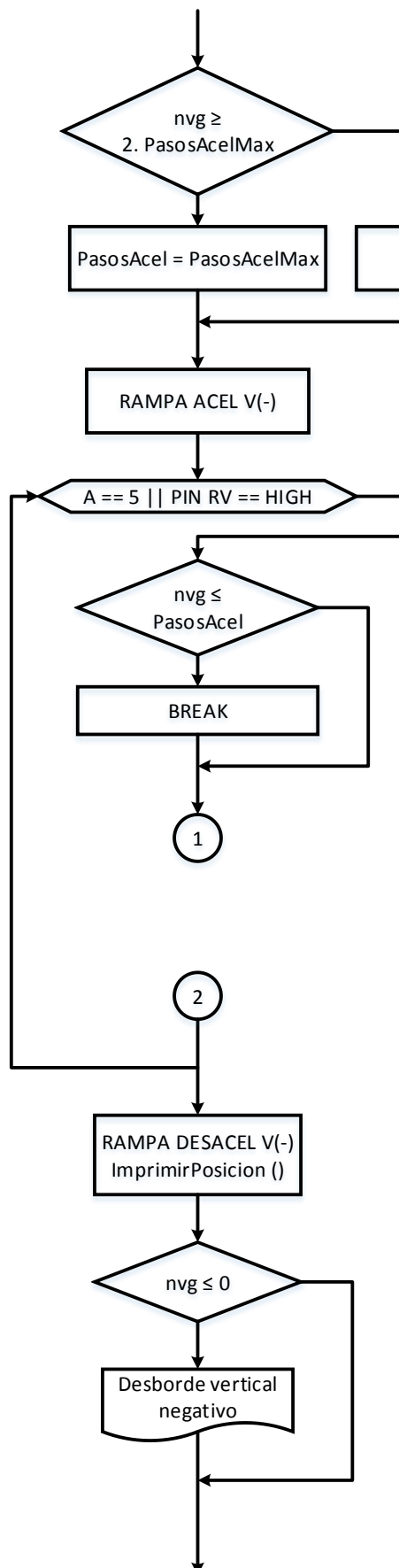


[2.1] AVANCE
VERTICAL
CON PCPP

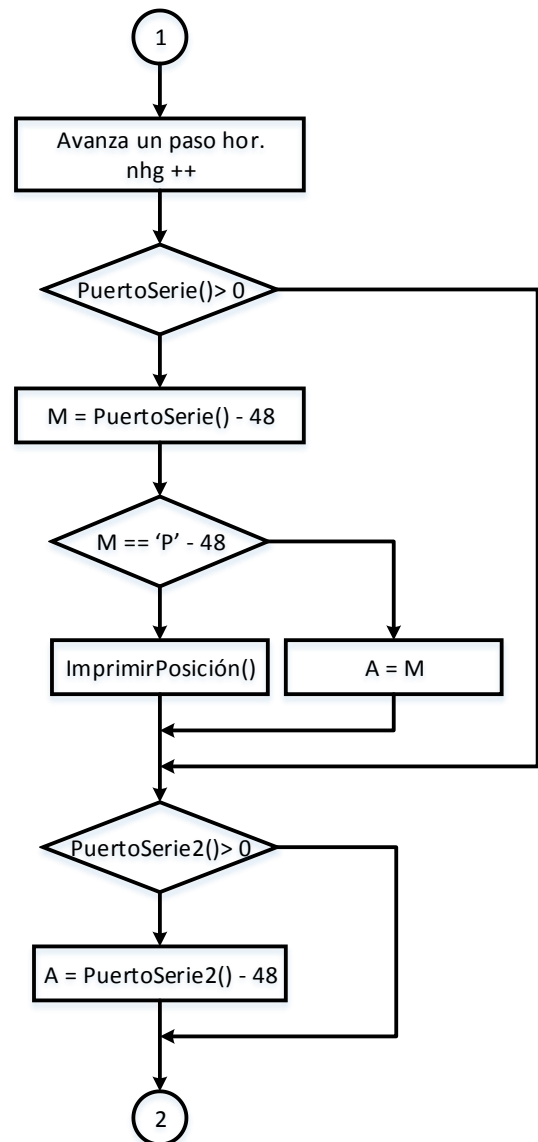
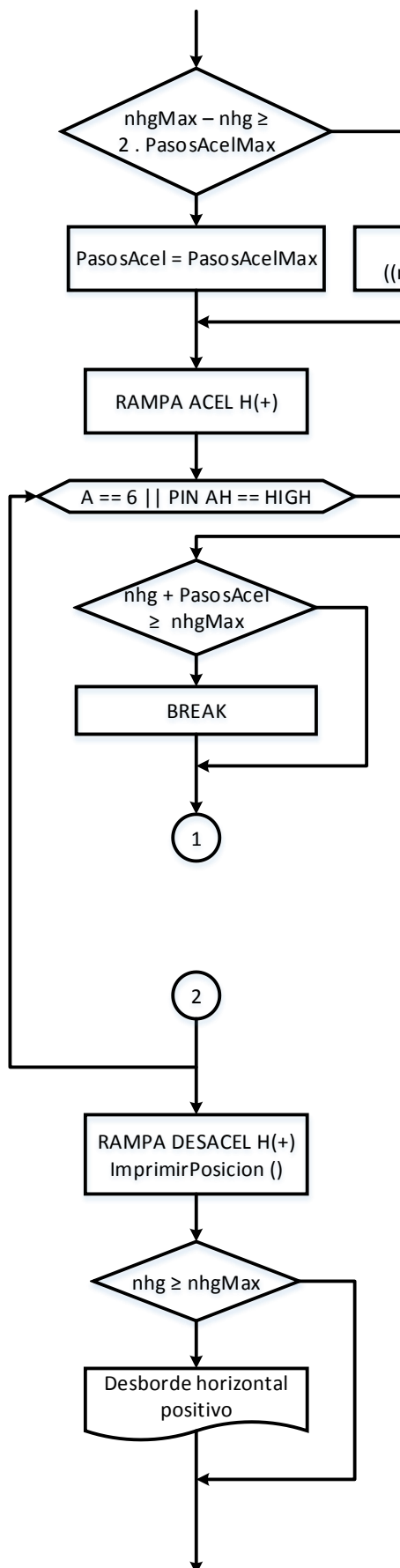




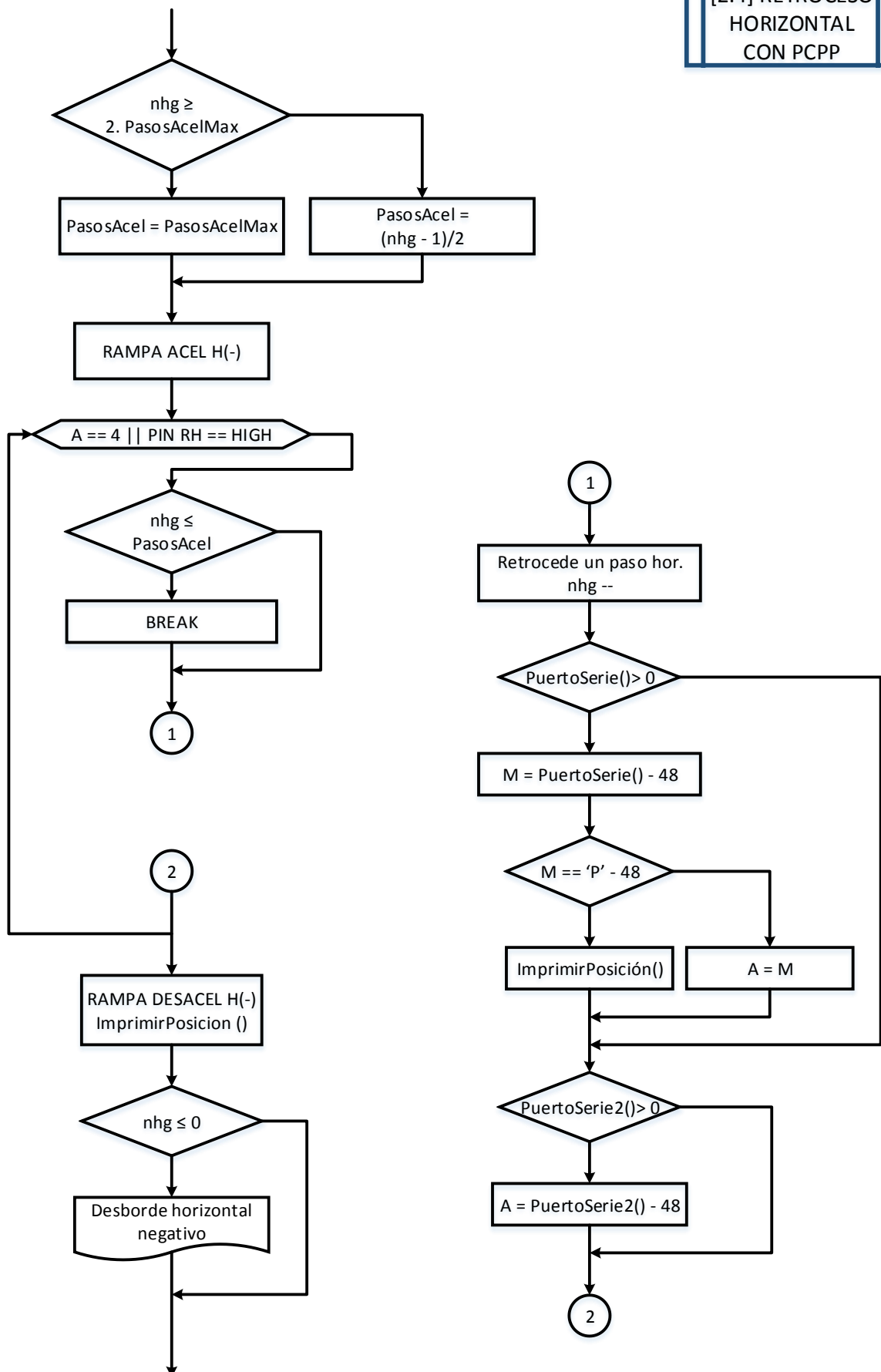
[2.2] RETROCESO
VERTICAL
CON PCPP



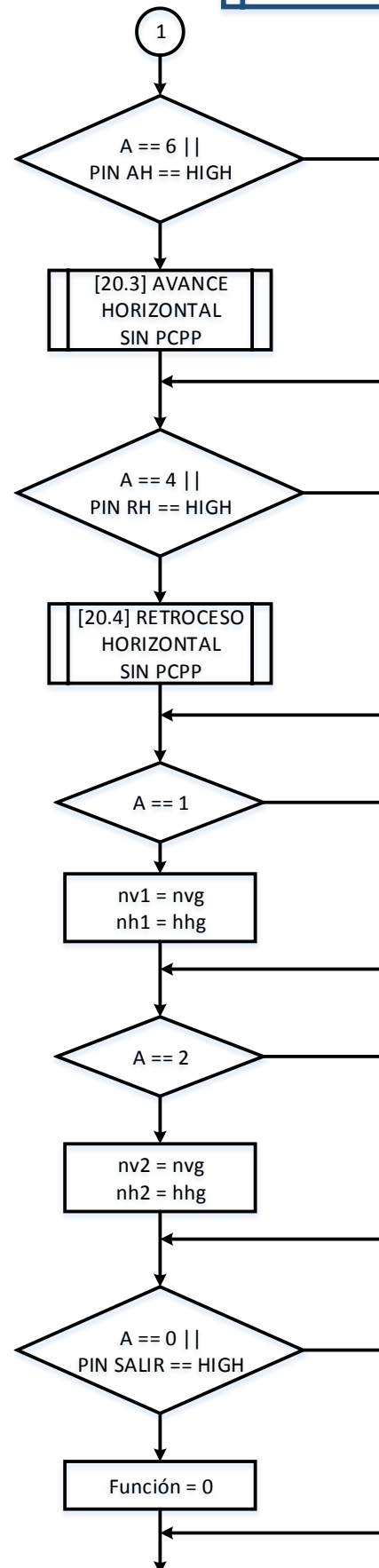
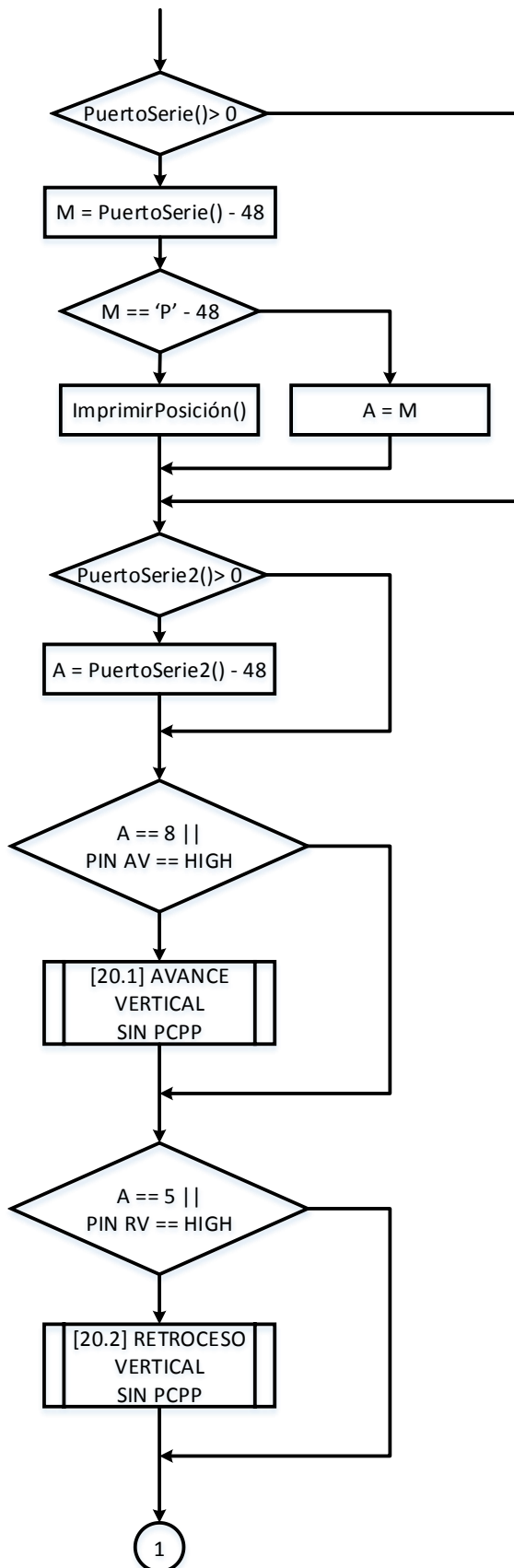
[2.3] AVANCE
HORIZONTAL
CON PCPP



[2.4] RETROCESO
HORIZONTAL
CON PCPP

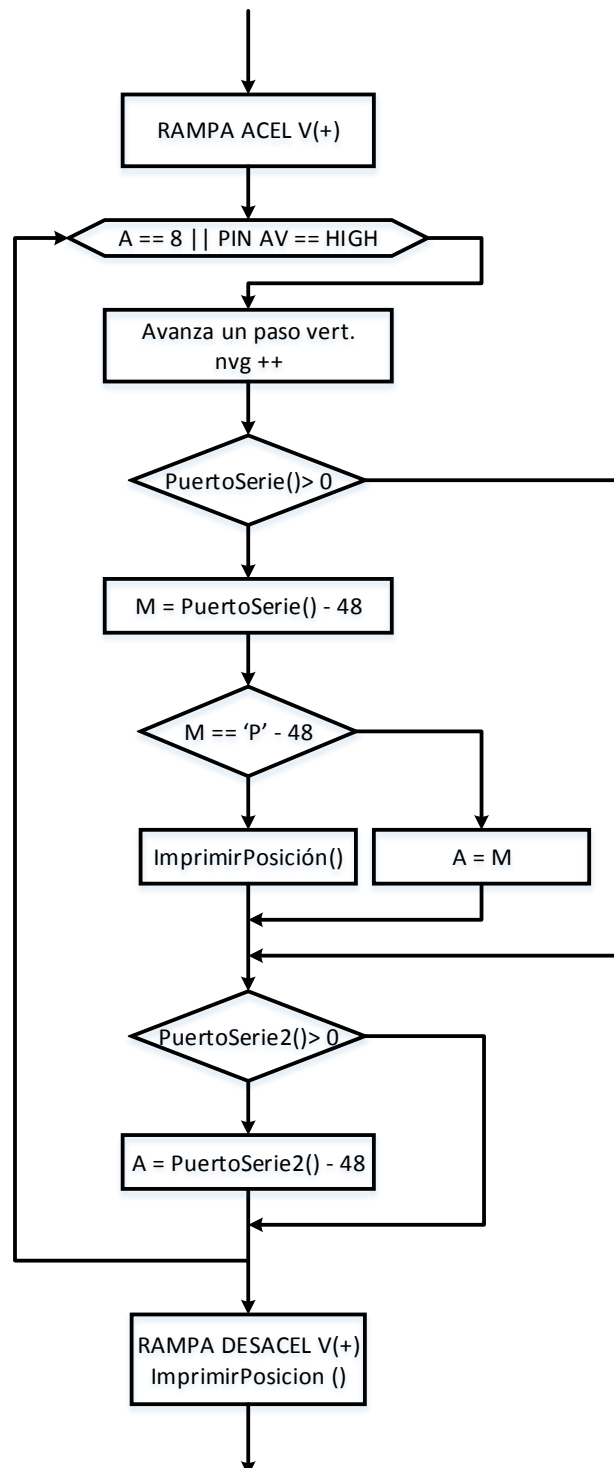


[20] MOV. LIBRE
CONTINUO



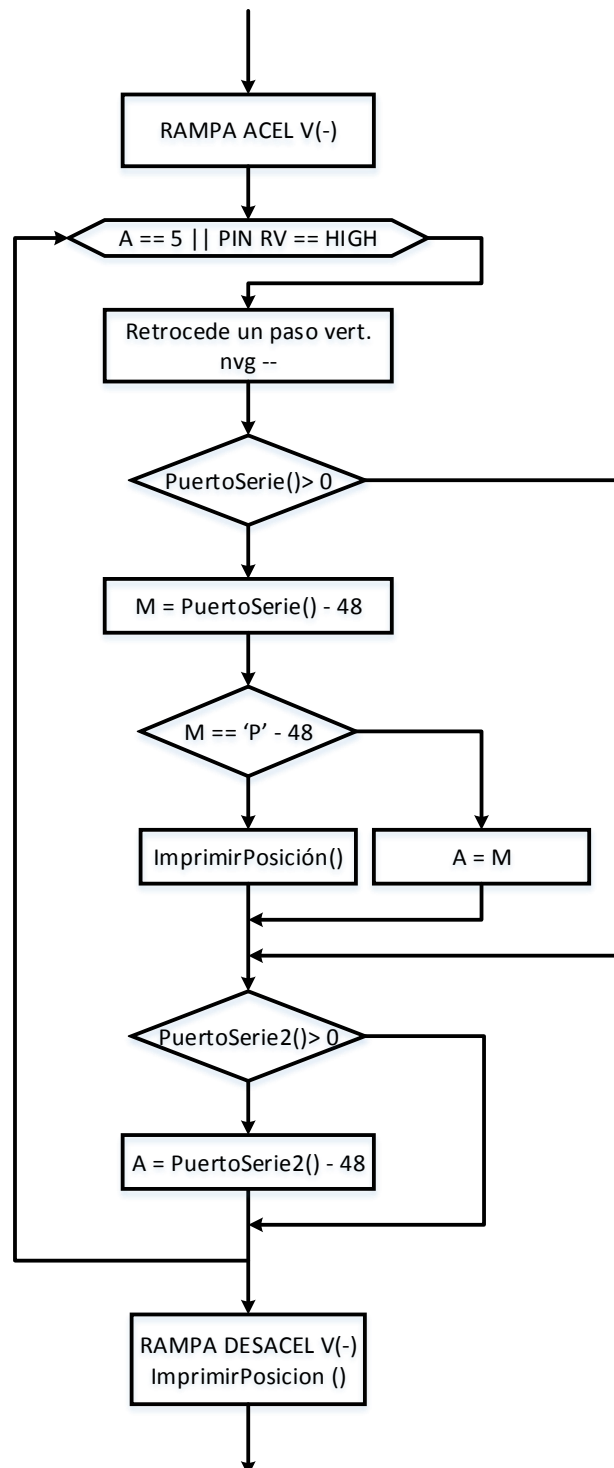


[20.1] AVANCE
VERTICAL
SIN PCPP



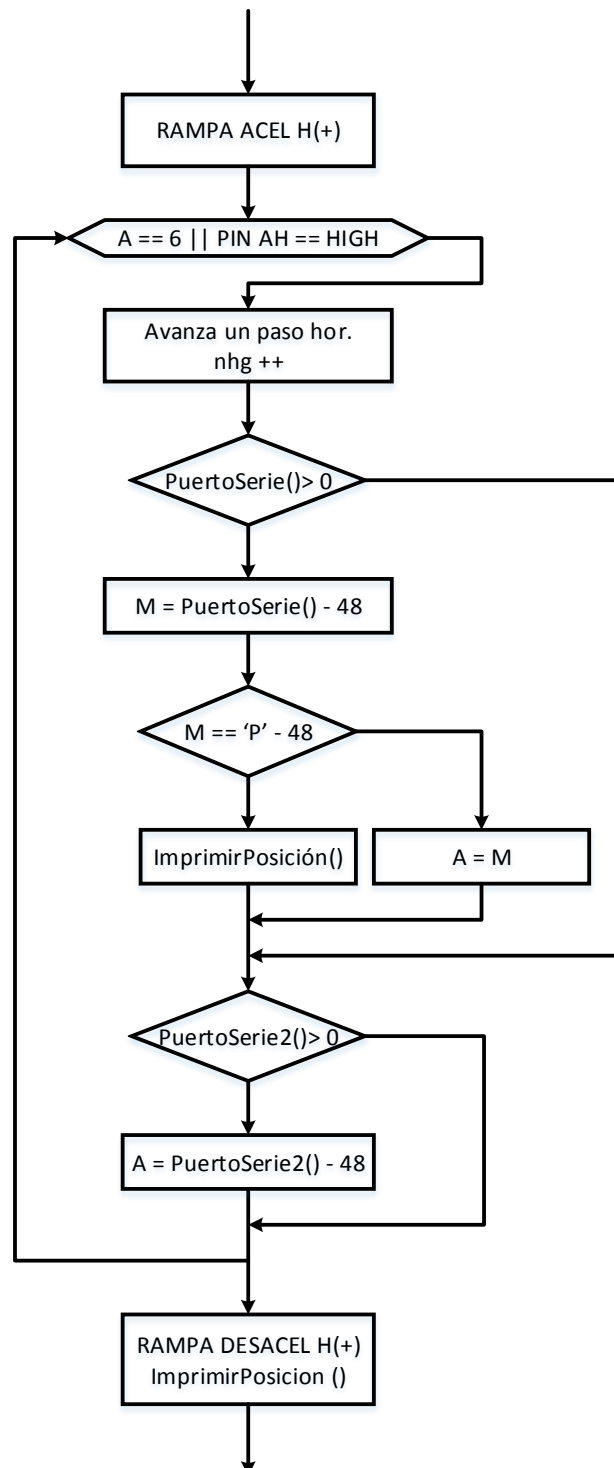


[20.2] RETROCESO
VERTICAL
SIN PCPP



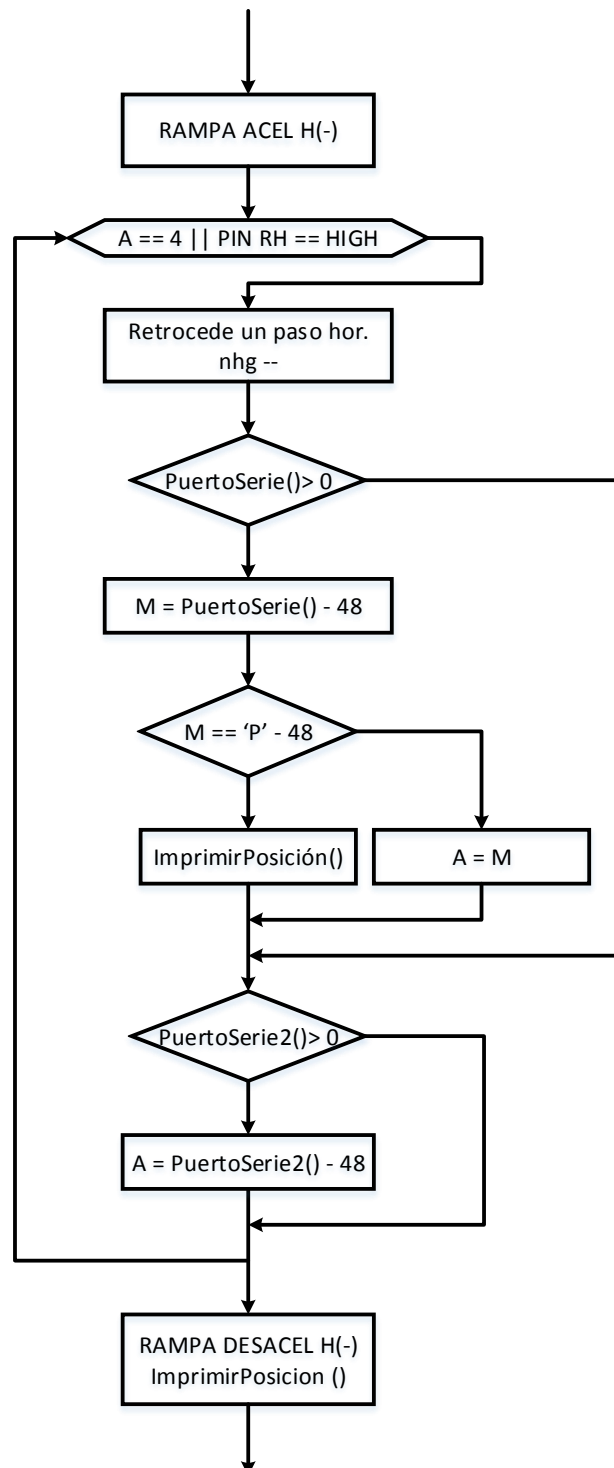


[20.3] AVANCE
HORIZONTAL
SIN PCPP



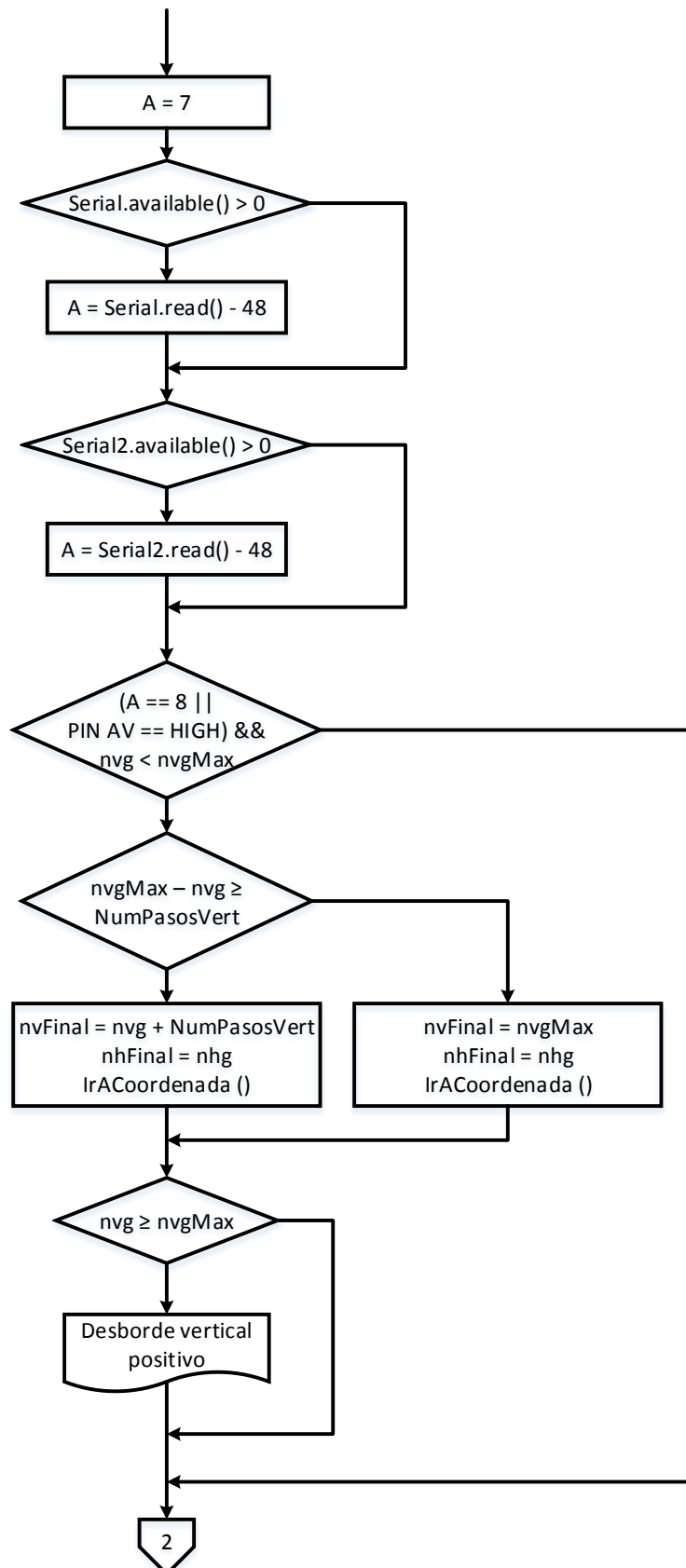


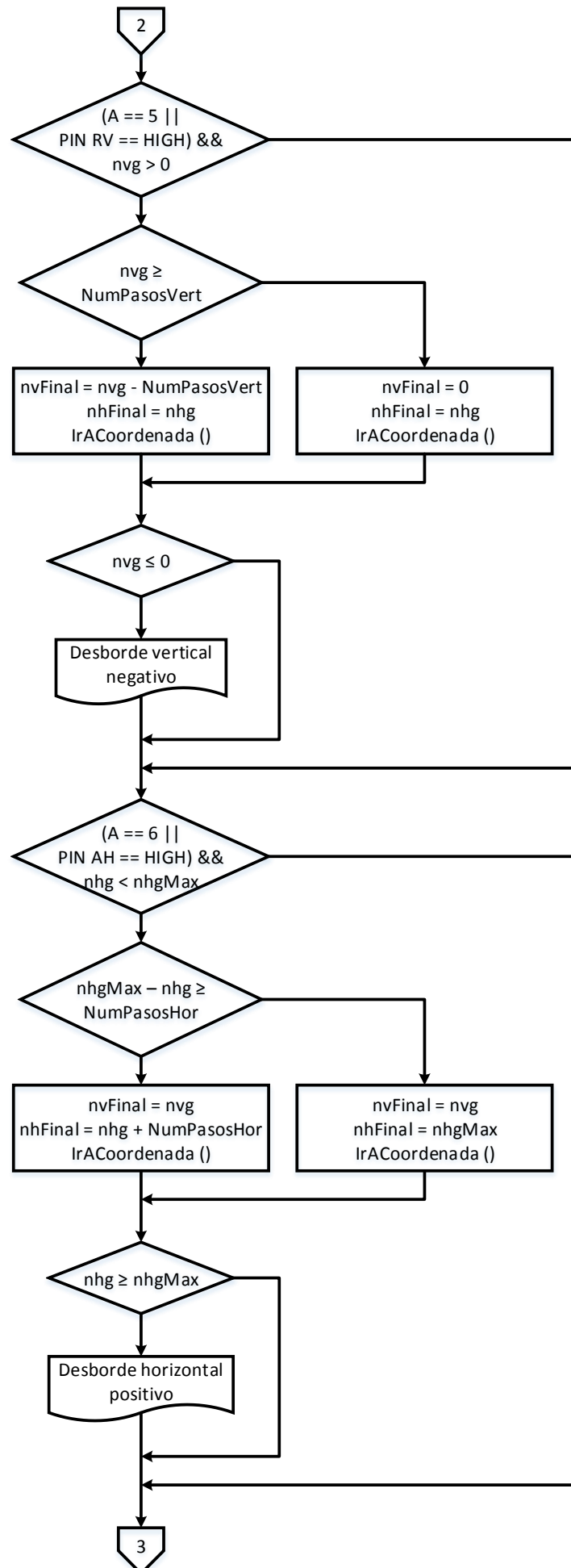
[20.4] RETROCESO
HORIZONTAL
SIN PCPP

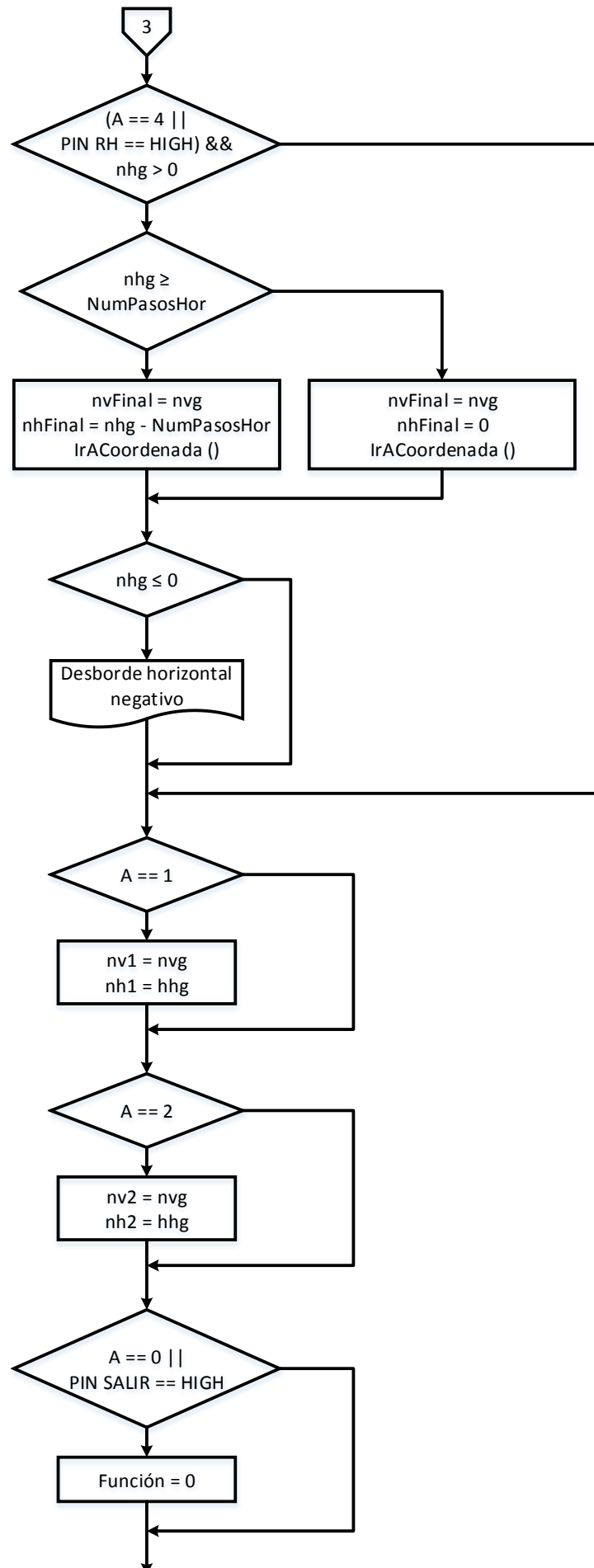




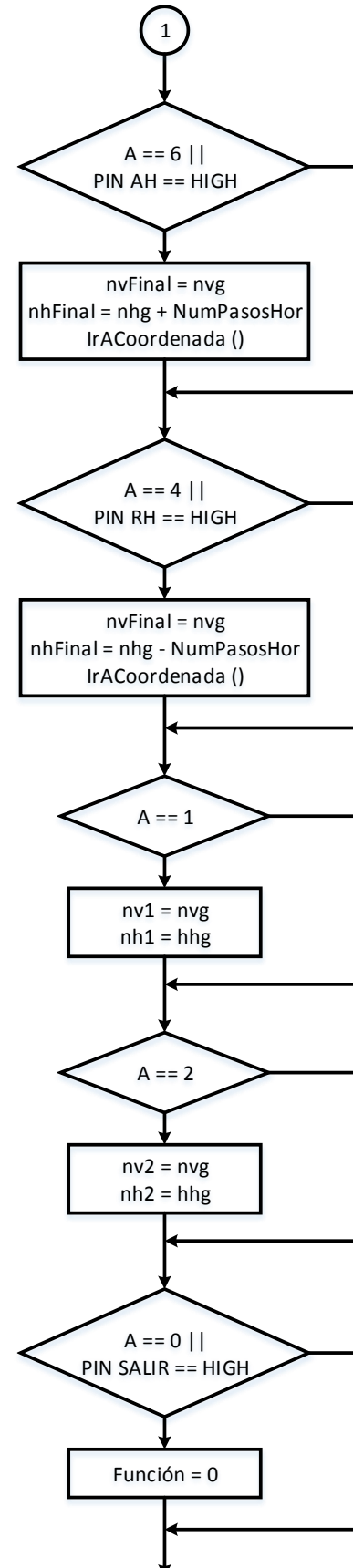
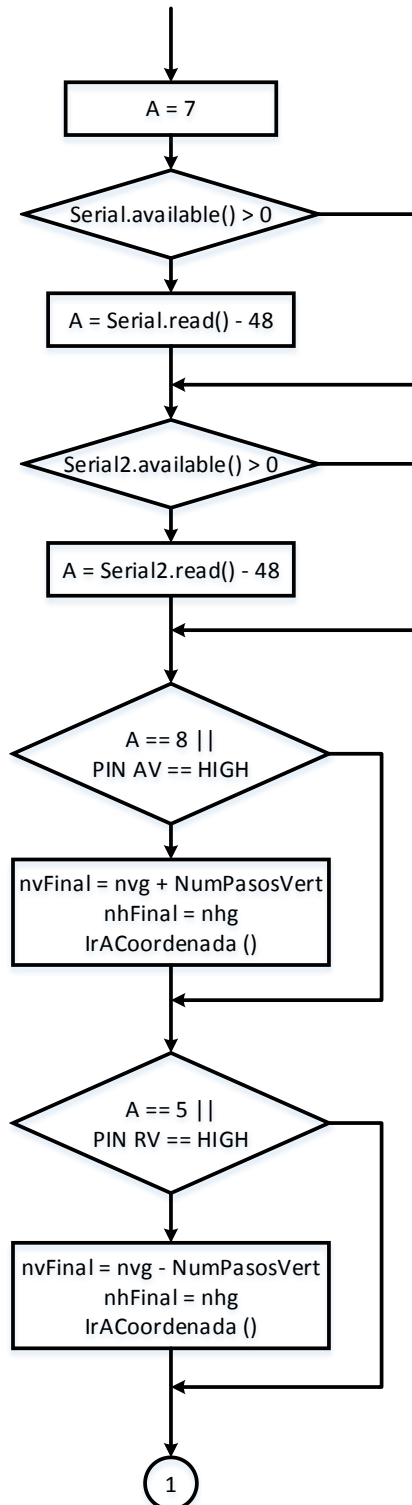
[3] MOV. LIBRE
ESCALONADO





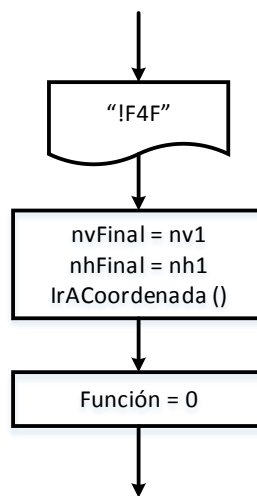


[30] MOV. LIBRE
ESCALONADO

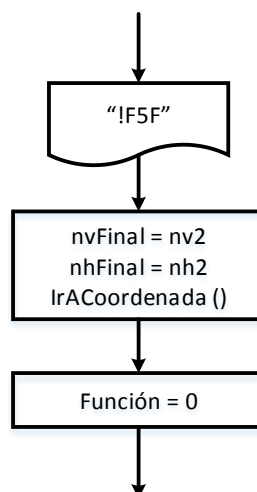




[4] IR A
MEMORIA 1

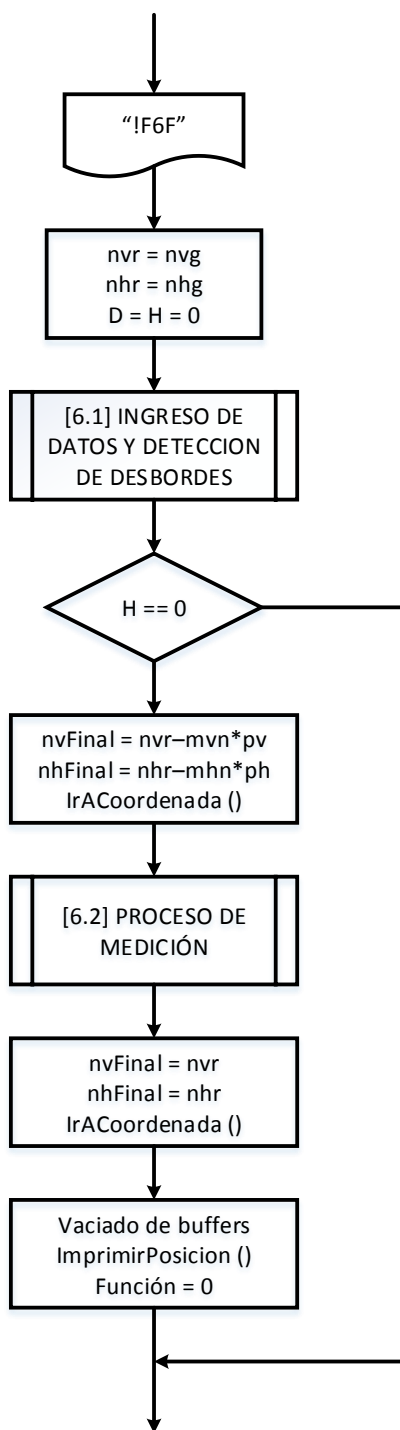


[5] IR A
MEMORIA 2

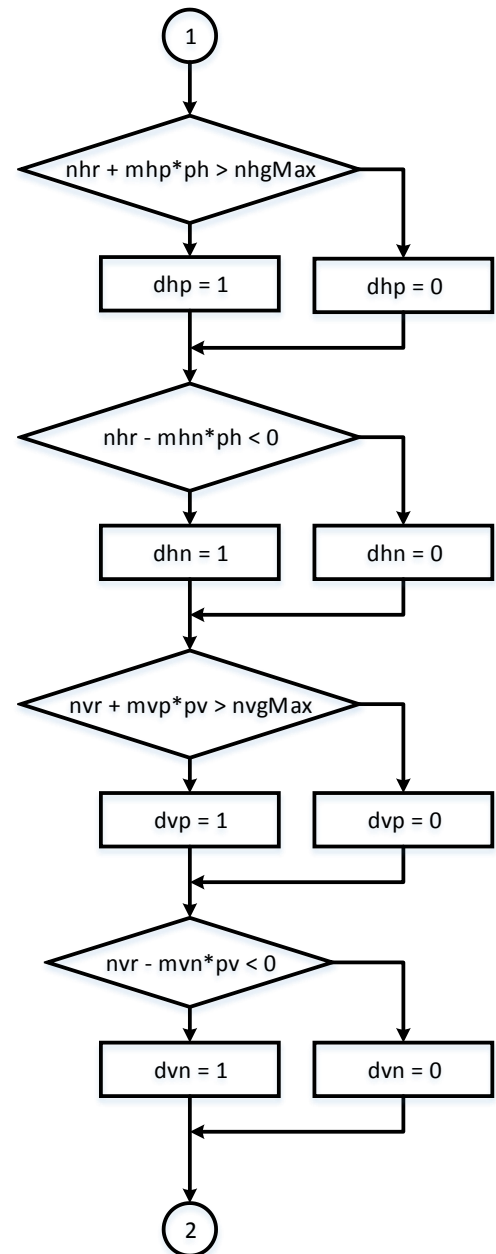
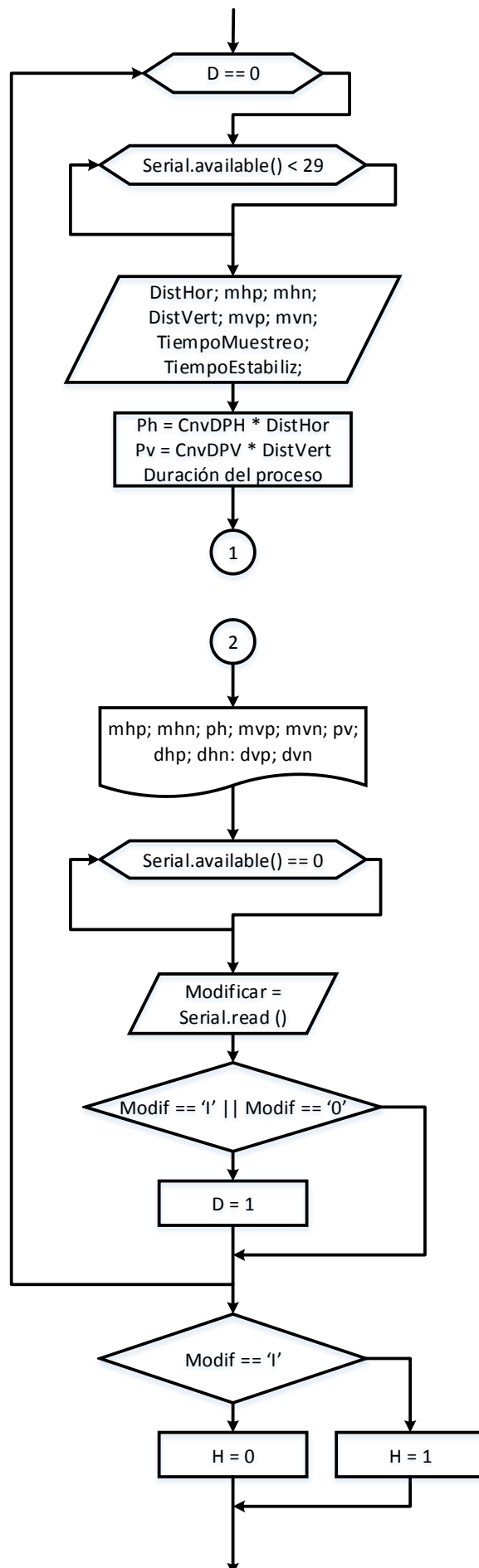




[6] MEDICION
AUTOMATICA



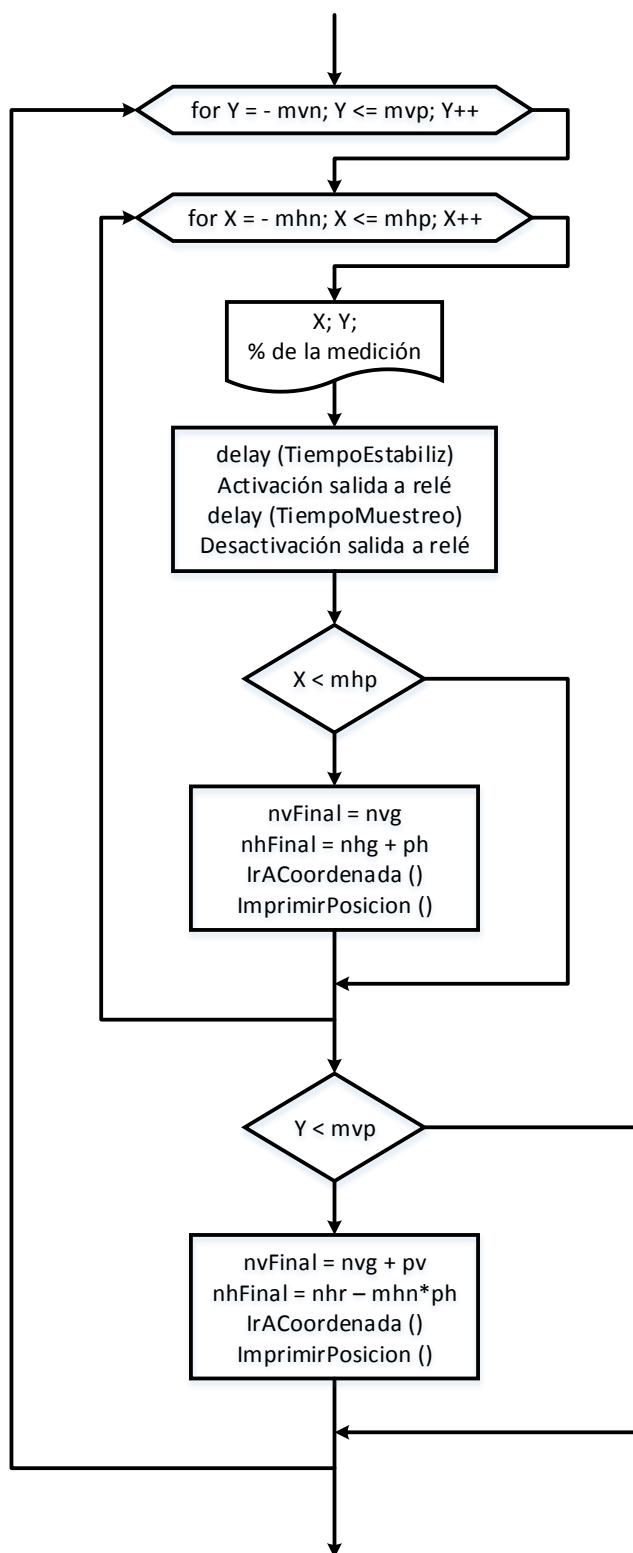
[6.1] INGRESO DE DATOS
Y DETECCIÓN DE
DESBORDES



DETECCIÓN DE DESBORDES

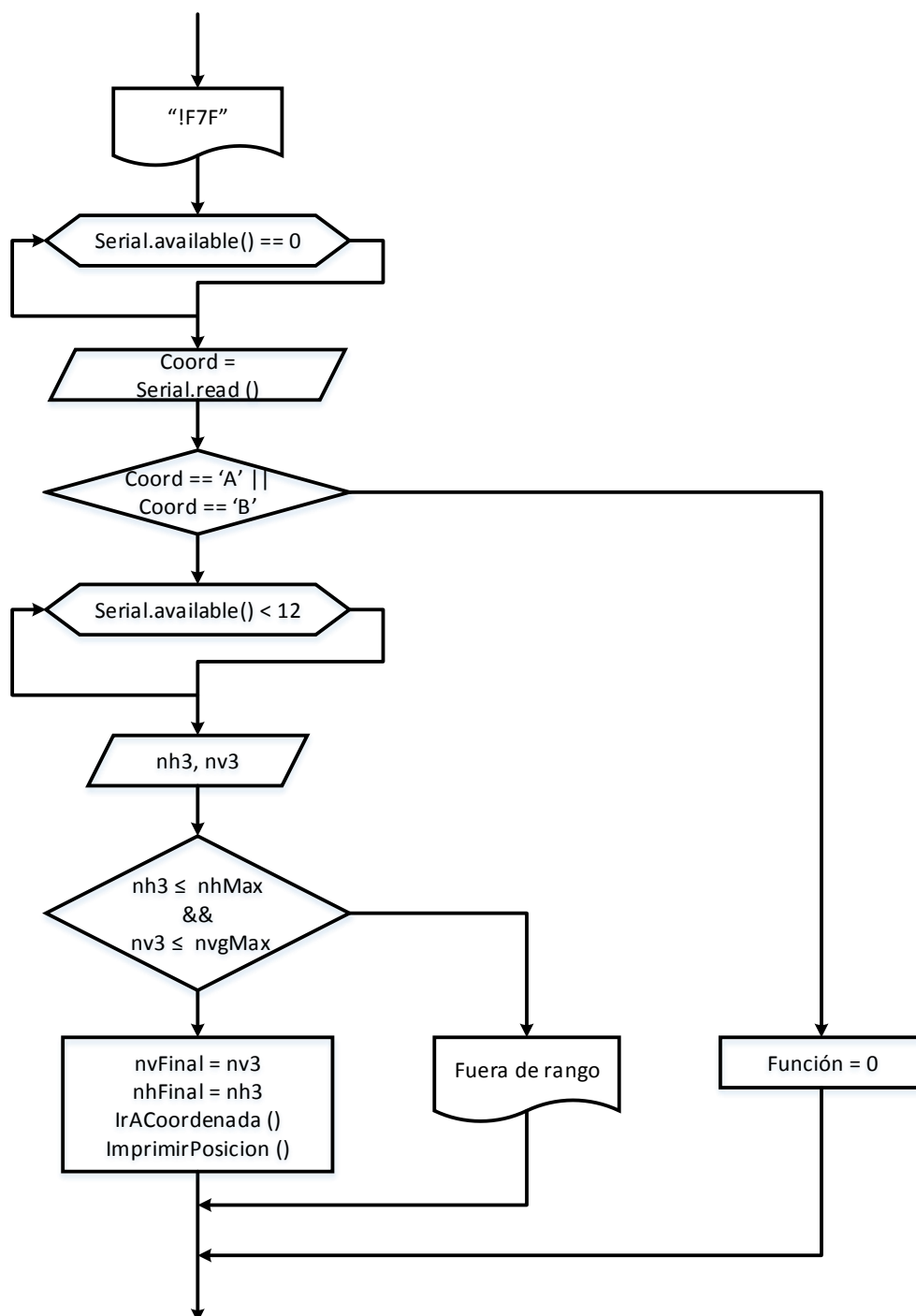


[6.2] PROCESO DE
MEDICION

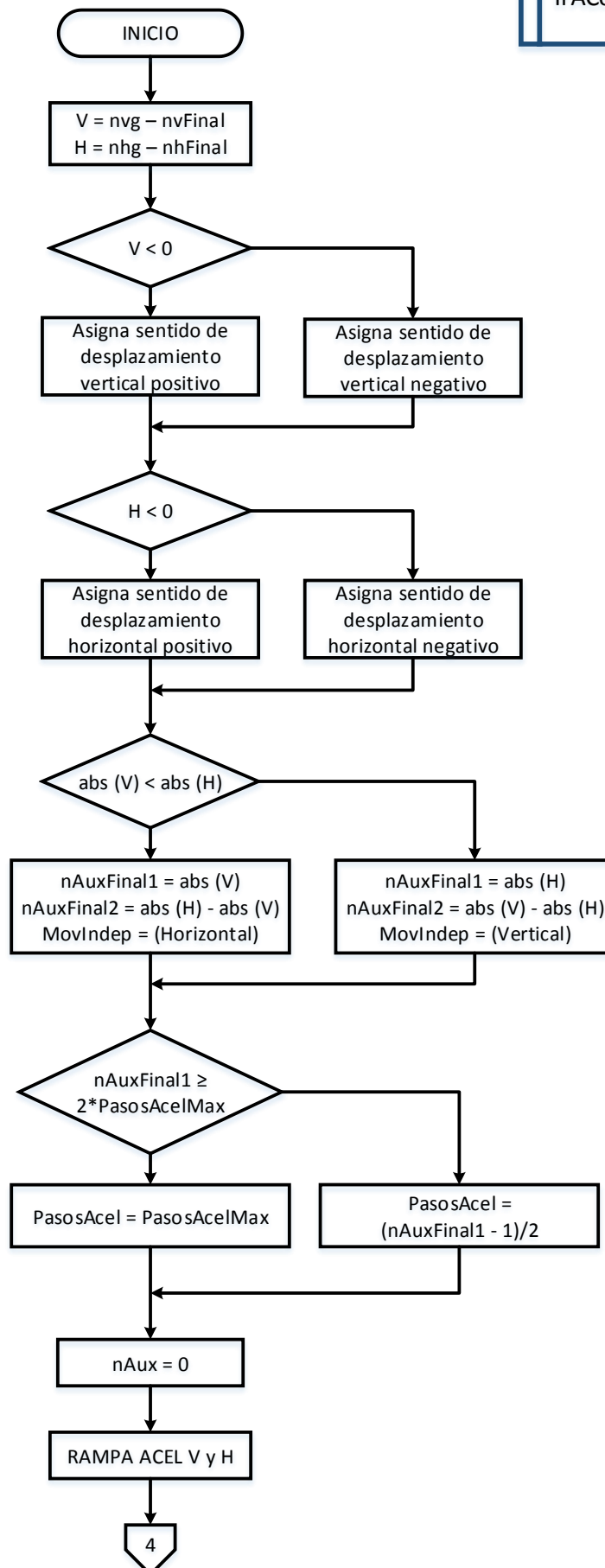


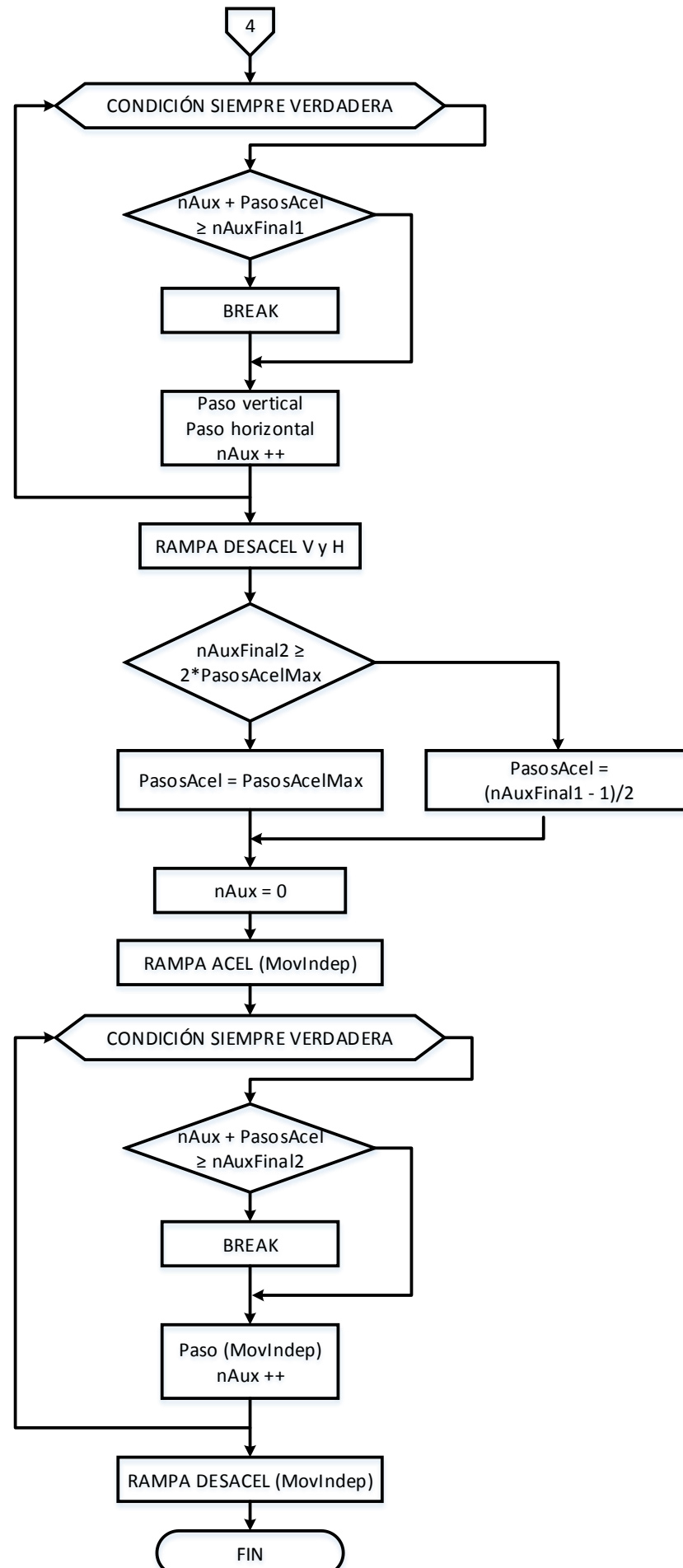


[7] IR A
COORDENADA



IrACoordenada ()







15.2 DESCRIPCIÓN DEL FUNCIONAMIENTO DE LOS PROCESOS DEL DIAGRAMA DE FLUJO RESUMIDO

▪ PROCESO PRINCIPAL

Declaración de variables de ajuste

En esta etapa se declaran variables cuyo valor numérico definitivo se deberá determinar una vez construido y montado el posicionador.

Como ya se indicó anteriormente estas variables son:

- long nvgMax: Variable utilizada en la Protección Contra Colisiones Mediante la Programación. Esta variable como se explicó es el máximo número de pasos que se podrá realizar en el sentido vertical positivo a partir del punto CERO GLOBAL de manera de evitar que el mismo colisione con la estructura.
- long nhgMax: Variable utilizada en la Protección Contra Colisiones Mediante la Programación. Esta variable como se explicó es el máximo número de pasos que se podrá realizar en el sentido horizontal positivo a partir del punto CERO GLOBAL de manera de evitar que el mismo colisione con la estructura.
- int NumPasosXMetroH: Cantidad de pasos necesarios para un desplazamiento horizontal de 1 metro.
- int NumPasosXMetroV: Cantidad de pasos necesarios para un desplazamiento vertical de 1 metro.
- float ConversorDistPasosVert: Constante para convertir una distancia en dirección vertical en número de pasos del motor. Se calcula a partir de NumPasosXMetroV.
- float ConversorDistPasosHor: Constante para convertir una distancia en dirección horizontal en número de pasos del motor. Se calcula a partir de NumPasosXMetroH.

Las dos primeras variables tomarán valores que sobrepasan los límites de las variables tipo int por ello se declaran como tipo long.

El procedimiento para la determinación de los valores de estas variables se explicó en el capítulo 13.2.

Declaración de variables fijas

Estas variables se denominan fijas ya que deberán iniciarse con un valor determinado. Por cuestión de orden estas variables al declararse se agrupan en:

- Declaración de las entradas digitales: Especifican que pin será destinado a cada entrada digital. Estas entradas digitales serán las provenientes de los pulsadores ubicados en la interfaz tipo botonera y los Sensores de Posición.
- Declaración de las salidas digitales: Especifican que pin digital será destinado a cada salida digital. Estas salidas serán las que envíen a los drivers de los motores las órdenes de avance, las que encenderán las luces leds de la interfaz tipo botonera y proveerán la Señal de Referencia que indica el momento de realizar las mediciones automáticas.



- Variables de uso global: Son las variables utilizadas en todo el sketch como por ejemplo los contadores de pasos nvg y nhg y los tiempos de espera entre pasos DelayAcel [].
- Variables propias para cada función: Estas son variables utilizadas únicamente dentro de cada función para su funcionamiento. Son principalmente banderas, variables numéricas utilizadas en comparaciones o Strings para almacenar cadenas de datos enviados desde labView. Algunas de estas variables son locales, es decir, se declaran directamente dentro de la función.

Void setup ()

Este proceso se ejecutará una única vez al encender o reiniciar el Arduino.

Void loop ()

Bucle donde se encuentra la programación de las diferentes funciones. Se ejecutará continuamente salvo cuando se invoquen las funciones ImprimirPosicion () ó IrACoordenada ().

ImprimirPosicion ()

Esta función enviará a LabVIEW una cadena de caracteres con las coordenadas del posicionador en pasos para que estas sean visualizadas en la pantalla de la aplicación. Al ser invocada desde el void loop () esta se ejecutará una única vez y se retornará al void loop () en el punto donde se la invocó.

IrACoordenada ()

Esta función lleva al posicionador hasta las coordenadas nvFinal y nhFinal previamente establecidas. Al ser invocada desde el void loop () esta se ejecutará una única vez y se retornará al void loop () en el punto donde se la invocó.

▪ SUB PROCESO void setup ()

Al comienzo del mismo se inician los puertos serie. El puerto serie 0 se utilizará para la comunicación vía USB con la aplicación de LabVIEW y el puerto serie 2 para la comunicación vía bluetooth con el App de para celular.

A continuación se establece cada pin digital utilizado como entrada o salida.

Variables de ajuste

Luego se asignará a las variables de ajuste su valor correspondiente. Para esto existen dos opciones, recibir las variables de ajuste desde la aplicación de LabVIEW, o cargar los últimos valores que fueron utilizados en estas variables los cuales se encontrarán almacenados en la memoria EEPROM del Arduino.

En caso de recibir las variables de ajuste a través de la aplicación, esta enviará las variables mediante un String de 24 caracteres de longitud. Por ello la llegada de 24 bytes al puerto serie 0 indicaría que las variables fueron enviadas desde este medio. En cambio, para asignar el valor a estas variables a través de



la memoria EEPROM, el operario debería oprimir simultáneamente en la botonera los pulsadores SALIR y MOVIMIENTO LIBRE CONTINUO.

La programación ingresará entonces en un bucle `while ()` cuya condición `Serial.available () < 24` es la presencia de menos de 24 caracteres en el buffer de entrada del puerto serie 0. Así, si desde la aplicación de LabVIEW se enviaran los valores de estas variables en el buffer de entrada de este puerto habrían 24 bytes, volviéndose falsa la condición del bucle por lo que se saldría del mismo manteniendo la bandera `Limites` su valor inicial 0.

Si estando ejecutándose el bucle anterior se oprimieran los pulsadores SALIR y MOVIMIENTO LIBRE CONTINUO, la lectura simultanea del valor HIGH en los pines de entrada asignados a estos volverían verdadera la condición de la estructura `if ()` ingresándose a la misma. Dentro de esta se asignaría a la bandera `Limites` el valor 1 y se saldría del bucle mediante la sentencia `break`.

A continuación se ingresará en la estructura `switch ()` cuya variable de control es la bandera `Limites`.

En caso de ser `Limites` igual a 0 se trabajaría con el String recibido desde LabVIEW. Este estaría compuesto por 24 caracteres, los cuales contendrían los datos de las variables de ajuste. Este String estaría conformado como se muestra en la siguiente tabla:

Bytes	Dato	Nº de bytes	Unidad	Variable asignada
[0-5]	Número máximo de pasos horizontales	6	Pasos	<code>nhgMax</code>
[6-11]	Número máximo de pasos verticales	6	Pasos	<code>nvgMax</code>
[12-17]	Numero de pasos por metro horizontal	6	Pasos	<code>NumPasosXMetroH</code>
[18-23]	Numero de pasos por metro vertical	6	Pasos	<code>NumPasosXMetroH</code>

Estando los 24 bytes en el bus de entrada estos se almacenarían en un String declarado como *char* `DatosVariablesAjuste [25]`¹. A continuación se descompondría el mismo para extraer los diferentes datos y almacenar cada uno de ellos en su respectiva variable. Para ello se utilizará un segundo string declarado como *char* `datos3 [7]` el cual iría tomando fragmentos del primer string para luego, mediante la función `atol`, convertir el string en variable numérica del tipo `long`.

Ya separado el valor de cada variable estas se almacenarían en la memoria EEPROM mediante la función `EEPROM.put`, la cual permite almacenar variables de más de un byte de extensión. A demás, esta función sobrescribe un valor almacenado solo si el nuevo valor es diferente al existente.

En el caso de ser `Limites` igual a 1 por haberse accionado los pulsadores correspondientes se trabajaría con los datos almacenados en la memoria EEPROM. Para ello se leerían las memorias mediante la función `EEPROM.get`, la cual puede leer variables de más de un byte de extensión y se asignaría a cada variable su valor correspondiente.

A continuación se calculará las variables `Hmaxmm` y `Vmaxmm`, las cuales indicarán el máximo desplazamiento en mm que puede realizarse en cada dirección. Esto se obtendrá a partir de la cantidad máxima de pasos `nhgMax` y `nvgMax` y la cantidad de pasos por metro `NumPasosXMetroH` y `NumPasosXMetroV`.

Todas estas variables recibidas más las calculadas se enviarán a la aplicación de LabVIEW para su visualización en pantalla a modo de confirmación. Por su parte el máximo desplazamiento en mm se utilizará para asignar los valores máximos del grafico de posición.

¹ Al declarar el String en su extensión debe colocarse un byte más de la cantidad de datos a almacenar para contemplar el carácter nulo que indica el final del mismo. De otra manera se producirían errores al utilizar la función `atol`.

Por ultimo las luces que indicarán la ejecución de las funciones en la botonera se encenderán en secuencia y luego se apagarán indicando que las variables de ajuste fueron cargadas correctamente. Esto se realizará para que el operario, en caso de estar trabajando sin la aplicación de labVIEW, sepa que el posicionador ya se encuentra listo para utilizarse.

- SUB PROCESO void loop ()

Este subproceso es un bucle, por lo cual, una vez ingresado al mismo, este se ejecutará continuamente salvo que sea invocada la función `ImprimirPosicion ()` o la función `IrACoordenada ()` como se indicó anteriormente.

Dentro del `void loop ()` se ingresará en primer lugar al sub proceso SELECCIÓN DE FUNCIÓN, en el cual se permanecerá indefinidamente hasta que el usuario indique la función que desea ejecutar. En ese momento se asignará a la variable `Funcion` un valor entre 1 y 7 según cual sea la función deseada y allí finalizará este sub proceso continuándose con la ejecución del bucle. Según el valor asignado a esta variable se ingresará luego a la estructura `if ()` o al bucle `while ()` que contiene el sub proceso con la función correspondiente. Cuando la función ejecutada haya finalizado o se desee finalizarla, dentro de ella se asignará valor 0 a la variable `funcion`, con lo cual ningún otro sub proceso del bucle se ejecutará llegando así al final del mismo e iniciándose luego un nuevo ciclo. En este nuevo ciclo se ingresará nuevamente al sub proceso SELECCIÓN DE FUNCIÓN con lo cual el todo el proceso se repetirá. De esta manera se ejecutará un ciclo del bucle `void loop ()` por cada orden recibida para ejecutar una función.

En cuanto a la asignación del valor 0 a la variable `Funcion` al finalizar una función o cuando se desee finalizarla, la misma dependerá de si el sub proceso que contiene a la función se encuentra dentro de una estructura `if ()` o dentro de un bucle `while ()`. En el caso de estar contenida dentro de una estructura `if ()`, esta finalizará de forma automática, por lo que asignación del valor 0 a esta variable será también automática. En cambio si se ejecutara dentro de un bucle `while ()`, recién cuando el usuario desee finalizarla se asignará el valor 0 a esta variable, dejándose entonces de cumplirse la condición del bucle y saliendo del mismo.

Al final del bucle `void loop ()`, luego de las estructuras `if ()` y los bucles `while ()` que contienen las funciones, se encuentra una estructura `if ()` cuya variable de control `CeroDeterminado` es una bandera que tendrá valor cero en tanto no se ejecute la función `DETERMINAR CERO` y tomará valor 1 luego de ejecutarse. Así, según se haya o no ejecutado la función, se enviarán por puerto serie hacia las interfaces de LabVIEW y la APP los caracteres correspondientes. Estos caracteres indicarán a las interfaces que la función que se encontraba en ejecución ha finalizado, con lo cual estas desbloquearán los botones correspondientes que inician las funciones. Según la función `DETERMINAR CERO` se haya ejecutado o no los botones a desbloquearse variarán. Así por ej., la cadena `!F8F`, la cual se enviará mientras la función no se haya ejecutado, indicará a la aplicación de LabVIEW que deben desbloquearse todas las funciones excepto `IR A COORDENADA` y `MEDICIÓN AUTOMÁTICA`. En cambio `¡F0F` indicará que deben desbloquearse todas las funciones excepto `DETERMINAR CERO`, debido que esta ya fue ejecutada.

SELECCIÓN DE FUNCIÓN

Aquí se recibirán desde las diferentes interfaces las órdenes que indicarán cual función desea ejecutar el usuario para luego asignarle a la variable numérica `Funcion` el valor correspondiente. Así, según el valor que posea esta variable se ingresará luego a la estructura de control `if ()` o al bucle `while ()` que contiene el sub proceso con la función correspondiente. Esta orden podrá llegar a través de los puertos serie o de la botonera. Los valores que tomará la variable `Funcion` según la función que se deba ejecutar y el/los sub procesos que los contienen serán los siguientes:



Variable Funcion	Función		Sub proceso en el que se ejecutará
1	DETERMINAR CERO	-	[1]
2	MOV. LIBRE CONTINUO	Con PCCP	[2]
		Sin PCCP	[20]
3	MOV. LIBRE ESCALONADO	Con PCCP	[3]
		Sin PCCP	[30]
4	IR A MEMORIA 1	-	[4]
5	IR A MEMORIA 2	-	[5]
6	MEDICIÓN AUTOMÁTICA	-	[6]
7	IR A COORDENADA	-	[7]
0	SALIR	-	-

En el caso de las funciones 2 y 3 estas se ejecutarán con o sin Protección Contra Colisiones Mediante la Programación (PCCP) dependiendo de si se está trabajando con el Sistema de Referencias Global (SRG) o el Sistema de Referencias Inicial (SRI) respectivamente, o dicho de otra manera, si previamente fue ejecutada o no la función DETERMINAR CERO. Para ello se utilizará nuevamente la bandera **CeroDeterminado**.

Como ya se explicó, la variable **Funcion** en realidad no tomará valor 0 en esta sub función, sino que lo hará dentro de los sub procesos.

[1] DETERMINAR CERO

Este sub proceso contiene a la función DETERMINAR CERO, la cual buscará el Punto de Referencia Físico y a partir de este determinará el punto Cero Global. Este sub proceso se coloca dentro de una estructura **If ()** ya que se ejecutará de principio a fin de forma automática, por lo que finalizará sin intervención del operario.

Si al ejecutarse esta función el Cero Global es determinado exitosamente la bandera **CeroRealizado** tomará valor 1 registrando que la función fue ejecutada sin inconvenientes. En cambio si se detecta problemas como por ej una obstrucción en alguno de los sensores, la bandera continuara con valor 0 indicando que no pudo determinarse el punto Cero Global. Así, el valor de la bandera se utilizará luego para iniciar las funciones MOVIMIENTO LIBRE con o sin PCCP y en la estructura **if ()** al final del bucle **void loop ()** para desbloquear o no el uso de las funciones MEDICIÓN AUTOMÁTICA e IR A COORDENADA.

[2] MOVIMIENTO LIBRE CONTINUO (Con PCCP)

Este sub proceso contiene la programación de la función MOVIMIENTO LIBRE CONTINUO cuando esta se ejecuta con PCCP. Es por esto que en la condición del bucle **while ()**, además de ser la variable **Funcion** igual a 2, la bandera **CeroRealizado** deberá tener valor 1.

En este caso es el usuario el que indicará el momento que desea finalizar la función, por ello a la misma se la ubica dentro de un bucle **while ()**. Así, cuando el usuario indique que desea finalizar la función se asignará valor 0 a la variable **Funcion** y la programación saldrá del buque **while ()** debido a que ya que no se verificará la condición de permanencia en el mismo.



En este sub proceso, por ejecutarse la función con PCCP, las variables *nvg* y *nhg* serán monitoreadas antes de realizar cada paso de los motores, detectando así si se alcanzaría alguno de los límites de la zona de trabajo.

[20] MOVIMIENTO LIBRE CONTINUO (Sin PCCP)

Sub proceso en el cual se ejecutará la función MOVIMIENTO LIBRE CONTINUO cuando esta se ejecute sin la PCCP. En este caso en la condición del bucle `while ()` además de ser la variable *Funcion* igual a 2, la bandera *CeroRealizado* deberá tener valor 0, indicando así que la función DETERMINAR CERO aun no fue ejecutada.

Al igual que el sub proceso anterior, cuando el usuario desee finalizar la función se asignará valor 0 a la variable *Funcion* para así no verificar la condición de permanencia en el bucle.

Al no ejecutarse en esta función la PCCP las variables *nvg* y *nhg*, si bien serán actualizadas luego de cada paso, sus valores no serán monitoreados.

[3] MOVIMIENTO LIBRE ESCALONADO (Con PCCP)

Este sub proceso contiene la función MOVIMIENTO LIBRE ESCALONADO cuando esta se ejecuta con PCCP. Se ingresará al mismo cuando *Funcion* toma el valor 3 y la bandera *CeroRealizado* posea valor 1. Cuando se desee finalizar la mismo a *Funcion* se le asignará valor 0 para salir del bucle `while ()`. Al igual que el sub proceso [2] las variables *nvg* y *nhg* se monitorearán antes de realizar cada paso determinando así si se ha llegado o no a alguno de los limites.

[30] MOVIMIENTO LIBRE ESCALONADO (Sin PCCP)

Sub proceso idéntico al [3] pero en el cual no se ejecutará la PCCP. Al mismo se ingresará y permanecerá mientras la variable *Funcion* posea valor 3 y la bandera *CeroRealizado* valor 0. Al igual que en el sub proceso [20], las variables *nvg* y *nhg* serán actualizadas pero no monitoreadas.

[4] IR A MEMORIA 1

Sub proceso que contiene a la función IR A MEMORIA 1, el cual llevará al posicionador a las coordenadas almacenadas en la memoria 1. Se ubica dentro de una estructura `if ()` ya que la misma finalizará luego de alcanzadas las coordenadas memorizadas y sin intervención del usuario.

[5] IR A MEMORIA 2

Sub proceso cuyo funcionamiento idéntico al [4] pero trabajando con los valores de coordenadas guardadas en la memoria 2.

[6] MEDICIÓN AUTOMÁTICA

Este sub proceso que contiene la programación de la función MEDICIÓN AUTOMÁTICA. Si bien el proceso de ingreso de valores se podrá realizar repetidas veces, una vez confirmado estos valores y realizadas las mediciones la función finalizará. Es por esto que el sub proceso se ubica dentro de una



estructura if (). En caso de que se haya ingresado a la función y se desee cancelarla durante su ejecución, la programación saltará partes del proceso llegando así anticipadamente al final.

[7] IR A COORDENADAS

En este sub proceso se encuentra la programación de la función IR A COORDENADAS. En el mismo el usuario indicará el momento de finalizar la función, por ello esta se ubica dentro de un bucle while () al igual que las funciones de movimiento libre.

▪ SELECCIÓN DE FUNCIÓN

Valor de la variable Funcion

En primer lugar este sub proceso se encargará de dar a la variable Funcion el valor que corresponde a la función que el usuario desee ejecutar. Así, según el valor de esta variable se ingresará en el void loop () a la estructura if () o al bucle while () correspondiente como ya se explicó.

La orden de ejecutar una función podrá ser ingresada mediante cualquiera de las interfaces. En caso de utilizar la aplicación de LabVIEW o la app la orden llegará en forma de carácter a través del puerto serie correspondiente y en caso de utilizar la botonera mediante un HIGH en la entrada digital asignada a la misma. Los caracteres recibidos por los puertos serie, sus correspondientes valores de código ASCII y los pines digitales asignados a cada función se indican a continuación junto con el valor que tomará en cada caso la variable Funcion:

	Puerto Serie 0 (Aplicación LabVIEW)		Puerto Serie 2 (App)		Pin digital	Variable Funcion
	Carácter	Cod. ASCII	Carácter	Cod. ASCII		
DETERMINAR CERO	'1'	49	'1'	49	32	1
MOV. LIBRE CONTINUO	'2'	50	'2'	50	34	2
MOV. LIBRE ESCALONADO	'3'	51	'3'	51	36	3
IR A MEMORIA 1	'4'	52	-	-	38	4
IR A MEMORIA 2	'5'	53	-	-	40	5
MEDICIÓN AUTOMÁTICA	'6'	54	-	-	-	6
IR A COORDENADA	'7'	55	-	-	-	7
SALIR	'0'	48	'0'	48	30	0

(-) Indica que la función no podrá ser ejecutada desde esa interfaz.

En este sub proceso se ingresará primeramente en un bucle while () de espera de función, denominada así porque la programación se mantendrá indefinidamente dentro de este bucle hasta que el usuario indique la función que desea ejecutar. Para ello, la condición del bucle será siempre verdadera, por lo que únicamente podrá salirse del mismo mediante las sentencias break, las cuales solo se ejecutarán cuando sea recibida la orden por parte del usuario.

Dentro de este bucle while () se encuentran 7 estructuras if (). Las dos primeras recibirán las órdenes enviadas a través de los puertos serie y las restantes 5 las órdenes ingresadas desde la botonera.



La primera estructura tiene como condición `Serial.available() > 0`, lo cual significa la existencia de al menos un byte en el buffer de entrada del puerto serie 0. Este byte será entonces el carácter enviado desde la aplicación de LabVIEW que indicará la función que se desea ejecutar. En caso de recibir este carácter la condición de la estructura será verdadera por lo que se ingresará a la misma. Dentro de esta se almacenará el valor recibido en la variable `Funcion` y a continuación se ejecutará la sentencia `break` que hará salirse del bucle `while()` de espera de función.

En este caso, por el puerto serie en realidad se recibirán los valores en código ASCII de los caracteres asignado a cada función, por ello a estos valores deberá restársele el número 48 (valor ASCII del número 0) para así poder obtener el número del 1 al 7 correspondiente a la función deseada.

La segunda estructura `if()` funcionará de manera idéntica a la primera, solo que en este caso se trabajará con el puerto serie 2 donde se recibirán las ordenes enviadas desde la App para celular.

Las restantes 5 estructuras `if()` tienen como condición que el PIN X se encuentre en estado HIGH, lo cual significa el accionamiento de uno de los 5 pulsadores asignados para iniciar funciones, estando estos conectados en configuración PULL DOWN. Así, cuando uno de estos pulsadores sea accionado se ingresará en la correspondiente estructura `if()` en la cual se asignará el debido valor a la variable `Funcion` y se ejecutará luego la sentencia `break` para salir del bucle.

Valores iniciales para las funciones 2 y 3

A continuación, si la función deseada fuera la MOV LIBRE CONT o la MOV LIBRE ESCAL, se ingresará a la última estructura `if()` de esta sub función. En este punto se enviará información a las interfaces de LabVIEW y la APP indicando cual función se estará por ejecutar, de este modo en cada interfaz se bloqueará o desbloqueará los botones correspondientes.

Este envío de información se realizará mediante el envío de la cadena “¡F2F” y el carácter ‘2’ para el caso de la función MOV LIBRE CONTINUO y la cadena “¡F3F” y el carácter ‘3’ para la función MOV LIBRE ESCALONADO. En las demás funciones esto se realizará recién al ingresar en el sub proceso que la contiene, en cambio, en estos casos se realizará en este punto ya que los sub procesos que contienen a estas funciones se ubican dentro de bucles `while()`, por lo que si estas acciones se realizaran allí la información se enviaría repetidas veces.

Se enviará también en este punto, en caso de haberse ya determinado la ubicación del Cero Global, información sobre la existencia o no de desbordes en la posición actual del posicionador, y se dará valor inicial a la variable `A` propia de estas funciones.

▪ [1] DETERMINAR CERO

Ubicación del Punto de Referencia Físico

La condición de ingreso a la estructura `if()` para la ejecución de esta función se realizará con la condición `Funcion == 1`. Al ejecutarse, en primer lugar se enviará la cadena “¡F1F” y el carácter ‘1’ a las interfaces de LabVIEW y la App respectivamente, indicándoles así que esta función se encuentra en ejecución.

A continuación se dará valor 0 a las banderas `V1` y `H1`. Estas banderas posteriormente mantendrán este valor si la función se ejecutó correctamente y se pudo determinar el punto CERO GLOBAL, y tomarán valor 1 cuando se considere que alguno de los sensores de posición se encuentre bloqueado o con problemas y por ello el CERO GLOBAL no se haya podido determinar.



Se ingresará entonces en el bucle `while ()` encargado de localizar el sensor de posición horizontal. La condición de este bucle será siempre verdadera, y por ello solo podrá salirse del mismo ejecutando una sentencia `break` dentro de este bucle. Aquí en primer lugar se verificará el estado del sensor de posición horizontal, si la entrada digital asignada a este sensor se encuentra en HIGH indicaría que el sensor se encuentra libre (sensor PNP en modo luz y configuración Pull Down como se explica en el capítulo 23.2) y por lo tanto puede iniciarse el proceso de retroceso horizontal. Se realizará la rampa de aceleración en dirección negativa y se volverá a verificar el estado del sensor, si el mismo sigue libre (entrada digital en HIGH) continuará el proceso de retroceso. Aquí se ingresará a otro bucle `while ()` cuya condición de permanencia es que el sensor siga libre, por ello, el posicionador retrocedería de forma permanente hasta obstruir el sensor, momento en el que se saldría del bucle, se iniciaría la rampa de desaceleración y al concluir esta se ejecutaría la sentencia para finalizar el bucle `while ()` de localización del sensor de posición horizontal. Durante este proceso la variable H1 mantendría su valor inicial 0, indicando que la posición horizontal del CERO GLOBAL habría sido determinada correctamente y el posicionador se encontraría en dicho punto.

Por otra parte, si tanto en las estructuras `if ()` anterior y posterior a la rampa de aceleración negativa se detectara que el sensor estuviera obstruido (entrada digital en LOW), no se realizaría el retroceso continuo, sino que el posicionador se detendría y posteriormente avanzaría en dirección horizontal positiva hasta liberar el sensor. Para ello en posicionador aceleraría en dirección positiva y se ingresaría al bucle `while ()` cuya condición de permanencia es que el sensor siga bloqueado, por ello, el posicionador avanzará de forma permanente hasta liberar el sensor, momento en el que se saldría del bucle, se detendría el avance y se ejecutaría nuevamente todo el ciclo `while ()` de localización del sensor de posición.

En este punto, puede darse el caso que el sensor se encuentre obstruido o exista algún inconveniente en el mismo o en el circuito de alimentación, esto haría que la entrada digital asignada a este sensor se encuentre siempre en estado LOW, por lo que el posicionador podría avanzar indefinidamente en dirección horizontal. Para evitar ello se pone un límite a la cantidad de pasos que se puede avanzar en dirección positiva, y en caso de superar este límite se considerará que existe algún problema técnico en el circuito y se detendrá el avance. Para esto se utilizará el contador auxiliar `nAux`, el cual se incrementará una unidad por cada paso avanzado en dirección positiva. Al alcanzar `nAux` el valor `NPBS` (Num. pasos bloqueo de sensor) previamente establecido (`NPBS = 5700` equivalente a un avance de aprox. 25cm en dirección horizontal), se dará valor 1 a la bandera H1 y mediante una sentencia `break` se saldrá del bucle `while ()` de avance positivo. A continuación, en la sentencia `if ()` al detectar que la bandera H1 ha tomado valor 1 se dará un mensaje de advertencia indicando el problema y mediante otra sentencia `break` se finalizará el bucle `while ()` de localización del sensor de posición horizontal.

Se ingresará luego al bucle `while ()` de localización del sensor de posición vertical, cuyo funcionamiento es idéntico al horizontal pero usando en este caso la bandera V1 para indicar si la posición vertical del CERO GLOBAL fue determinada correctamente o no.

A continuación, mediante una estructura `if ()` se analizará el valor de las banderas V1 y H1. En caso de que ambas banderas permanezcan en valor 0 se habría hallado exitosamente el punto CERO GLOBAL, por lo que se daría a la bandera `CeroRealizado` el valor 1 (la misma tendrá valor 0 desde que es declarada al inicio de la programación). En cambio, si al menos una de las banderas tuviera valor 1, la condición de la estructura `if ()` sería falsa por lo que se finalizaría la función manteniendo la bandera `CeroRealizado` el valor 0, e indicando así que el origen del sistema de referencia global sigue sin ser determinado.

Definición del punto CERO GLOBAL

Si al finalizar los bucles de localización de los sensores de posición las banderas V1 y H1 mantuvieran los valores 0, el posicionador se encontraría en ese momento en el punto CERO GLOBAL, origen del SRG. Luego, simplemente se deberá dar valor 0 a los contadores `nvg` y `nhg` como se indicó en el capítulo 11.4.



Este último paso no se realizará en este momento si no recién al finalizar la actualización de coordenadas de las memorias.

Actualización de coordenadas de las memorias

Antes de asignar valor 0 a los contadores *nvg* y *nhg* deberá actualizarse las posiciones guardadas en las memorias, ya que para esto se requerirán los valores actuales de estos contadores. Actualizar las posiciones de las memorias implicará determinar los nuevos valores de las variables *nh1*, *nv1*, *nh2* y *nv2*, ya que estas variables estarían almacenando las coordenadas de las posiciones guardadas referidas SRI, por lo cual deberán convertirse para que estén referidas al SRG.

Cada memoria en tanto no haya sido utilizada almacenará la posición del origen del sistema de referencias en uso, es decir, almacenará la coordenada (0;0). Si alguna de estas memorias fuera utilizada para guardar una posición estando trabajando con el SRI, esta posición se mantendrá invariable luego de pasar a trabajar con el SRG. Para ello la programación convertirá las coordenadas utilizadas de un sistema a otro. Esto requerirá conocer la posición relativa entre los centros de ambos sistemas de referencias, lo que se consigue mediante los contadores de pasos *nhg* y *nvg*, los cuales irán actualizando su valor durante la ejecución de esta función. Es decir, los valores actuales de estos contadores serán justamente las coordenadas del centro del SRG respecto al SRI. Luego, la actualización de coordenadas de las memorias al pasar del SRI al SRG se obtendrá, para el caso de la memoria 1, mediante las expresiones:

$$nh1 = nh1 - nhg$$

$$nv1 = nv1 - nvg$$

Y para el caso de la memoria 2 mediante:

$$nh2 = nh2 - nhg$$

$$nv2 = nv2 - nvg$$

Para saber si una memoria fue utilizada o no y así convertir las coordenadas almacenadas en ellas mediante las expresiones anteriores o mantener el valor (0;0), se utilizarán las banderas *Mem1Utilizada* y *Mem2Utilizada*. Estas banderas se declaran con valor 0 y tomarán valor 1 cuando la memoria correspondiente sea utilizada para almacenar una posición dentro de alguna función de MOVIMIENTO LIBRE. Por ello mediante dos estructuras *if ()* (no mostradas en el diagrama de flujo), una para cada memoria, según cual sea el valor de estas banderas, se realizará o no la conversión de coordenadas de las memorias.

En caso de que al convertir las coordenadas de alguna memoria del SRI al SRG, las mismas se encuentren fuera de los límites de la zona trabajo, esta memoria se eliminará y en su lugar se almacenará las coordenadas (0 ; 0).

Finalmente se dará valor 0 a las variables *nvg* y *nhg* convirtiendo el punto actual del posicionador en el CERO GLOBAL, origen de coordenadas del SRG.

- [2] MOV. LIBRE CONTINUO



Esta función se ubica dentro de un bucle `while ()` definido por las condiciones `Funcion == 2` y `CeroRealizado == 1`, por lo tanto la misma se ejecutará continuamente hasta que esta condición resulte falsa. Como ya se explicó, la condición de salida será dando a `Funcion` el valor 0.

Al inicio del sub proceso se ubican dos estructuras `if ()` las cuales recibirán los datos provenientes de los puertos serie y los almacenarán en la variable A. La primer estructura recibirá los datos provenientes de labVIEW y la segunda los de la App.

Luego, las ordenes ingresadas a través de las diferentes interfaces se transformarán en acciones mediante las siguientes estructuras `if ()`. Estas se ejecutarán según el valor que tome A, o cual pin digital sea activado. Las cuatro primeras realizarán los desplazamientos del posicionador en cada dirección, las dos siguientes son utilizadas para almacenar la posición actual en las memorias y la última para realizar la condición de salida (dar a `Función` valor 0).

En esta función el valor `A = 7` se considera un valor de reposo, ya que con este valor en la variable se permanecerá dentro de la función, pero al no ingresar a ninguna de las estructuras `if ()` no se realizaría ninguna acción.

Ingreso de órdenes

La acción a realizarse podrá ser ingresada mediante cualquiera de las interfaces. Los caracteres recibidos por puerto serie, su correspondiente valor en código ASCII, el valor que tomará la variable A en cada caso y los pines digitales asignados a cada acción se indican a continuación:

	Puerto Serie 0 (Aplicación LabVIEW)		Puerto Serie 2 (App)		Variable A	Pin digital
	Carácter	ASCII	Caracter	ASCII		
SALIR	'0'	48	'0'	48	0	30
MEMORIA 1	'1'	49	'1'	49	1	50
MEMORIA 2	'2'	50	'2'	50	2	52
AVANCE VERT. POSITIVO	'8'	56	'8'	56	8	42
AVANCE VERT. NEGATIVO	'5'	53	'5'	53	5	46
AVANCE HOR. POSITIVO	'6'	54	'6'	54	6	48
AVANCE HOR. NEGATIVO	'4'	52	'4'	52	4	44
REPOSO	'7'	55	'7'	55	7	-

Desplazamientos

Los desplazamientos verticales positivos y negativos y los horizontales positivos y negativos, se realizarán dándole los valores 8, 5, 6 y 4 respectivamente a la variable A, o bien, accionando los pulsadores correspondientes a cada desplazamiento. De esta forma se ingresaría a la estructura `if ()` correspondiente, la cual contendrá el sub proceso encargado del desplazamiento correspondiente.

La función MOVIMIENTO LIBRE CONTINUO se ejecutará aquí con protección contra colisiones mediante programación, por lo que, para ingresar a alguna de las estructuras `if ()` que contiene los subprocesos de desplazamiento, además de verificarse el valor de A o del accionamiento de los



pulsadores, se deberá verificar también que los contadores `nhg` y `nvg` no hayan alcanzado los valores mínimos "0" o máximos "`nhgMax`" y "`nvgMax`" según corresponda.

Una vez ingresado a estos sub procesos se ejecutará el avance correspondiente y el mismo finalizará cuando el usuario así lo desee o cuando la PCCP detecte que los contadores `nhg` y `nvg` han llegado a sus valores límites. Al momento de finalizar los desplazamientos, dentro de los sub procesos se asignará valor 7 a la variable A, quedando así la función en reposo sin producirse ninguna acción hasta tanto el usuario no ingrese una nueva orden.

Almacenamiento de coordenadas en las memorias

Si la variable A tomara valor 1 o 2, o si se oprimieran los botones correspondientes de la botonera, se ingresará a la estructura `if()` de la memoria correspondiente. Allí las coordenadas actuales del posicionador se guardarán en las memorias `nv1` y `nh1` en caso de seleccionarse la memoria 1, y `nv2` y `nh2` en el caso de la memoria 2.

Condición de salida

Al enviar el carácter '0' a través de los puertos serie u oprimir el botón destinado a esta acción en la botonera, se ingresará a la última estructura `if()` en la cual a la variable `Funcion` se le asignará valor 0, por lo que la condición del bucle `while()` será falsa saliendo de la misma y por ello de la función de MOVIMIENTO LIBRE CONTINUO.

▪ [2.1] AVANCE VERTICAL CON PCCP

En esta sub función se ejecutará el avance en sentido vertical positivo.

Numero de pasos de las rampas de aceleración y desaceleración

En principio todo avance cuenta con tres etapas, en primer lugar la rampa de aceleración, a continuación un avance a velocidad continua, y por último la rampa de desaceleración. Cuando los desplazamientos fueran muy cortos (pocos pasos) puede no realizarse la etapa del avance continuo, ya que ni bien finalizaría la rampa de aceleración debería ya iniciarse la rampa de desaceleración. Incluso para desplazamientos muy cortos, de solo algunos pasos, la rampa de aceleración debería interrumpirse antes de finalizarse para iniciar ya la rampa de desaceleración. Este último caso se produciría cuando el posicionador se encuentre muy próximo al límite vertical positivo. En cualquier caso la cantidad de pasos de las rampas de aceleración y desaceleración estarán determinados por la variable `PasosAcel`, la cual, según el caso, podrá tomar valores desde 1 hasta `PasosAcelMax`, siendo este último la cantidad máxima de pasos de la rampa de aceleración, lo que implicaría que el posicionador alcanzaría la velocidad máxima.

En condiciones normales el posicionador acelerará hasta su velocidad máxima, es decir, `PasosAcel` tomará el valor máximo `PasosAcelMax`. Para esto, es necesario que el desplazamiento sea como mínimo de $2 * \text{PasosAcelMax}$ pasos. En caso de que la cantidad de pasos del desplazamiento deba ser menor por encontrarse el contador de pasos vertical `nvg` a menos de $2 * \text{PasosAcelMax}$ pasos de su valor máximo `nvgMax`, se deberá calcular la cantidad de pasos de la rampa de aceleración y desaceleración para que, al

finalizar el desplazamiento, nvg no supere a $nvgMax$. En este caso la cantidad de pasos de las rampas se determinaran mediante la expresión $PasosAcel = ((nvgMax - nvg) - 1)/2$.

Por ello, en primer lugar mediante una estructura `if ()` se determinará la cantidad de pasos de las rampas según la diferencia existente entre el contador nvg y el valor máximo alcanzable $nvgMax$. Habiendo determinado ya la cantidad de pasos de las rampas se ejecutará la rampa de aceleración y se ingresará al bucle de avance continuo.

Avance continuo

Ingresado a este bucle el posicionador avanzará un paso por cada ciclo del bucle. La condición de permanencia en este será que la variable A continúe con valor 8, o bien el pulsador correspondiente a esta acción permanezca accionado.

Como se indicó anteriormente, en caso de estar utilizándose la interfaz de LabVIEW o la App, la orden de detener el avance y dejar a la función en reposo vendría indicada por la recepción del carácter '7' mediante alguno de los puertos serie. Este carácter sería enviado desde las interfaces al momento de soltar los botones direccionales, por ello, dentro del bucle se monitoreará la llegada de datos a través del puerto serie para luego almacenarlos en la variable A cuando corresponda.

Al salir del bucle de avance continuo, se ejecutará la rampa de desaceleración deteniendo el posicionador.

Protección contra colisiones mediante la programación

En esta función la PCCP consiste en monitorear constantemente el valor de la variable nvg para evitar que este tome valores mayores que el límite $nvgMax$. Por ello antes de cada paso del avance continuo se verificará que el valor actual de la variable nvg sumado a la cantidad de pasos que deberán realizarse durante de la rampa de desaceleración, no sea mayor que $nvgMax$. Para ello se utilizará la primer estructura `if ()` dentro del bucle, la cual, en caso de ser verdadera la condición, haría finalizar el bucle y ejecutar la rampa de desaceleración, de manera que al finalizar el desplazamiento la variable nvg haya alcanzado exactamente el valor máximo $nvgMax$.

Por su parte, como ya se indicó más arriba, antes de iniciar la rampa de aceleración el valor de nvg será también monitoreado, de manera de calcular la cantidad de pasos de las rampas en caso de que no haya espacio suficiente para acelerar hasta el valor máximo y luego desacelerar.

En caso de que se haya alcanzado el límite vertical positivo ($nvg = nvgMax$) la última estructura `if ()` enviará esta información a LabVIEW de manera que este indique al usuario que se ha alcanzado el límite de la zona de trabajo. Al mismo tiempo se encenderá también en la botonera una luz junto al botón de avance horizontal positivo.

Pedido de posición

Durante toda la ejecución de esta función cada cierto intervalo de tiempo LabVIEW le solicitará al Arduino las coordenadas actuales del posicionador para poder indicarlo numérica y gráficamente en pantalla para visualización del usuario. Esta solicitud se realizará enviando desde LabVIEW al Arduino el carácter 'P' cada 4 ciclos del mismo (80 ms). La programación del Arduino, al recibir este carácter, invocará la función `ImprimirPosicion ()` del PROCESO PRINCIPAL. En esta función se enviará desde el Arduino al LabVIEW una cadena de caracteres con las coordenadas en pasos. Al finalizar el envío de esta cadena, la programación volverá al `void loop ()` en el mismo punto donde se invocó la función.

Ahora bien, recordando que si al utilizar la interfaz de LabVIEW se deseara realizar un avance continuo, al presionar uno de los botones direccionales se enviará el carácter correspondiente y se almacenará en la variable A, produciendo esto un avance en forma continua hasta que el botón sea liberado y la variable A cambie de valor. Por lo tanto, si durante el avance, LabVIEW solicita la posición enviando el carácter 'P', este modificará la variable A deteniendo anticipadamente el avance. Por ello, los caracteres recibidos desde LabVIEW se almacenarán primeramente en una variable intermedia M. Luego, mediante una estructura if (), si el valor de esta variable M fuera 'P' - 48 se invocará la función ImprimirPosicion (), y si fuera diferente se asignará este valor a la variable A. Esto se entiende también como que la variable A tomará el valor de M siempre y cuando el valor de esta no sea 'P' - 48, ya que este valor no indicaría una acción si no una solicitud de posición.

▪ [2.2] RETROCESO VERTICAL CON PCCP

En esta sub función se ejecutará el desplazamiento en sentido vertical negativo. El funcionamiento de esta sub función es similar al de la sub función [2.1] recién explicada. Las principales diferencias se encontrarían en el cálculo de la cantidad de pasos de las rampas de aceleración y desaceleración, en la protección contra colisiones dentro del bucle de avance continuo y en el hecho de que aquí la variable A deberá tomar valor 5 en lugar de 8 para ejecutarse el desplazamiento.

En este caso, como el desplazamiento será en sentido negativo y la variable *nvg* decrecerá con cada paso realizado, la rampa de aceleración podrá alcanzar el valor máximo siempre que al iniciar el desplazamiento el valor de *nvg* sea mayor que $2 * \text{PasosAcelMax}$, y en caso de ser menor, la cantidad de pasos de las rampas de aceleración y desaceleración vendrán determinadas por la expresión $\text{PasosAcel} = (\text{nvg} - 1)/2$.

La protección contra colisiones dentro del bucle while () de desplazamiento continuo monitoreará a la variable *nvg*, y en caso de que esta alcance el valor de *PasosAcel* se finalizará el bucle e iniciará la desaceleración, de manera tal que al finalizar el desplazamiento *nvg* posea valor 0.

▪ [2.3] AVANCE HORIZONTAL CON PCCP

Funcionamiento idéntico al [2.1] pero en sentido horizontal, y por ello utilizando la variable *nhg* y *nhgMax* en lugar de *nvg* y *nvgMax* respectivamente. Por su parte aquí la variable A deberá tomar valor 6 para realizarse el desplazamiento.

▪ [2.4] RETROCESO HORIZONTAL CON PCCP

Funcionamiento idéntico al [2.2] pero en sentido horizontal, y por ello utilizando la variable *nhg* y *nhgMax* en lugar de *nvg* y *nvgMax* respectivamente. Por su parte aquí la variable A deberá tomar valor 4 para realizarse el desplazamiento.

▪ [20] MOV. LIBRE CONTINUO (Sin PCCP)

Esta función se ubica dentro de un bucle `while ()` definido por las condiciones `Funcion == 2` y `CeroRealizado == 0`, por lo tanto la misma se ejecutará siempre que no se haya ejecutado previamente la función DETERMINAR CERO y por ello no se ejecutará la PCCP.

La programación de esta función y sus sub funciones [20.1], [20.2], [20.3] y [20.4] son idéntica a la [2] MOVIMIENTO LIBRE CONTINUO y sus sub funciones, salvo que se elimina el monitoreo de la variable `nvg` tanto en las condiciones de las estructuras `if ()` para ingresar a las sub funciones como dentro de los bucles de avance continuo. En estos desplazamientos las rampas de aceleración llegarán siempre hasta su valor máximo, es decir, será siempre `PasosAcel = PasosAcelMax`.

▪ [3] MOV. LIBRE ESCALONADO

En esta función la condición de ingreso será `Funcion == 3` y `CeroRealizado == 1`. Luego, las órdenes enviadas a través de las interfaces, el almacenamiento en la variable `A`, el ingreso a las estructuras `if ()` correspondiente a cada desplazamiento, al almacenamiento de las coordenadas en las memorias o a la condición de salida, funcionarán de la misma manera que en la función [2] MOVIMIENTO LIBRE CONTINUO.

La variación aquí se presenta en primer lugar en los desplazamientos, ya que en este caso al enviar una orden de avance el posicionador realizará una cantidad determinada de pasos en la dirección indicada y se detendrá. La cantidad de pasos que se avanzará en cada desplazamiento vertical u horizontal estará determinada por las variables `NumPasosVert` y `NumPasosHor` respectivamente, las cuales al declararse se les asigna el valor 10.

Avance discreto

Al recibir una orden de desplazamiento, ya sea por puerto serie o por accionamiento de un pulsador direccional, se ingresará a la estructura `if ()` correspondiente. Dentro de esta se calcularán las coordenadas que tendrá el posicionador después del desplazamiento. Para ello se sumará o restará a las variables `nvg` y `nhg` los valores `NumPasosVert` y `NumPasosHor` respectivamente según sea el caso. Así por ejemplo, en caso de un avance vertical, la coordenada final vertical será `nvg + NumPasosVert` mientras que la coordenada final horizontal se mantendría invariable en `nhg`. Estas coordenadas finales se almacenarán en las variables `nvFinal` y `nhFinal` para luego ejecutar con estas la función `IrACoordenada ()`.

Teniendo en cuenta que, si la cantidad de pasos avanzados es igual a 10, la duración de cada desplazamiento será sumamente breve, por lo que este finalizará antes de que el pulsador direccional que dio la orden pueda ser liberado. Por ello, para evitar que el desplazamiento se realice más de una vez, debe asegurarse que este no se vuelva a ejecutar hasta no soltarse y volverse a oprimir nuevamente el botón direccional. En el caso de utilizar los puertos serie esto se soluciona vaciando los buffers de entrada al finalizar cada desplazamiento y asignando a la variable `A` el valor 7 al inicio de cada nueva ejecución del bucle `while ()`, de esta manera la orden de parada la produce la misma programación y no las interfaces. Luego, al soltar el botón accionado, se recibirá el '7' producto de la liberación de este lo cual no produciría ningún efecto.

En el caso de estar utilizándose la botonera, luego de avanzarse los 10 pasos, en cada estructura `if ()` se ingresará en un bucle `while ()` sin sentencias en su interior y cuya condición de permanencia será que el pin digital asignado permanezca en estado HIGH. De esta manera la programación se mantendrá dentro del bucle sin realizarse ninguna acción hasta tanto no se suelte el botón direccional (por simplicidad esta acción no se muestra en el diagrama de flujo).

Protección contra colisiones mediante la programación

Dado que en esta función en primer lugar se determinará las coordenadas finales del posicionador para luego ejecutar la función `IrACoordenada ()`, la protección contra colisiones consistirá entonces en evitar que estas coordenadas finales tomen valores fuera de los límites de la zona de trabajo, es decir, que `nvFinal` no tome valores menores a 0 ni mayores a `nvgMax` y que `nhFinal` no tome valores menores a 0 ni mayores a `nhgMax`.

Para esto, si por ej., se quisiera realizar un avance vertical positivo y se determinase que la posición vertical final `nvFinal = nvg + NumPasosVert` fuera mayor a `nvgMax`, a la variable `nvFinal` se le asignará directamente el valor `nvgMax`. De la misma manera ante un retroceso vertical si `nvFinal = nvg - NumPasosVert` tomara valor menor a cero, a `nvFinal` se le asignará directamente el valor 0.

Al igual que en la función [2], si luego de finalizado algún desplazamiento se ha alcanzado algún límite de la zona de trabajo se enviará esta información a LabVIEW de manera que este indique al usuario, al mismo tiempo que se encenderá en la botonera la luz junto al botón direccional correspondiente.

▪ [30] MOV. LIBRE ESCALONADO (Sin PCCP)

Aquí la condición de ingreso al bucle `while ()` será ahora `Funcion == 3` y `CeroRealizado == 0`, ya que esta función se ejecutará siempre que el Cero Global no haya sido determinado.

La programación de esta función es idéntica a la anterior excepto que al ejecutar `IrACoordenada ()` se utilizarán los valores de `nvFinal` y `nhFinal` que resulten directamente del cálculo correspondiente, sin importar que hayan tomado valores menores a 0 o mayores a `nvgMax` y `nhgMax` respectivamente.

▪ [4] IR A MEMORIA 1

La condición de ingreso a la estructura `if ()` que contiene la programación de esta función como se vio será `Funcion == 4`. El objetivo de esta función será llevar al posicionador a las coordenadas `nv1` y `nh1` pertenecientes a la MEMORIA 1.

Una vez dentro de la estructura `if ()` en primer lugar se enviará la cadena “iF4F” y el carácter ‘4’ a las interfaces de LabVIEW y la App respectivamente, indicándoles así que esta función se encuentra en ejecución.

A continuación se dará a las variables `nvFinal` y `nhFinal` los valores almacenados en `nv1` y `nh1` respectivamente para luego con ellas ejecutar la función `IrACoordenada ()`.

Ya de vuelta en el `void loop ()` se vaciarán los buffers de entrada de los puertos serie para eliminar cualquier dato que se pudo haber recibido durante este proceso, se dará a la variable `Funcion` valor 0 y se finalizará la función.

▪ [5] IR A MEMORIA 2

La condición de ingreso a la estructura `if ()` que contiene la programación de esta función será `Funcion == 5`. El objetivo de esta función será llevar al posicionador a las coordenadas `nv2` y `nh2` pertenecientes a la MEMORIA 2.



El funcionamiento de esta función será idéntico al de [4] excepto que ahora se dará a las variables `nvFinal` y `nhFinal` los valores almacenados en `nv2` y `nh2`, y para indicar a las interfaces que esta función se encuentra en ejecución se enviarán ahora la cadena “¡F5F” y el carácter ‘4’ a las interfaces de LabVIEW y la App respectivamente (a la App se enviará el carácter ‘4’ ya que en esta se ejecutará el mismo bloqueo de teclas que con la función [4]).

▪ [6] MEDICIÓN AUTOMÁTICA

La condición de entrada para la ejecución de la estructura `if ()` que contiene esta función será `Funcion == 6`.

Al ingresar a la estructura `If ()`, en primer lugar se enviará la cadena “¡F6F” y el carácter ‘4’ a las interfaces de LabVIEW y la App respectivamente, indicándoles así que esta función se encuentra en ejecución (a la App se enviará el carácter ‘4’ ya que en esta se ejecutará el mismo bloqueo de teclas que con la función [4]).

Luego se almacenará las coordenadas del punto inicial mediante en las variables `nvr` y `nvh`. Este punto se llamará CERO RELATIVO y el mismo será el origen del Sistema de Referencias Relativo (SRR) utilizado en esta función.

Se dará valor 0 a las banderas `D` y `H`. La bandera `D` se utilizará para el sub proceso de ingreso de datos y detección de desbordes. Este sub proceso se ejecutará dentro de un bucle `while ()` cuya condición de ingreso y permanencia será `D == 0`. La bandera `H` se utilizará para ejecutar o cancelar el proceso de medición, por ello esta bandera deberá poseer valor 0 para que el proceso de medición se realice.

Ingreso de datos y detección de desbordes

En el sub proceso [6.1] que se verá más adelante se realizará el ingreso de datos por parte del usuario. Mediante estos se calculará la duración del proceso de medición, la existencia de desbordes y la distancia real entre mediciones. En este sub proceso se interactuará con el usuario mediante LabVIEW recibiendo y enviando información hasta que el usuario, si el programa se lo permite, de la orden de iniciar el proceso de medición o cancelarlo y finalizar esta función.

Posicionamiento inicial para el proceso de medición

Ya dentro del bucle `if ()` donde se realizará el proceso de medición automática, en primer lugar se llevará al posicionador al punto inicial del proceso de medición. Este punto será la posición del punto de medición inferior izquierdo del Sistema de Referencias Relativo. Para esto se calcularán las coordenadas verticales y horizontales de este punto, se asignarán estos valores a las variables `nvFinal` y `nhFinal` y se ejecutará la función `IrACoordenada ()`. Los valores de estas coordenadas serán determinados entonces mediante las expresiones:

$$\begin{aligned}nvFinal &= nvr - mvn * pv \\nhFinal &= nhr - mhn * ph\end{aligned}$$

Siendo `mvn`, `mhn`, `pv` y `ph` el número de mediciones en dirección vertical negativa y horizontal negativa y numero de pasos entre mediciones verticales y horizontales respectivamente. Estas variables se explican con detalle en [6.1]



Proceso de medición automática

En el sub proceso [6.2] se realizarán las mediciones. Esto incluye posicionar el instrumento de medición en cada punto, realizar el tiempo de espera y activar la Señal de Referencia durante el tiempo correspondiente.

Posicionamiento final y finalización de la función

Al completar el proceso de medición automática el posicionador se encontrará en el punto de medición superior derecho, por lo que se lo deberá llevar hasta el origen del Sistema de Referencia Relativo para así finalizar la función en el mismo punto en donde esta se inició. Para ello se dará a las variables `nvFinal` y `nhFinal` los valores `nvr` y `nh` y se ejecutará la función `IrACoordenada ()`.

Por último se vaciará los buffers de entrada por cualquier dato que pudiera haberse ingresado o quedado almacenado y se finalizará la función de medición automática, no sin antes dar valor 0 a la variable `Función`.

▪ [6.1] INGRESO DE DATOS Y DETECCIÓN DE DESBORDES

Dado que el usuario luego de enviar los datos para el proceso de medición tendrá la posibilidad de modificarlos o corregirlos, se ubica este sub proceso dentro de una estructura `while ()`. Aquí se utilizará a la variable `D` como bandera, la cual al poseer valor 0 indicará la permanencia dentro del ciclo. En cambio, al poseer los datos sus valores definitivos, o bien preferir cancelar el proceso, se modificará el valor de la bandera `D` finalizando así el bucle y continuando con el resto del proceso.

Ingreso de datos

En primer lugar se recibirán los datos para la medición provenientes de la aplicación de `labVIEW`. Estos datos estarán compuestos por 29 caracteres los cuales se enviarán de forma continua desde la aplicación hacia el Arduino cuando el usuario presione el botón `ENVIAR DATOS`. Por lo tanto la programación del Arduino deberá esperar la llegada de todos estos datos para poder continuar. Para esto se agrega al inicio un bucle `while ()` cuya condición `Serial.available () < 29` será verdadera siempre que haya menos de 29 bytes en el bus de entrada del puerto serie 0. Dentro de este bucle la única instrucción es una espera de 200 ms. Esto funciona entonces como una pausa en el programa a la espera de recibir por el puerto serie todos los datos. Así, solo cuando hayan sido recibidos los 29 caracteres la condición se volverá falsa saliéndose del bucle y continuándose con el proceso.

Ya estando los 29 caracteres en el bus de entrada estos se almacenarán ahora en un String declarado como `char DatosMedAut [30]`¹. Así, este String contendrá todos los datos enviados desde `LabVIEW` para el proceso de medición. Restará ahora descomponer el mismo para extraer los diferentes datos y almacenarlos cada uno en su respectiva variable. Para ello se utilizará un segundo String declarado como `char datos1 [6]` el cual irá tomando fragmentos del primer String para luego, mediante la función `atol`, convertir este String en variable numérica del tipo `long`. El orden en que los datos se enviaran desde `LabVIEW` y se ubicarán en el String `DatosMedAut [30]`, su extensión, unidades y variables en que se almacenarán serán los siguientes:

¹ Ver nota al pie en pag. 126 [Sub proceso void setup ()]



Bytes	Dato	Nº de bytes	Unidad	Variable asignada
[0-2]	Distancia entre mediciones horizontales	3	Mm	DistHor
[3-6]	Numero de mediciones horizontales positivas	4	Nº de med	mhp
[7-10]	Numero de mediciones horizontales negativas	4	Nº de med	mhn
[11-14]	Distancia entre mediciones verticales	3	Mm	DistVer
[14-17]	Numero de mediciones verticales positivas	4	Nº de med	mvp
[18-21]	Numero de mediciones verticales negativas	4	Nº de med	mvn
[22-26]	Tiempo de muestreo	5	mSeg	TiempoMuestreo
[27-29]	Tiempo de estabilización	2	Seg	TiempoEstabiliz

Nº de pasos entre mediciones, distancia real y duración del proceso de medición

Ya almacenado cada dato en su respectiva variable se determinará la cantidad de pasos que deberá realizarse entre cada medición horizontal y vertical y la distancia real entre ellas. Para esto se utilizará las variables de ajuste *ConversorDistPasosHor* (CnvDPH) y *ConversorDistPasosVert* (CnvDPV).

Como antes se explicó al realizar los motores paso a paso avances discretos difícilmente existirá un número de pasos con el cual se realice exactamente el desplazamiento en mm requerido. Por ello se determinará el número de pasos con el cual se obtenga el menor error y los mismos se almacenarán en las variables *ph* y *pv* para el desplazamiento horizontal y vertical respectivamente. En el caso de *ph* (procedimiento idéntico para *pv*) el mismo se determinará mediante el siguiente cálculo:

<pre>Int ph = CnvDPH*DistHor; Float phExtra = DistHor*CnvDPH; phExtra = (PhExtra - Ph)*100; If (phExtra > 49){ ph++; }</pre>	Almacena solo la parte entera del producto Almacena la parte entera y la decimal del producto Obtiene la parte decimal del producto multiplicada por 100 Si la parte decimal del producto alcanza o supera 0,5 se Incrementa la parte entera en una unidad
---	---

Luego, la distancia real entre mediciones horizontales será $ph/CnvDHP$.

En este punto se calculará también el tiempo aproximado de duración del proceso de medición. Para ello se tendrá en cuenta la cantidad de puntos de medición, los tiempos de estabilización y muestreo y los tiempos para realizar los desplazamientos. El tiempo T1 empleado para estabilización y muestreo, así como el tiempo T2 utilizado para los desplazamientos entre puntos de medición, vendrán dados por las expresiones indicadas a continuación:

$$T1 [\text{min}] = (mhn+mhp+1) * (mvn+mvp+1) * (\text{TiempoMuestreo} [\text{Seg}] + \text{TiempoEstabiliz} [\text{Seg}] + 1) / 60$$

$$T2 [\text{min}] = [ph*(mhn+mhp) * (2 * (mvn+mvp) + 1) + ph*mhn + pv*(mvn+2*mvp)] * T_{\text{entre pasos}} [\text{Seg}] / 60$$



Por simplificación, para el cálculo de T2 no se considera la variación de tiempo entre pasos en las rampas de aceleración y desaceleración, si no que se considera que el tiempo entre todos los pasos es el mismo e igual al tiempo entre pasos a velocidad máxima.

En la programación real al intentar calcular estos tiempos mediante una única expresión, se producen desbordes en las variables al utilizar los valores máximos posibles de cada dato. Por ello en estos casos se utilizan variables tipo `unsigned long` (max valor: 4.294.967.296) y las fórmulas se dividen y calculan por partes.

Detección de desbordes

A continuación se determinará la existencia de desbordes o mediciones fuera de rango. Para ello, conociendo las coordenadas en pasos del origen del SRR, la cantidad de mediciones en cada dirección y la cantidad de pasos entre ellas, puede calcularse si los valores máximos y mínimos que deberán tomar `nvg` y `nhg` se ubicarán dentro de la zona de trabajo. Por ello, para que existan desbordes en las mediciones deben verificarse alguna de las siguientes condiciones:

$nhr + mhp \cdot ph > nhgMax$	Desborde horizontal positivo
$nhr - mhn \cdot ph < 0$	Desborde horizontal positivo
$nvr + mvp \cdot pv > nvgMax$	Desborde vertical positivo
$nvr - mvn \cdot ph < 0$	Desborde vertical positivo

Si alguna de las condiciones se verifica se dará valor 1 a una de las banderas correspondientes.

Este procedimiento forma parte de la PCCP y es por esto que esta función solo podrá ejecutarse estándose trabajando con el SRG y por ello se desbloqueará recién luego de haberse ejecutado la función DETERMINAR CERO.

Confirmación del usuario

Una vez determinada la distancia real entre pasos, la duración del proceso de medición y la existencia de mediciones fuera de rango, tanto estos datos calculados, como los originalmente ingresados por el usuario, serán enviados a LabVIEW para su visualización en pantalla. Luego, el usuario deberá oprimir el botón de INICIAR MEDICIONES para comenzar con el proceso de medición, SALIR para anular el proceso y salir de la función o MODIFICAR VALORES para ingresar nuevos datos y repetir este sub proceso.

En este punto, de no haberse producido desbordes, Arduino enviará a la aplicación de LabVIEW la cadena "@M2M" indicando que se desbloqueen los botones SALIR, INICIAR MEDICIONES Y MODIFICAR VALORES. En cambio, de haberse producido algún desborde, se enviará la cadena "@M3M" para indicar que solamente se desbloqueen los botones SALIR y MODIFICAR VALORES.

Al oprimir los botones INICIAR MEDICIONES, SALIR o MODIFICAR VALORES se enviarán desde la aplicación de LabVIEW al Arduino los caracteres "I", "O" o "M" respectivamente. Al igual que en la espera de los datos para la medición, para aguardar la llegada del carácter que indique la orden a realizar se ingresará en un bucle `while ()` cuya condición de permanencia será `Serial.available () == 0` (ya que se espera la llegada de un único byte) y en su interior solo contendrá un `delay` de 50 mSeg. Al llegar el carácter que indique la orden a realizar, el mismo se almacenará en la variable `Modificar`. El valor de esta variable será luego analizado en una estructura `if ()`, si la misma es igual a "M" la condición será falsa por lo que la bandera D permanecerá en 0 repitiéndose este bucle `while ()` de ingreso de datos. En caso de

que Modificar fuera igual a "I" o "0" se verificará la condición de la estructura por lo que se dará valor 1 a la bandera D con lo cual se saldrá del bucle.

Luego, mediante una segunda estructura if () se diferenciará si el carácter recibido fue "I" o "0". En el caso de ser "I" por desearse continuar y realizar el proceso de medición la bandera H mantendrá el valor 0 dado inicialmente, y de haberse recibido un "0" a la bandera H se le dará valor 1. De esta manera, si H posee valor 0 podrá ingresarse a la estructura donde se realizará las mediciones y de lo contrario se producirá el fin de esta función como se muestra en diagrama de flujo del sub proceso [6].

▪ [6.2] PROCESO DE MEDICIÓN

Consideraciones

Las mediciones se realizarán en una cuadrícula de $mhn+mhp+1$ columnas separadas ph pasos horizontales y $mvn+mvp+1$ filas separadas pv pasos verticales. Si enumeramos las filas de abajo hacia arriba y las columnas de izquierda a derecha, el primer punto de medición será el ubicado en la fila N°1 columna N°1, es decir, el punto ubicado en el extremo inferior izquierdo de la cuadrícula. De allí se avanzará horizontalmente hasta muestrear todos los puntos de la fila N°1. Luego se continuará con los puntos de la fila N°2 iniciando el muestreo desde el punto ubicado en la primera columna.

La medición de los puntos de una fila comenzará entonces muestreándose el punto del extremo izquierdo. Luego se avanzará ph pasos horizontalmente hasta el siguiente punto, se realizará el nuevo muestreo y se volverá a avanzar. Ya que este procedimiento se repetirá para los $mhn+mhp+1$ puntos de muestreo de la fila, el mismo podrá realizarse dentro de un bucle for () cuya variable de control X vaya de $-mhn$ hasta mhp incrementándose en una unidad luego de cada ejecución. Dentro del bucle en primer lugar se realizará el muestreo del punto actual y luego se avanzará ph pasos hasta la posición siguiente, el cual será muestreado en la siguiente ejecución del bucle. Esta variable de control X en realidad no será otra cosa que la coordenada horizontal del punto de medición según el SRR, ya que la misma tomará valor $-mhn$ en el punto izquierdo de la cuadrícula y se incrementará una unidad cada vez que se avanzara al siguiente punto, tomando al final de la fila el valor mhp .

Por su parte, dentro de este bucle, el avance horizontal hasta el siguiente punto de muestreo deberá realizarse siempre que el punto actual de muestreo no sea el último de la fila, es decir, deberá realizarse solamente mientras sea $X < mhp$. Por ello, este avance se ubica dentro de una estructura if (), de manera que el mismo se realice únicamente si se verifica esta condición.

Una vez X haya tomado valor mhp y se haya realizado el muestreo del último punto de la fila, se saldrá del bucle y deberá regresar $(mhn+mhp)*ph$ pasos horizontalmente hasta la primera columna y avanzar pv pasos verticalmente, ubicándose así en el primer punto de la fila superior siguiente. Ya en esta nueva fila se repetirá el proceso de muestreo recién explicado iniciando nuevamente el bucle for (). Debido a que este procedimiento deberá repetirse $mvn+mvp+1$ veces por haber esa cantidad de filas, este podrá realizarse dentro de un segundo bucle for () cuya variable de control Y vaya de $-mvn$ hasta mvp incrementándose también una unidad en cada ejecución. Al igual que el caso de la variable de control X, Y sería ahora la coordenada vertical del punto de medición según el SRR. De igual manera que el avance horizontal, este retroceso horizontal y avance vertical hasta la siguiente fila deberá realizarse únicamente si la fila que acaba de muestrearse no es la última, es decir, mientras se verifique la condición $Y < mvp$.

Para los avances desde un punto de medición al siguiente se utilizará la función $IrACoordenada()$, habiendo para ello calculado previamente las variables $nvFinal$ y $nhFinal$ que serán luego las coordenadas del punto de destino.



Proceso de medición

Teniendo en cuenta las consideraciones anteriores, el diagrama de flujo del proceso de medición automática funcionará de la siguiente manera:

En primer lugar se ingresará a un bucle `for ()` cuya variable de control `Y` tomará el valor `-mvp` y luego a un segundo bucle cuya variable `X` tomará valor `-mhn`. Es decir, en este momento los valores de estas variables en el formato `(X ; Y)` serán las coordenadas en el SRR del punto de medición inferior izquierdo de la cuadrícula. A continuación se indicará a la aplicación de LabVIEW el valor de estas coordenadas y el porcentaje realizado del proceso, que en este primer momento será 0%. Luego se esperará el tiempo de estabilización y a continuación se activará la Señal de Referencia durante el tiempo indicado de muestreo. Realizado ya el muestreo se ejecutará la estructura `if ()` por ser en este primer caso `X = -mhn < mhp`. Dentro de esta estructura se ordenará el avance horizontal de `ph` pasos para ubicar el posicionador en el siguiente punto de medición. Ya finalizadas las instrucciones de este bucle la variable `X` se incrementará en una unidad y se ejecutará un nuevo ciclo del mismo, teniendo ahora la variable `X` valor `-mhn+1`. En este momento la coordenada del punto de medición será entonces `(-mhn+1 ; -mvp)`.

Teniendo ya la variable `X` valor `mhp-1` y ejecutado el muestreo, se avanzará horizontalmente hasta el último punto de medición de la fila. En este momento `X` se incrementará una unidad alcanzando el valor `mhp`. Con este valor de `X` podrá ejecutarse por última vez el bucle `for ()`, teniendo aquí el punto de medición las coordenadas `(-mvp;mhp)`. Se realizará entonces el muestreo y al finalizarlo, por ser `X = mhp` no se realizará otro avance horizontal, por lo que finalizadas las instrucciones del bucle se saldrá del mismo.

A continuación, habiéndose muestreado la primer fila y por ello ser `Y = -mvp < mvp`, la instrucción de la siguiente estructura `if ()` enviará al posicionador nuevamente hasta la primer columna pero en este caso a la segunda fila. Luego, por finalizarse las instrucciones del bucle, `Y` tomará ahora valor `-mvp+1` y se ejecutará nuevamente el mismo. Para ello se ingresará también al bucle `for ()` de variable de control `X` el cual se iniciará nuevamente con valor `-mhn`, repitiéndose todo el proceso de muestreo ya en la segunda fila. El primer punto de esta fila poseerá entonces coordenadas `(-mhn ; -mvp+1)` y el ultimo `(mhp ; -mvp+1)`. Finalizada todo esta fila se retrocederá horizontalmente y avanzará verticalmente hasta el primer punto de la tercera fila para repetir el proceso.

Estando ya en la fila `mvp-1`, al terminarse esta de muestrear se avanzará a la siguiente y última fila y la variable `Y` tomará valor `mvp`. Con este valor se ejecutará el bucle nuevamente y por última vez. Al finalizar el muestreo sobre esta última fila, por no cumplirse la condición `Y < mvp`, no se producirá un nuevo retroceso horizontal y avance vertical. Luego `Y` se incrementará una unidad y por ello no se volverá a ejecutar el bucle, finalizando así este sub proceso estando el posicionador en el punto de medición superior derecho.

Cancelación del proceso de medición automática

Durante este sub proceso puede cancelarse las mediciones si el usuario lo desea oprimiendo el botón de CANCELAR de la interfaz o el pulsador SALIR de la botonera. Para ello, al finalizar los tiempos de muestreo y al finalizar las funciones `IrACoordenada ()` que realizan los desplazamientos, se monitoreará el puerto serie 0 y el estado del pulsador SALIR (no se muestra en el diagrama de flujo). En caso de recibirse por el puerto serie el carácter '0' o ponerse el HIGH el pin digital correspondiente al pulsador, se finalizará de forma anticipada el proceso. Esto se consigue dando a las variables `X` e `Y` los valores `mhp+1` y `mvp+1` respectivamente, forzando de esta manera a finalizar los bucles `for ()` iniciales que tienen a estas como variables de control.

▪ [7] IR A COORDENADA

Esta función se ubica dentro de un bucle `while ()` definido por la condición `Funcion == 7`, por lo tanto el mismo se ejecutará hasta que esta condición resulte falsa. Como ya se explicó, la condición de salida será dando a la variable `Funcion` el valor 0.

El objetivo de esta función será llevar al posicionador a las coordenadas ingresadas por el usuario mediante la interfaz de LabVIEW. Estas coordenadas al ser ingresadas se almacenarán en las variables `nv3` y `nh3`, las cuales luego se utilizarán para ejecutar la función `IrACoordenada ()`.

Ingreso de datos

En primer lugar el Arduino aguardará la llegada de un carácter el cual le indicará la acción a realizar. Esta espera se realizará mediante el primer bucle `while ()` cuya condición de permanencia en el mismo será que el buffer de entrada del puerto serie 0 se encuentre vacío, es decir `Serial.available()==0`. Por ello la programación se mantendrá continuamente dentro del bucle en estado de espera hasta tanto se reciba el byte por el puerto serie. En este punto desde LabVIEW se enviará el carácter 'A' para indicar que a continuación se enviarán las coordenadas deseadas expresadas en pasos, 'B' para indicar que a continuación se enviarán las coordenadas en mm o 'O' para indicar que se desea finalizar la función. Luego, al llegar el carácter que indica la acción a realizar se saldrá del bucle de espera y este se almacenará en la variable `Coordenadas`.

A continuación se ingresará a la estructura `if ()` en la cual, si `Coordenadas` tuviera valor 'O' la condición sería falsa y por ello a la variable `Funcion` se le daría valor 0 avanzándose luego hasta el final del bucle `while ()` para salir del mismo por no verificarse la condición de permanencia y finalizar así la función.

En cambio, si el usuario deseara ir a alguna coordenada, al oprimir el botón IR en la aplicación de LabVIEW, esta enviará primeramente el carácter 'A' o 'B' según se haya seleccionado la opción de ingresar coordenadas en pasos o milímetros respectivamente. Luego, la aplicación esperará 100 mSeg (aguarda al siguiente ciclo del LabVIEW) y enviará 12 bytes que contendrán las coordenadas deseadas, siendo los primeros 6 los pertenecientes a la coordenada horizontal y los últimos 6 los de la vertical. Por ello, si la variable `Coordenadas` fuera igual a 'A' o 'B', se deberá esperar la llegada de estos 12 caracteres antes de continuar el proceso. Para esto se utilizará el bucle `while ()` sin instrucciones y cuya condición de permanencia será la presencia de menos de 12 bytes en el buffer de entrada del puerto serie 0, es decir, `Serial.available() < 12`. Con esto la condición resultará falsa recién luego de recibir y almacenar en el buffer los 12 caracteres.

Ya estando los 12 caracteres en el buffer de entrada estos se almacenarán en un String declarado como `char DatosIrACoord [13]`¹. Así, este String contendrá los datos de las coordenadas deseadas enviados desde LabVIEW. Restará ahora descomponer el mismo para extraer las diferentes coordenadas y almacenarlas en sus respectivas variables. Para ello se utilizará un segundo String declarado como `char datos2 [7]` el cual tomará fragmentos del primer String para luego, mediante la función `atoi`, convertir a este segundo String en variable numérica del tipo `long`. El orden en que los datos se enviarán desde LabVIEW y se ubicarán en el String `DatosIrACoord [13]`, su extensión, unidad y variable en que se almacenarán serán los siguientes:

¹ Ver nota al pie en pag. 126 [Sub proceso void setup ()]



	Bytes	Dato	Nº de bytes	Unidad	Variable asignada
Coordenadas = A	[0-5]	Coord. horizontal en pasos	6	Pasos	nh3
	[6-11]	Coord. vertical en pasos	6	Pasos	nv3
Coordenadas = B	[0-5]	Coord. horizontal en mm multiplicada por 10	6	Mm	nh3
	[6-11]	Coord. Vertical en mm multiplicada por 10	6	Mm	nv3

Asignación de las variables nh3 y nv3

Si las coordenadas se ingresaran en número de pasos, estas se almacenarán directamente en las memorias nh3 y nv3. En cambio, si las coordenadas se ingresaran en mm deberá determinarse el equivalente de estas en número de pasos para recién luego poder almacenar estos valores en las variables nh3 y nv3.

Como ya se explicó, al realizar los motores paso a paso avances discretos, difícilmente existirá un número de pasos con el cual pueda ubicarse en la coordenada en mm deseada. Por ello se determinará el número de pasos con el cual se obtenga el menor error y serán estos los que se almacenen en las variables nh3 y nv3. En el caso de nh3 (procedimiento idéntico para nv3) el mismo se determinará mediante el siguiente cálculo:

```
if (Coordenadas == 'A'){  
    nh3 = atol (datos2);  
}  
else{  
    nh3 = CnvDPH * atol (datos2)/10;  
    nh3Extra = CnvDPH * atol (datos2)/10;  
    nh3Extra = (nh3Extra - nh3)*100;  
    if (nh3Extra > 49){  
        nh3++;  
    }  
}
```

- (1) De ser la coordenada ingresada en número de pasos (Coordenada == 'A'), esta se almacenará directamente en la memoria nh3.
- (2) Para determinar el error primeramente se almacenará en nh3 la parte entera del producto dividido 10 (el cociente se debe a que por el puerto serie se recibirán las coordenadas ingresadas por el usuario multiplicadas por 10). nh3 almacenará solo la parte entera ya que esta es declarada como variable entera int.
- (3) Se almacenará luego en nh3Extra la parte entera y decimal del producto dividido 10, ya que nh3Extra es declarada como variable de punto flotante float.
- (4) Se obtiene la parte decimal del producto (3) y se lo multiplica por 100
- (5) Si la parte decimal del producto en (3) es mayor o igual a 0,5 indica que el menor error se obtendría con el paso siguiente, por lo que se incrementará nh3 en una unidad.

Ya asignado los valores correspondientes a las variables nh3 y nv3, antes de dirigirse a estas coordenadas, se comprobará que las mismas se encuentren dentro de la zona de trabajo del

posicionador. Al ingresar las coordenadas en la aplicación de LabVIEW los controles de la misma mediante su configuración impedirán que se ingresen valores negativos, por lo que solamente deberá comprobarse que se cumplan las condiciones $nh3 \leq nhgMax$ y $nv3 \leq nvgMax$. Por ello a continuación se ingresará a una estructura `if ()` cuya condición será que se cumplan estas condiciones.

En caso de ser la condición verdadera se dirigirá al posicionador hacia las coordenadas deseadas mediante la función `IrACoordenada ()`, dando previamente a las variables `nvFinal` y `nhFinal` los valores de `nv3` y `nh3` respectivamente. Ya alcanzada las coordenadas deseadas, se ejecutará el siguiente ciclo del bucle `while ()` que contiene toda la función por estarse cumpliendo la condición de permanencia `Funcion == 7` del mismo, ingresándose así nuevamente al bucle `while ()` de espera hasta recibir el carácter que indique la acción a realizar.

En caso de que la condición $nv3 \leq nvgMax$ y $nh3 \leq nhgMax$ sea falsa se imprimirá en la pantalla de la aplicación un mensaje indicando que las coordenadas ingresadas son incorrectas junto con el rango posible de valores que se pueden ingresar en pasos o mm, llegándose luego al final del bucle e iniciándose una nueva ejecución del mismo por seguir cumpliéndose con la condición de permanencia `Funcion == 7`.

▪ FUNCIÓN `IrACoordenada ()`

Esta función será la encargada de llevar al posicionador hasta las coordenadas `nvFinal` y `nhFinal` deseadas. La misma será invocada desde el `void loop ()`, donde previamente, según la función a ejecutarse se calculará el valor de las variables `nvFinal` y `nhFinal`.

En primer lugar se determinará la cantidad de pasos totales a avanzar en cada dirección y el sentido de cada desplazamiento. Para ello se calcularán las variables `V` y `H`, como la diferencia entre la coordenada actual y la coordenada de destino:

$$V = nv - nvFinal$$
$$H = nh - nhFinal$$

Luego si estas variables poseen signo negativo indicaría que el posicionador deberá avanzar en esa dirección o si poseen valor positivo se deberá retroceder. Luego, según el signo de estas se indicará a los driver el sentido de los desplazamientos activando o desactivando los pines digitales que indican los sentidos de avance.

Suponiendo ahora que el desplazamiento vertical fuera menor que el horizontal, en ese caso el desplazamiento vertical podría realizarse en un solo tramo y el horizontal podría dividirse en dos tramos. El primer tramo horizontal sería de la misma extensión que el desplazamiento total vertical, y el segundo tramo sería el restante para completar el desplazamiento horizontal total. Luego, el desplazamiento vertical y el primer tramo del horizontal podrían realizarse simultáneamente, por lo que el posicionamiento constará de dos partes: Desplazamiento común y Desplazamiento independiente. En el desplazamiento común se ejecutará de forma simultánea el desplazamiento vertical y el primer tramo del horizontal, y en el desplazamiento independiente se ejecutará el segundo tramo del horizontal.

Un análisis similar se realizaría si el desplazamiento vertical fuera mayor que el horizontal. Luego la cantidad de pasos del desplazamiento en común se almacenará en la variable `nAuxFinal1` y la del desplazamiento independiente en la variable `nAuxFinal2`. En ambos casos los desplazamientos se resumirían en la siguiente tabla:



	Dirección	nAuxFinal1 (Desplaz. comun)	nAuxFinal2 (Desplaz. indep.)
V < H	Vertical	V	-
	Horizontal	V	H - V
H > V	Vertical	H	V - H
	Horizontal	H	-

Para realizar estos desplazamientos se utilizará el contador auxiliar $nAux$ el cual contará cuantos pasos se realizan en cada etapa.

A partir de aquí ambos desplazamientos se realizarán de manera similar. Primeramente se calcula en cada caso los pasos $PasosAcel$ de las rampas de aceleración y desaceleración según el valor de $nAuxFinal1$ o $nAuxFinal2$. De allí se ejecutará la rampa de aceleración y se ingresará a un bucle `while ()` para ejecutar el avance continuo. La condición de este bucle será siempre verdadera, y para finalizarla deberá cumplirse la condición de que la suma de pasos ya realizados $nAux$ y la de los pasos aun por realizarse durante la rampa de desaceleración sean igual a la cantidad de pasos totales a realizar en cada desplazamiento. Es decir, para el desplazamiento en común debe ser $nAux + PasosAcel = nAuxFinal1$ y para el desplazamiento independiente $nAux + PasosAcel = nAuxFinal2$. Cumplida esta condición se ejecutará la sentencia `break` para finalizar el bucle de avance continuo e iniciar la rampa de desaceleración, la cual finalizará habiéndose realizado la cantidad exacta de pasos necesarios.



3RA PARTE

INTERFACES

16. TIPOS DE INTERFACES DEL POSICIONADOR

Las interfaces hombre maquina HMI son las que permitirán la interacción entre el operador y el posicionador. En este caso existirán 3 interfaces:

- Aplicación de LabVIEW
- APP (Android)
- Botonera

La primera de estas es la interfaz principal ya que solo desde esta se tendrá el manejo completo del posicionador, pudiendo con ella ejecutarse todas las funciones, visualizar datos y coordenadas en tiempo real e ingresar datos numéricos. Esta interfaz es una aplicación para su instalación en PC desarrollada mediante el Software LabVIEW y la misma se conectará al Arduino mediante cable USB.

Por otro lado se cuenta con la APP la cual es una aplicación para celulares Android desarrollada mediante la plataforma de programación App Inventor 2. Mediante esta interfaz se poseerá un manejo reducido del posicionador, ya que algunas funciones no podrán ejecutarse desde la misma debido a que esta no permite ingreso de datos numéricos. Tampoco desde esta podrán visualizarse las coordenadas del posicionador. La conexión con el Arduino se realizará a través de bluetooth, permitiendo así al operario manejar el posicionador desde cualquier lugar. El objetivo de crear esta interfaz inalámbrica es el permitir al operario manejar el posicionar desde dentro del túnel, facilitando así la tarea de ubicar el instrumento de medición de manera precisa en algún punto de interés.

Por último se encuentra la interfaz tipo botonera la cual está formada por un circuito electrónico compuesto de pulsadores y luces LED conectados a los pines digitales del Arduino. Al igual que la APP esta no permitirá el ingreso de valores numéricos y visualización de coordenadas, por ello solo algunas de las funciones podrán ejecutarse desde esta. Esta interfaz se encuentra instalada en la misma caja donde se ubica el Arduino, conformando así un único elemento. El objetivo de esta interfaz es el de contar con alguna manera, al menos reducida, de manejar el posicionador ante la ausencia de las otras interfaces, ya que esta siempre estará disponible por encontrarse montada junto con el Arduino. A su vez, por poseer un funcionamiento mucho más simple que las otras interfaces ya que no requiere programación ni tiene problemas de comunicación, fue muy útil al momento de desarrollar las otras dos interfaces para seguir el proceso y comprobar su funcionamiento.

Las interfaces se conectarán directamente al Arduino y podrán trabajar las 3 a la vez. Para ello Arduino según la función que se encuentre en ejecución enviará información a todas ellas por igual. De esta manera, si por ej. desde una de las interfaces se indicara ejecutar una función, la confirmación de la ejecución de la misma se enviará también a las otras, bloqueándose y desbloqueándose botones o encendiéndose y apagándose las luces correspondientes de todas estas. De esta manera se unifica el funcionamiento de las interfaces evitando que estas puedan enviar órdenes al Arduino en momentos incorrectos lo cual produciría el funcionamiento errático del posicionador

Así, cada interfaz poseerá características diferentes las cuales se indican en el siguiente cuadro comparativo:



	APLICACIÓN DE LABVIEW	BOTONERA	APP PARA CELULAR
DETERMINAR CERO	✓	✓	✓
MOV. LIBRE CONTINUO	✓	✓	✓
MOV. LIBRE ESCALONADO	✓	✓	✓
MEMORIZAR POSICIÓN (Función Mov. libre)	✓	✓	✓
IR A POSICIÓN 1 Y 2	✓	✓	✗
IR A COORDENADA	✓	✗	✗
MEDICIÓN AUTOMÁTICA	✓	✗	✗
SALIR	✓	✓	✓
PERMITE VISUALIZAR LAS COORDENADAS ACTUALES	✓	✗	✗
PERMITE INGRESAR VARIABLES DE AJUSTE	Permite ingreso de valores	Habilita los últimos valores utilizados	✗
INDICA LIMITE DE ZONA DE TRABAJO	✓	✓	✗
PERMITE RESETEO DEL ARDUINO	✓	✓	✗
CONEXIÓN INALÁMBRICA	✗	✗	✓



17. APLICACIÓN DE LabVIEW

Esta interfaz consiste en una aplicación desarrollada para su instalación en PC mediante la cual se controlará el posicionador. Esta será la interfaz principal, ya que solo a través esta se tendrá el manejo completo del posicionador y de todas sus funciones. Para crear esta aplicación se utilizó el software LabVIEW de National Instruments. Mediante este software, además de desarrollar la aplicación, se puede crear un ejecutable de la misma, es decir, un archivo que permite instalar únicamente la aplicación desarrollada en cualquier computadora, sin necesidad de tener que instalar también el software LabVIEW.

La conexión entre esta interfaz y el Arduino se realizará mediante cable USB, el cual al mismo tiempo se utilizará para alimentar eléctricamente a la placa.

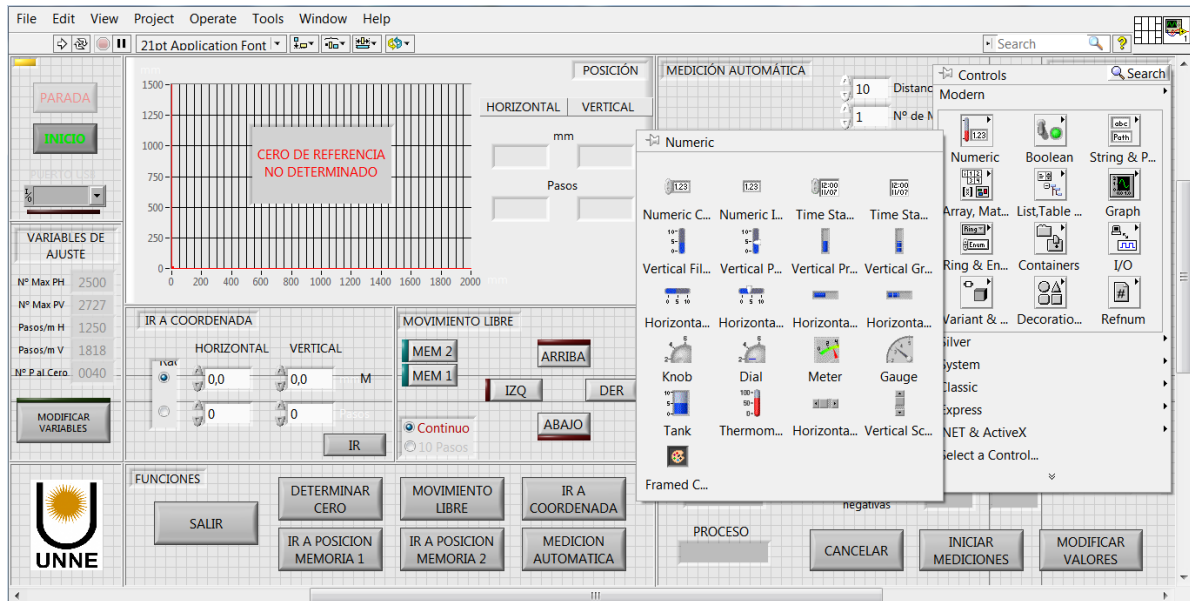
Si bien LabVIEW es el nombre del software mediante el cual se crea la aplicación, a los fines didácticos, se utilizará también este nombre para hacer referencia a la aplicación para control de posicionador.

17.1 LabVIEW

LabVIEW es un entorno de desarrollo integrado y diseñado específicamente para ingenieros y científicos. El mismo cuenta con un lenguaje de programación gráfico (G) que utiliza un modelo de flujo de datos en lugar de líneas secuenciales de código de texto. Gracias a esto puede desarrollarse de forma más fácil aplicaciones de adquisición de datos, análisis de medidas, desarrollar algoritmos o diseñar interfaces de usuario personalizadas y fáciles de usar. El software para la creación de aplicaciones consta de 2 partes:

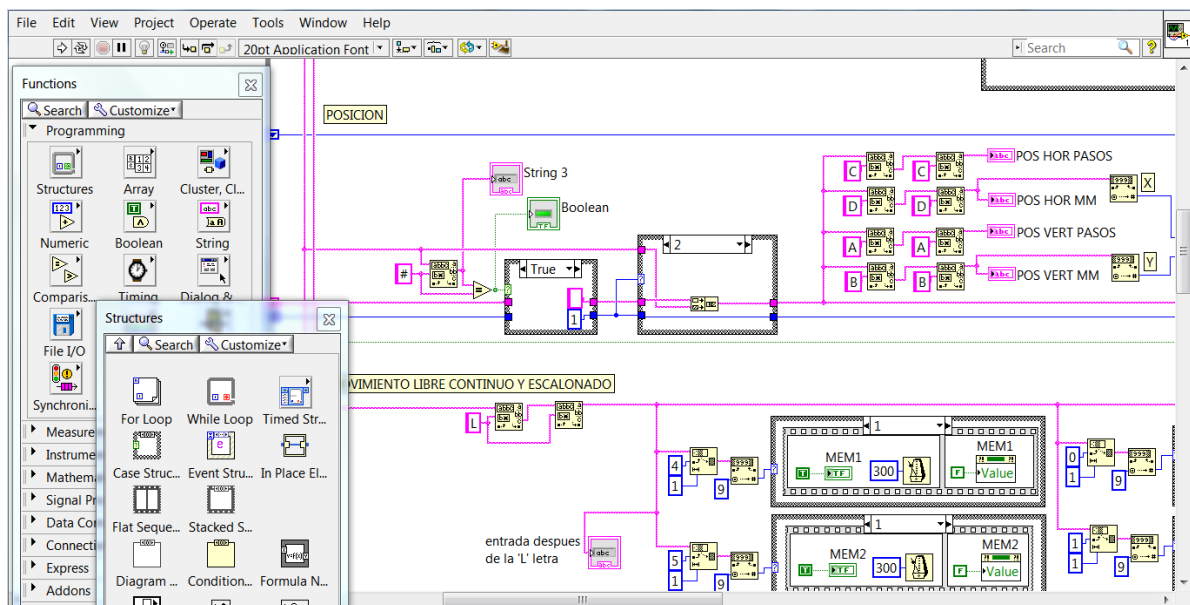
- PANEL FRONTAL

En esta parte se diseña la pantalla de la interfaz, la cual se utilizará para interactuar con el usuario cuando la aplicación esté en ejecución. En esta interfaz se ubicarán diferentes elementos como ser controles o indicadores numéricos, booleanos, strings, etc., listas, tablas, gráficos, Arrays, etc, mediante los cuales el usuario podrá interactuar con la programación. Estos elementos serán entonces las entradas y salidas mediante las cuales el usuario podrá ingresar órdenes o datos y observar el funcionamiento del proceso en tiempo real.



■ DIAGRAMA DE BLOQUES

En este sector se realiza la programación utilizándose para ello el lenguaje G o gráfico. Aquí se encuentra una terminal asignada a cada elemento del panel frontal, luego, según se requiera para el funcionamiento de la aplicación, se deberá completar la programación mediante objetos que realizan determinadas acciones como ser bucles, estructuras Case, Arrays, temporizadores, funciones lógicas o matemáticos, etc. Todas estas terminales y objetos serán interconectados mediante cables que indicarán el flujo de datos entre ellos, pudiendo estos ser datos numéricos, Strings, booleanos, etc, identificándose cada uno de ellos por distintos trazos y colores de cable.





17.2 INSTALACIÓN Y DESINSTALACIÓN DE LA APLICACIÓN DE LabVIEW

Para instalar la Aplicación deberá copiarse la carpeta “INSTALADOR_APLICACIÓN_LabVIEW” (240 MB) en la computadora donde se lo desea instalar y seguir los siguientes pasos:

- 1) Ejecutar el archivo “setup.exe” ubicado en la carpeta “INSTALADOR_APLICACIÓN_LabVIEW”.
En la ventana que aparece a continuación dar click en “Next”
En esta ventana se muestra las ubicaciones donde se instalará la aplicación y los productos de National Instruments. En caso de querer modificar las ubicaciones que aparecen por defecto puede hacerlo mediante el botón “Browse”. Luego dar click en “Next”
Seleccionar “I accept the above 2 License Agreement(s)” y dar click en “Next”
Dar click en “Next” y esperar que la instalación finalice
- 2) A continuación el programa le pedirá que reinicie la computadora
- 3) Crear un archivo de texto (.txt) con el nombre “Variables_de_ajuste.txt” en cualquier ubicación (por ej., en el escritorio). CORTAR este archivo (no copiar) y pegarlo dentro de la carpeta “data” ubicada donde se instaló la aplicación (por defecto C:\Program Files (x86)\POSICIONADOR\data) y reemplazar el archivo .txt del mismo nombre que allí se encuentra.

La aplicación ya estaría lista para utilizarse

NOTA: Si al querer ejecutar la aplicación aparece el mensaje "Unable to locate the LabVIEW Run-Time Engine..." debe instalarse el archivo "LVRTE2013SP1_f6Patchstd" (260 MB)

Para desinstalar la aplicación deberá dirigirse a Inicio > Panel de Control > Programas y Características. En la lista de programas habrá 3 que corresponden a la aplicación y que deberán desinstalarse. Estos serán:

- POSICIONADOR
- Software de National Instruments
- VISA Shared Components

17.3 DESCRIPCIÓN DE LA PANTALLA

La pantalla de esta interfaz está dividida en 7 cuadros.

- CUADRO SUPERIOR IZQUIERDO: Aquí se encuentran los botones de INICIO, PARADA y el selector del Puerto USB al cual se conectará el Arduino.
- CUADRO VARIABLES DE AJUSTE: En este se encuentran los indicadores/controles que permitirán visualizar o modificar las variables de ajuste.
- CUADRO FUNCIONES: En este recuadro se encuentran los botones para seleccionar la función a ejecutar o salir de la función en ejecución. Estos botones se bloquearán o desbloquearán a la vez que cambiarán de color según cual sea la función que se ejecute.

- **CUADRO POSICIÓN:** En este sector se visualizará la ubicación en tiempo real del posicionador. Es su parte derecha se encuentran indicadores que mostrarán las coordenadas del posicionador tanto mm como en pasos y en su parte izquierda se encuentra un gráfico a escala de la zona de trabajo, dentro del cual se mostrará la ubicación del posicionador respecto de la misma.
- **CUADRO IR A COORDENADA:** Aquí se ubican los controles que permiten ingresar las coordenadas deseadas por el usuario al cual desea que se dirija el posicionador mediante la función IR A COORDENADA. Los controles permanecerán bloqueados y se habilitarán únicamente cuando se ejecute esta función.
- **CUADRO MOVIMIENTO LIBRE:** En este sector se encuentran los botones direccionales del movimiento libre para indicar en qué dirección y sentido avanzar. Junto a cada botón direccional se encuentra una luz roja que se encenderá cuando el posicionador haya alcanzado el límite correspondiente de la zona de trabajo. En este recuadro se encuentran también los botones para guardar la posición en las memorias y sus respectivas luces para confirmar que el guardado de la posición fue realizado. Se encuentra aquí también el selector para elegir entre el movimiento libre continuo y el escalonado de 10 pasos.
- **CUADRO MEDICIÓN AUTOMÁTICA:** Este se utiliza para ingresar los datos y seguir el proceso de medición automática. En la parte superior se encuentran los controles para ingresar los datos deseados, como ser número de mediciones y distancias entre ellos y tiempos de muestreo y estabilización. En la parte inferior se encuentran indicadores que mostrarán la información arriba ingresada a modo de confirmación además de información calculada por el Arduino como ser distancia real entre mediciones, duración del proceso y existencia de mediciones fuera del rango. El indicador PROCESO de la parte inferior mostrará el avance del proceso de medición cuando este se esté realizando.



17.4 MANEJO LA APLICACIÓN DE LabVIEW

▪ EJECUCIÓN DE LA APLICACIÓN

La aplicación empezará a ejecutarse ni bien se habrá la misma, y se detendrá al oprimir el botón de PARADA.

En caso de haber detenido la aplicación, para volver a ejecutarla puede cerrarla y volverla a abrir, o bien, sin cerrarla, oprimir el botón “Run” simbolizado con la flecha ➡ que se encuentra encima del botón de PARADA.

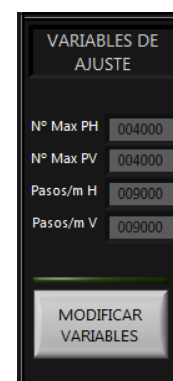
La ejecución de la aplicación puede ser confirmada mediante la luz amarilla ubicada en el extremo superior izquierdo de la pantalla, la cual se mantendrá encendida siempre que la función se esté ejecutando, apagándose al detenerse la misma.



▪ VARIABLES DE AJUSTE

En los 4 indicadores de este cuadro se mostrarán los valores de las variables de ajuste con los cuales se van a trabajar. Estas variables son:

- N° Max PH: Cantidad máxima de pasos que el posicionador puede desplazarse en dirección horizontal a partir del punto CERO GLOBAL [N° de pasos].
- N° Max PV: Cantidad máxima de pasos que el posicionador puede desplazarse en dirección vertical a partir del punto CERO GLOBAL [N° de pasos].
- Pasos / mH: Cantidad de pasos que se deben realizar horizontalmente para que el desplazamiento sea de 1 metro [N° de pasos].
- Pasos / mV: Cantidad de pasos que se deben realizar verticalmente para que el desplazamiento sea de 1 metro [N° de pasos].

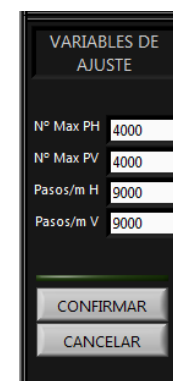


Los valores de estas variables deberán ser determinados por el operario luego de cada nuevo montaje del posicionador siguiendo las indicaciones descriptas en el capítulo 13.2.

Al ejecutar la aplicación estos indicadores mostrarán los últimos valores con los cuales se trabajó, los cuales habrán quedado guardados en el archivo “Variables_de_ajuste.txt” creado en el paso 3 de la instalación. De manera que si el posicionador no fue montado nuevamente ni sufrió ninguna modificación estos valores tampoco necesitarán ser modificados.

Si se necesitara modificar los valores de las variables se deberá oprimir el botón MODIFICAR VARIABLES con lo cual los indicadores se convertirán en controles, permitiendo el ingreso mediante teclado de los valores deseados. Luego estos nuevos valores se guardarán al oprimir el botón CONFIRMAR o se mantendrán los valores anteriores si se oprime el botón CANCELAR.

La modificación de estas variables solo puede realizarse antes de oprimir el botón de INICIO, luego de lo cual esta opción quedará bloqueada y para modificar nuevamente las variables se deberá cerrar la aplicación y luego volverse a ejecutar.



Una vez que se oprime el botón de INICIO los valores de las variables mostrados en este cuadro se enviarán al Arduino y este luego los regresará a la aplicación de LabVIEW a modo de confirmación. En ese momento los valores recibidos se visualizarán en estos mismos indicadores y se encenderá la luz verde

ubicada encima del botón MODIFICAR VARIABLES indicando así que los valores mostrados son los recibidos desde el Arduino y con los cuales este trabajará. Será recién en este momento que las variables en uso se almacenarán en el archivo .txt.

Por seguridad, mientras se estén modificando los valores de estas variables se bloqueará el botón de INICIO

NOTA: Al iniciar la aplicación por primera vez luego de su instalación los indicadores se mostrarán vacíos ya que a estas variables no se les asignó ningún valor anteriormente. Deberá entonces asignarse obligadamente valores a estas variables para poder operar. En este caso podría asignarse, solo como una aproximación: N° Max PH = 99999; N° Max PV = 99999; Pasos/mH = Pasos/mV = 1000. Con estos valores las coordenadas en mm mostrado en la pantalla serán incorrectas pero el posicionador podrá moverse libremente.

▪ CONEXIÓN AL ARDUINO MEDIANTE PUERTO USB

Si el Arduino aún no fue conectado mediante USB a la computadora en este punto deberá hacerlo. Al conectarse el Arduino se alimentará eléctricamente y se ejecutará el Sketch almacenado en él, lo cual se deberá verificar observando que la luz LED ubicada encima del pulsador SALIR en la interfaz tipo botonera se encuentre encendida.

▪ SELECCIÓN DE PUERTO USB

Una vez asignado los valores de las variables de ajuste con los cuales se trabajará se deberán seleccionar el puerto USB al cual se conectará el Arduino con la computadora. Para ello se utiliza el selector ubicado debajo del botón de INICIO, el cual desplegará una lista donde deberá seleccionar el puerto en uso.

Este paso deberá realizarse antes de oprimir el botón de INICIO. En caso de oprimir este botón sin antes haber seleccionado el puerto USB aparecerá una ventana emergente con la leyenda "NO SE SELECCIONÓ NINGÚN PUERTO USB" y se encenderá una luz roja debajo del selector.



NOTA 1: Si al conectar el Arduino al puerto USB dice que no se encontraron controladores para el dispositivo y en el selector de puertos USB solo aparece la opción "Refresh" debe instalarse el Software de Arduino "arduino-1.8.3-windows.exe" o alguna versión más reciente.

Si luego de instalado el software en el selector sigue sin aparecer el puerto donde se conectó el Arduino, desconectar y reconectar el USB y en el selector poner "Refresh" varias veces hasta que aparezca el puerto correspondiente.

NOTA 2: Si no sabe a cuál puerto se ha conectado el Arduino puede dirigirse a Inicio > panel de control > Administrador de dispositivos > Puertos (COM y LPT) y allí se indicará el puerto al cual se está conectado.

▪ INICIO DE LA APLICACIÓN

Una vez que las variables de ajuste tienen su valor correspondiente y se ha seleccionado el puerto USB al cual se conectó el Arduino puede iniciarse la aplicación oprimiendo el botón de INICIO.

En este momento se realizará la conexión entre la aplicación y el Arduino y este último se reseteará, lo cual podrá apreciarse mediante la luz de la función SALIR en la botonera, la cual se apagará y volverá a encender. Debido a que el reseteo del Arduino en ocasiones tarda varios segundos, aparecerá en pantalla una ventana con la leyenda “ESPERE” la cual se mantendrá durante 10 segundos, asegurándose así que ha finalizado el reseteo.

En este punto la aplicación de LabVIEW enviará a Arduino las variables de ajuste y, como ya se indicó, estas las devolverá a modo de confirmación para visualizarlos en los indicadores correspondientes. Al mismo tiempo, para indicar que estas variables fueron cargadas correctamente en la programación, en la interfaz tipo botonera se encenderán en secuencia todas las luces de las funciones y se volverán a apagar¹.

Finalizado esto aparecerá en pantalla una ventana emergente con la leyenda “DESEA DETERMINAR EL CERO DE REFERENCIA?” junto con las opciones DETERMINAR CERO, SALTEAR y AYUDA. Mediante esta puede seleccionarse ejecutar la función DETERMINAR CERO ya en un principio habilitando así todas las funciones o no hacerlo y ejecutarla más adelante cuando se desee. En la opción AYUDA se coloca una advertencia indicando las diferencias y riesgos al trabajar sin ejecutar esta función.

IMPORTANTE: Si en este momento se produjeran problemas de comunicación entre el Arduino y la aplicación esto se observaría ya que el LED verde sobre el botón SALIR de la botonera no parpadearía y no se produciría el encendido en secuencia recién mencionado de las luces LED. A su vez en la aplicación no se habilitarían los botones de selección de función. En este caso se deberá cerrar la aplicación y volverla a abrir. Si luego de esto en el selector del puerto USB no se mostrara el puerto en uso deberá oprimirse repetidas veces la opción “Refresh” o bien desconectar y reconectar el cable USB en diferentes puertos.

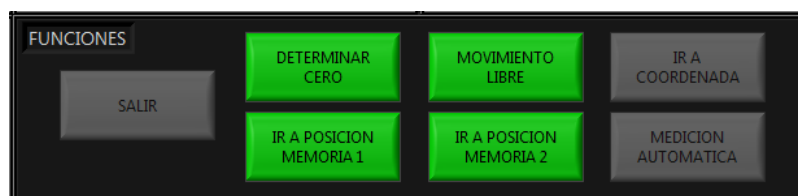
▪ SELECCIÓN DE FUNCIONES

En el cuadro FUNCIONES se encuentran los botones para seleccionar la función que se desea ejecutar. Las funciones habilitadas y que pueden ejecutarse se mostrarán de color verde y estarán desbloqueadas, mientras que las funciones deshabilitadas estarán de color gris y bloqueadas.

En tanto no se ejecute la función DETERMINAR CERO y por ello se esté trabajado con el SRI, las funciones IR A COORDENADAS y MEDICIÓN AUTOMÁTICA permanecerán bloqueadas como se explicó en el capítulo 12.1, ya que estas trabajan solo con el SRG.

Una vez ejecutado DETERMINAR CERO y estando entonces trabajando con el SRG, todas las funciones estarán disponibles para su uso, bloqueándose solo esta primera ya que no es necesario volver a ejecutarla en tanto no se detenga la aplicación.

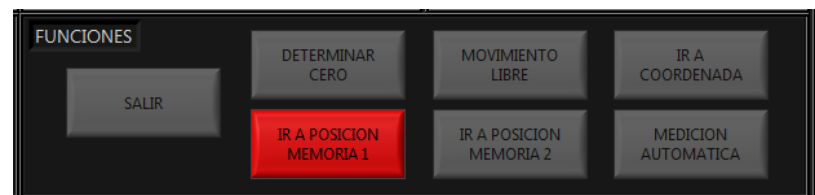
Al seleccionar una función la aplicación enviará la orden al Arduino, el cual confirmará la ejecución de la función devolviendo a la aplicación una



¹ El objetivo real del encendido de estas luces se indica en el capítulo 19.2 al explicar el uso de la interfaz tipo botonera.

cadena de caracteres en particular. Luego de recibir esta confirmación el botón de la función en ejecución se pondrá de color rojo mientras que los de las demás funciones se pondrán de color gris, indicándose de esta manera la función en uso. A su vez todos los botones de funciones además de cambiar de color se bloquearán, evitando de esta manera posibles inconvenientes.

Por su parte el botón de SALIR se pondrá de color verde y se desbloqueará solo cuando la función ejecutándose sea MOVIMIENTO LIBRE, IR A COORDENADA o MEDICIÓN AUTOMÁTICA, ya que solo en estas funciones el operador es el que dará la orden para finalizarla. En tanto en la función DETERMINAR CERO y las funciones IR A MEMORIA la finalización de la función es automática.



En particular, durante la función MEDICIÓN AUTOMÁTICA, el botón de SALIR no se desbloqueará desde un comienzo sino recién luego de enviarse los valores para las mediciones.

▪ POSICIÓN

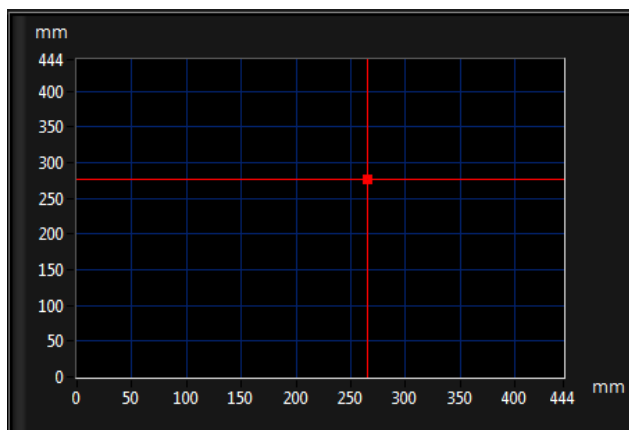
En los indicadores de la parte derecha de este recuadro se observarán las coordenadas tanto en pasos como en mm del posicionador.

Las coordenadas en pasos serán siempre las reales, ya que se determinan contando los pasos realizados por los motores en cada dirección. En cambio, las coordenadas en mm son calculadas a partir de estas coordenadas en pasos y de las variables de ajuste Pasos/mH y Pasos/mV. Por lo tanto si estas variables no son correctas por estar mal determinadas o mal ingresadas las coordenadas en mm mostradas serían erróneas.

Estos indicadores funcionarán y mostrarán las coordenadas en todo momento, ya sea que se trabaje con el sistema de referencias inicial o el global. En caso de trabajar con el sistema inicial las coordenadas mostradas podrán tomar valores negativos dependiendo de la posición del cero de este sistema de referencia, mientras que al trabajar con el sistema global los valores mostrados siempre serán positivos



Por su parte el grafico es una representación a escala de la zona de trabajo, donde se indica mediante el cursor rojo la ubicación real del posicionador dentro de esta. Las unidades de los ejes se encuentran en mm para facilitar su interpretación siendo los valores máximos de los mismos calculados a partir las variables de ajuste sabiendo el número máximo de pasos que se puede realizar en cada dirección y transformándolos a mm mediante las variables Pasos/mH y Pasos/mV. De esta manera el grafico indicará siempre la zona de trabajo del posicionador.



Para que la visualización de la posición en este gráfico tenga sentido debe estar trabajando con el Sistema de Referencias Global, ya que solo en este sistema se conoce la ubicación real del posicionador respecto de la zona de trabajo. Es por ello que desde el inicio y hasta que se ejecute la función DETERMINAR CERO el gráfico permanecerá bloqueado y con la leyenda “CERO DE REFERENCIA NO DETERMINADO”, y recién luego de ejecutar esta función el mismo podrá visualizarse.

▪ FUNCIÓN DETERMINAR CERO

En caso de no haberse ejecutado esta función en el inicio, el botón de la misma permanecerá de color verde y desbloqueado, así esta función podrá ejecutarse en el momento que el operador lo desee.

Al ejecutarla el posicionador buscará automáticamente el Punto de Referencia Físico y a partir de este determinará el punto cero global. Esta función finalizará automáticamente sin intervención del operario, por ello el botón SALIR del cuadro de FUNCIONES permanecerá bloqueado.

Al finalizar este proceso, en el recuadro FUNCIONES todos los botones de selección de funciones se desbloquearán y aparecerán en color verde, mientras que el botón correspondiente a esta función se pondrá en color gris y bloqueará. A su vez, en el cuadro POSICIÓN se desbloqueará el gráfico y tanto este como los indicadores de posición mostrarán la coordenada (0 ; 0), ya que al finalizar esta función el posicionador se localizará en el punto cero global.

En caso de que ocurra una falla en algún sensor de posición el punto cero global no podrá ser determinado y mediante una ventana emergente se indicará cual sensor estaría presentando problemas.

▪ FUNCIÓN MOVIMIENTO LIBRE

Al ejecutarse esta función se habilitarán los botones del cuadro correspondiente al mismo y en botón SALIR del cuadro FUNCIONES.

En la parte inferior izquierda del cuadro MOVIMIENTO LIBRE se encuentra un selector mediante el cual se podrá elegir el tipo de movimiento libre a utilizar, si el continuo o el escalonado de 10 pasos.

Para realizar el desplazamiento del posicionador se deberán utilizar los botones direccionales ARRIBA, ABAJO, IZQUIERDA y DERECHA. En caso de estar trabajando en el modo CONTINUO, al pulsar sobre uno de los botones direccionales se enviará al Arduino la orden de avanzar en la dirección correspondiente y al liberarlo se enviará la orden de detener el avance. Por ello al pulsar y mantener pulsado uno de los botones direccionales, el posicionador avanzará de forma continua hasta que el botón sea liberado.



En cambio, cuando se está trabajando en el modo ESCALONADO (10 pasos), al pulsar los botones direccionales se enviará al Arduino la orden de avanzar en la dirección correspondiente y el posicionador

se desplazará únicamente 10 pasos en esta dirección, independientemente de que el botón direccional haya sido liberado o no. Luego, para avanzar otros 10 pasos el botón deberá ser oprimido nuevamente o liberado y vuelto a oprimir.

En caso de haberse ejecutado anteriormente la función DETERMINAR CERO y estando por ello trabajando con el sistema de referencia global, se encontrará activa la Protección Contra Colisiones Mediante la Programación. Por ello cuando se hiciera avanzar el posicionador en una dirección y este alcanzara el límite de la zona de trabajo, el Arduino detendrá el posicionador y enviará a la aplicación una cadena de caracteres en particular. Mediante esta cadena se bloqueará el botón de avance en esta dirección y encenderá una luz roja junto a él, indicando así que se ha alcanzado el límite al cual se puede avanzar en esa dirección. Luego, al avanzar en sentido contrario y por ello abandonar el límite de la zona de trabajo, la luz se apagará.



A demás, al iniciar la función, si el posicionador se encontrara en uno de los límites de la zona de trabajo, el botón correspondiente se bloquearía ni bien se inicie la función.

En caso de no haberse ejecutado la función DETERMINAR CERO y por ello no encontrarse activa la protección contra colisiones mediante la programación, el posicionador avanzará cada vez que el operario se lo indique, sin importar la posición en la que se encuentre. Por ello debe tenerse especial cuidado cuando se esté trabajando en la cercanía de la estructura. De cualquier manera una colisión se evitaría mediante los Fines de Carrera de Protección, pero ello implicaría la desconexión de la alimentación de los drivers.

Estando utilizando esta función podrá guardarse la posición actual en una de las dos memorias existentes. Para ello simplemente debe oprimirse los botones MEM 1 o MEM 2 según cual memoria quisiera utilizarse. Al oprimirse alguno de estos botones se enviará la orden al Arduino el cual confirmará la acción devolviendo a la aplicación una cadena de caracteres en particular que hará encender por un breve momento la luz verde ubicada junto a la memoria, confirmando de esta manera que el guardado de la posición en la memoria fue realizado correctamente.



En caso de haberse guardado una posición en una de las memorias y se deseara guardar una nueva posición en esa misma memoria, al oprimir el botón la posición actual se sobrescribirá a la posición anterior guardada.

Si se guardase una posición estando utilizando el Sistema de Referencias Inicial, al ejecutar la función DETERMINAR CERO y empezar a trabajar con el Sistema de Referencia Global, esta posición guardada no se perderá ya que las coordenadas almacenadas en la memoria se convertirán de un sistema a otro, pudiéndose así volver a el mismo punto guardado antes de realizar el cambio de sistema de referencia.

Para finalizar esta función se deberá oprimir el botón SALIR del cuadro FUNCIONES.

IMPORTANTE: Al estar utilizando esta función en modo CONTINUO, si se deseara realizar avances muy cortos, al accionar los botones direccionales por momentos muy breves (un único clic rápido), en ocasiones la aplicación no reconoce que el botón haya sido liberado, por lo que el posicionador avanzará permanentemente hasta tanto se vuelva a oprimir el botón o bien se cambie de posición el mouse. El motivo de este error nace en el propio funcionamiento del LabVIEW el cual, al oprimirse y liberarse rápidamente el botón, no registra el momento de liberación del mismo, lo cual impide que este problema pueda

solucionarse mediante la programación propia de esta aplicación. Por lo cual se recomienda que para realizar avances muy pequeños se utilice esta función en modo ESCALONADO, se utilice la función IR A COORDENADAS, o bien la función MOVIMIENTO LIBRE en modo CONTINUO pero realizando los desplazamientos desde alguna de las otras dos interfaces.

▪ FUNCIONES IR A MEMORIA 1 E IR A MEMORIA 2

Al ejecutar estas funciones el posicionador se dirigirá a las coordenadas guardadas en la memoria correspondiente. Esta función finalizará automáticamente una vez se haya alcanzado la posición almacenada, por lo que el botón SALIR del cuadro de FUNCIONES permanecerá bloqueado.

Cabe resaltar que las memorias almacenarán por defecto las coordenadas (0 ; 0) hasta tanto las mismas no se sobrescriban con otras coordenadas. Por lo tanto, mientras no se haya utilizado alguna de las memorias para almacenar una posición, al ejecutar la función IR A MEMORIA correspondiente a esta memoria, el posicionador se dirigirá al origen del sistema de referencias con el cual se esté trabajando en ese momento.

▪ FUNCIÓN IR A COORDENADA

Al ejecutarse esta función se desbloquearán los botones y controles del cuadro IR a coordenadas y el botón SALIR del cuadro FUNCIONES

Mediante el selector ubicado a la izquierda del cuadro se deberá seleccionar el tipo de coordenadas a ingresar, en mm o en pasos. Según la opción seleccionada se desbloquearán los controles correspondientes de este tipo de coordenadas y se bloquearán los correspondientes al otro.

Ya ingresadas la coordenada horizontal y la vertical deseada se oprime el botón IR. En este momento la aplicación enviará los datos de las coordenadas deseadas a Arduino y este los analizará. De ser correctas las coordenadas ingresadas, Arduino enviará una cadena de caracteres específicos a la aplicación de LabVIEW para informar que el posicionador se dirigirá hacia ese punto. Esta confirmación se utilizará para bloquear el botón IR y el botón SALIR. Una vez alcanzada las coordenadas indicadas, Arduino enviará otra cadena de caracteres confirmando que el posicionador ha llegado a dicho punto, con lo cual los botones utilizados en esta función volverán de desbloquearse.

En el caso de que el Arduino al analizar las coordenadas enviadas por la aplicación detecte que estas se encuentran fuera de la zona de trabajo, hará imprimir en una ventana emergente la leyenda “Coordenada fuera de rango” junto a la cual se indicará también, según el tipo de coordenada que se esté utilizando, el rango de valores permitidos que puede tomar cada una de estas.

Para finalizar esta función se deberá oprimir el botón SALIR.

▪ FUNCIÓN MEDICIÓN AUTOMÁTICA

Al ejecutar esta función se activarán los controles de la parte superior del cuadro MEDICIÓN AUTOMÁTICA junto con el botón ENVIAR VALORES.



Allí se encuentra un gráfico que representa al Sistema de Referencia Relativo, siendo el origen del mismo la ubicación actual del posicionador. En los extremos de sus ejes se encuentran controles mediante los cuales se deberá ingresar la cantidad de mediciones que se desea realizar en cada dirección y sentido a partir del origen. Se encuentran también junto a estos, dos controles para indicar la distancia en mm que desea realizarse entre mediciones horizontales y verticales. Por otro lado, arriba a la derecha se ubica los controles para ingresar los tiempos de muestreo y estabilización deseados.

Una vez ingresado los parámetros se deberá oprimir el botón ENVIAR VALORES, con lo cual estos datos serán enviados al Arduino. Este, según los parámetros recibidos calculará si existen desbordes en las mediciones, la distancia real entre mediciones verticales y horizontales, y el tiempo total del proceso de medición. A continuación, Arduino devolverá a la aplicación los datos recibidos desde esta más los calculados, los cuales se visualizarán en pantalla. Este conjunto de datos se mostrará en los indicadores de la parte inferior del recuadro.

Los desbordes se producen cuando, según los parámetros ingresados, se calculara que algunos puntos de medición se ubicarían fuera de la zona de trabajo. Estos desbordes dependerán entonces tanto del número de mediciones en cada sentido y la distancia entre ellos, como de la ubicación del origen del sistema de referencias.

Si existieran desbordes en las mediciones estos serían indicados mediante las luces rojas ubicadas debajo de la leyenda MEDICIONES FUERA DE RANGO, las cuales se encenderían mostrando las direcciones en las cuales se produjeron. Estos desbordes se indicarán también en el gráfico superior mediante luces ubicadas junto a los controles correspondientes.

En este punto, si no se han producido desbordes se

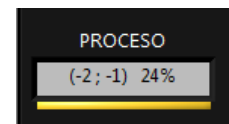


habilitarán los botones INICIAR MEDICIÓN y MODIFICAR VALORES ubicados en la parte inferior del recuadro, como así también el botón SALIR del cuadro FUNCIONES. En caso de haberse producido algún desborde el proceso de medición no podría realizarse, por lo que solamente se habilitarían los botones MODIFICAR VALORES y SALIR.

Si se seleccionara MODIFICAR VALORES se habilitaría nuevamente los controles de la parte superior para ingresar los datos para la medición y el botón ENVIAR VALORES, permitiendo así modificar o corregir los parámetros ingresados anteriormente y volver a enviarlos.

Oprimiendo el botón SALIR finalizaría la función.

Al oprimir el botón INICIAR MEDICIONES, el proceso de medición automática comenzaría. En este momento se activaría el indicador de PROCESO ubicado en la parte inferior izquierda del recuadro. Este indicador irá mostrando la coordenada del punto actual de medición, respecto del sistema de Referencias Relativo, a medida que el proceso se va ejecutando y el porcentaje del proceso ya realizado. Debajo de este indicador aparecerá también una luz amarilla la cual se encenderá y apagará indicando los momentos en los cuales se realizan los muestreos.



En este momento se desbloqueará también el botón CANCELAR. Al ser este oprimido, se finalizará la medición en el punto actual y luego el posicionador se dirigirá al punto en el cual se inició la función luego de lo cual la función finalizará. Durante este intervalo el indicador PROCESO mostrará el mensaje "CANCELANDO" indicando así que esta orden fue recibida.

18. APP PARA CELULAR

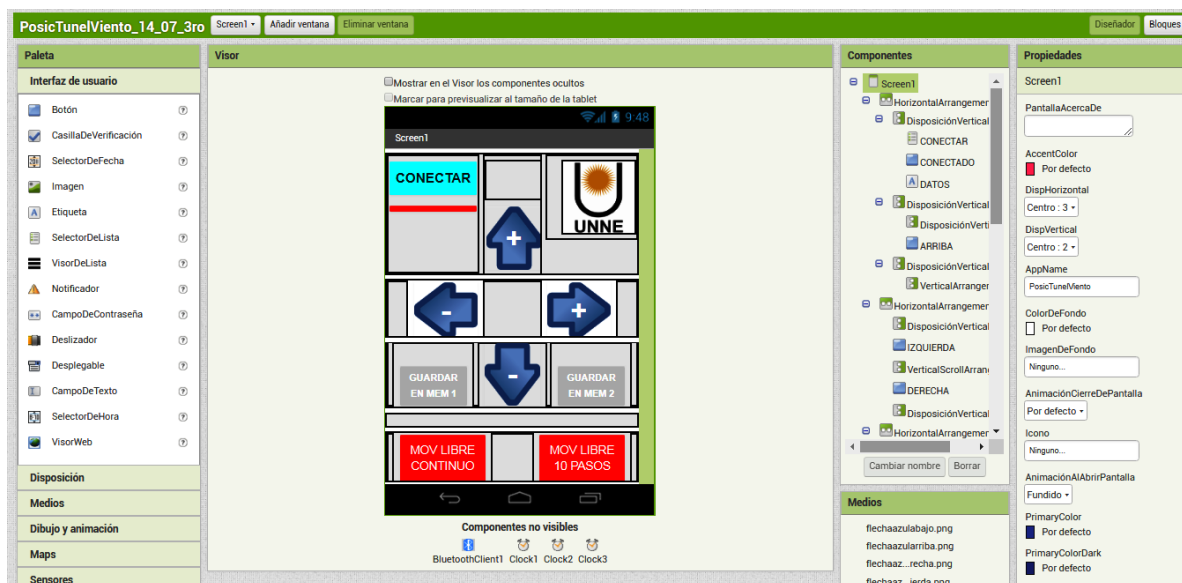
Esta interfaz consiste en una aplicación desarrollada para su instalación en celulares con sistema operativo Android, la cual permitirá el manejo del posicionador de forma remota mediante bluetooth. La misma fue desarrollada a través la plataforma de programación visual App Inventor 2.

18.1 APP INVENTOR 2

App Inventor 2 es una plataforma de programación creada por el MIT (Instituto Tecnológico de Massachusetts) y que fue adoptada por Google para sus usuarios como solución para crear de forma sencilla aplicaciones para dispositivos Android. El programa de creación consta de 2 partes:

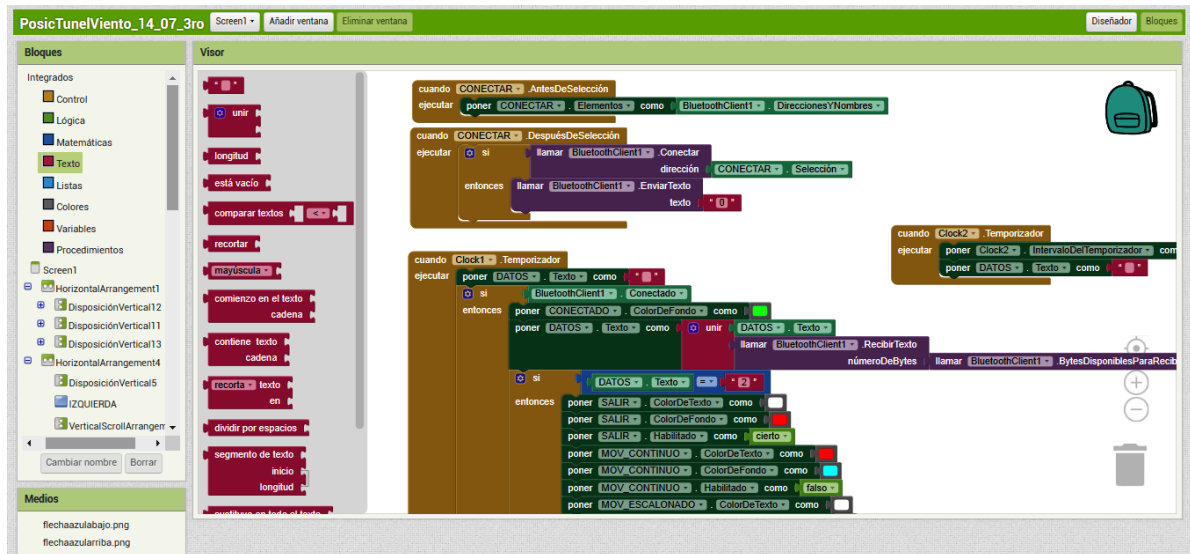
■ DISEÑADOR

En esta parte se diseña la pantalla de la aplicación, es decir, como se verá la aplicación en el dispositivo móvil. Aquí se ubicarán los distintos componentes como ser botones, luces, imágenes, textos, listas, etc y se configuran las propiedades de los mismos como ser posición, tamaño, color, estado inicial, tamaño y tipo de letra, etc.



■ BLOQUES

Aquí se realiza la programación de forma visual e intuitiva utilizando bloques drag and drop (arrastrar y soltar).



Una vez terminada la aplicación se puede generar el instalador APK obteniéndose un código QR para su descarga al celular, o bien crear el propio archivo APK para descargar a la CPU y luego enviar al celular.

18.2 INSTALACIÓN DE LA APP PARA CELULAR

- 1) Copiar el archivo APP_POSICIONADOR.apk (3,37 MB) en un celular de sistema operativo Android.
- 2) Abrir el archivo y seleccionar "Instalar".
- 3) Opcionalmente puede abrirse la aplicación u oprimir "Hecho" y finalizar la instalación sin abrirla.

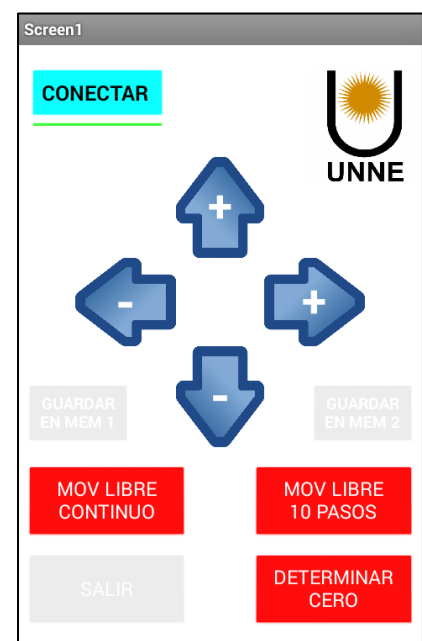
La aplicación ya estaría lista para utilizarse.

18.3 DESCRIPCIÓN DE LA PANTALLA

En la parte superior izquierda el botón CONECTAR se utiliza para enlazar el celular con el dispositivo bluetooth conectado al Arduino.

Debajo de estos se encuentran los botones direccionales y los botones de guardar posición, utilizados por las funciones MOVIMIENTO LIBRE.

En la parte inferior se encuentran los botones que inician las funciones y el botón SALIR para finalizarlas. En este caso mediante esta interfaz solo podrán ejecutarse las funciones de MOVIMIENTO LIBRE y DETERMINAR CERO.



18.4 MANEJO DE LA APP PARA CELULAR

▪ EJECUCIÓN DE LA APLICACIÓN

La aplicación empezará a ejecutarse ni bien se abra la misma y el celular se conecte al módulo Bluetooth, y se detendrá al cerrar la aplicación.

▪ VARIABLES DE AJUSTE

Esta interfaz no permite ingresar ni cargar los valores de las variables de ajuste.

▪ CONEXIÓN AL ARDUINO MEDIANTE BLUETOOTH

En primer lugar deberá encenderse el modulo bluetooth conectado al Arduino mediante el interruptor ubicado en el lateral de la interfaz tipo botonera junto a la leyenda BLUETOOTH. Al oprimir el interruptor se encenderá una luz led verde ubicado encima de este indicando que el modulo está recibiendo alimentación eléctrica.



A continuación habrá que vincular el celular con el modulo. Para ello desde el celular deberá buscarse el dispositivo e ingresar la clave correspondiente. En este caso el nombre del mismo será "HC-05" y la clave "1234".

Luego, ya en la aplicación al seleccionar el botón CONECTAR en la parte superior izquierda de la pantalla aparecerá una lista con todos los dispositivos bluetooth vinculados al celular. Deberá seleccionarse entonces en la lista el hc-05.

Cuando la conexión se haya realizado correctamente la luz de color rojo en forma de línea horizontal ubicada debajo del botón CONECTAR, se pondrá de color verde, indicando así que la interfaz está lista para ser utilizada.

NOTA: Si se cambiara el modulo por otro el nombre podría variar, siendo los más comunes "HC-05", "HC-06"¹ y "TOTEISON". La clave también puede variar de un módulo a otro siendo por lo general "1234" o "0000".

▪ SELECCIÓN DE FUNCIONES

En la parte inferior de la pantalla se encuentran los botones para seleccionar la función que se desea ejecutar. En este caso solo se encontrarán los botones para iniciar las funciones de MOVIMIENTO LIBRE y DETERMINAR CERO ya que solo estas pueden ejecutarse desde esta interfaz. Se encuentra también el botón SALIR para finalizar las funciones de MOVIMIENTO LIBRE. A diferencia de la interfaz de LabVIEW donde al ejecutar la función MOVIMIENTO LIBRE luego se selecciona trabajar de forma continua o escalonada de 10 pasos, en este caso deberá



¹ En estos casos los nombres HC-05 y HC-06 hacen referencia directamente al modelo del módulo.

seleccionarse directamente el tipo de movimiento libre deseado.

Las funciones habilitadas y que pueden ejecutarse se mostrarán de color rojo y estarán desbloqueadas, mientras que las funciones deshabilitadas estarán de color gris y bloqueadas (caso de la función DETERMINAR CERO luego de que la misma ya se haya ejecutado).

Al seleccionar una función la aplicación enviará la orden al Arduino, el cual confirmará la ejecución de la función devolviendo a la aplicación un caracter en particular. Luego de recibir esta confirmación el botón de la función en ejecución se pondrá de color celeste con letras rojas, mientras que los de las demás funciones se pondrán de color gris, indicándose de esta manera la función en uso. A su vez todos los botones de selección de funciones además de cambiar de color se bloquearán, evitando de esta manera posibles inconvenientes.

Por su parte el botón de SALIR se pondrá de color rojo y se desbloqueará solo cuando la función ejecutándose sea de MOVIMIENTO LIBRE, ya que la función DETERMINAR CERO finalizará de forma automática.

▪ FUNCIÓN MOVIMIENTO LIBRE

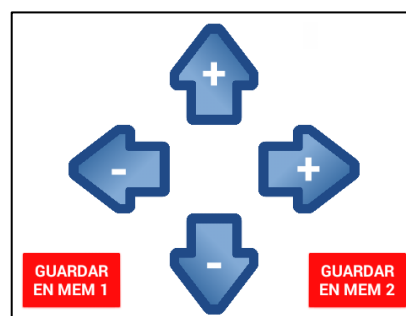
Al ejecutarse una de estas funciones se habilitarán los botones direccionales, los botones para guardar la posición en una de las memorias y en botón SALIR.

Para realizar el desplazamiento del posicionador se deberán utilizar los botones direccionales. Al igual que lo explicado para la aplicación de LabVIEW, en caso de estar trabajando en el modo CONTINUO, al pulsar y mantener pulsado uno de los botones direccionales, el posicionador avanzará en la dirección correspondiente de forma continua hasta que el botón sea liberado.

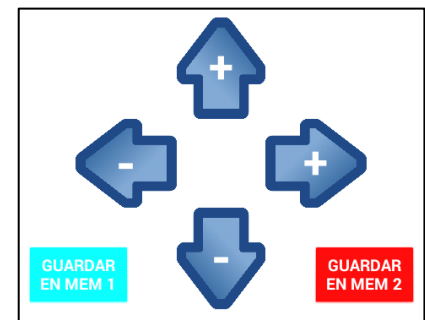
En cambio, si se trabajara en el modo ESCALONADO de 10 pasos, al pulsar los botones direccionales el posicionador se desplazará únicamente 10 pasos en la dirección, independientemente de que el botón direccional haya sido liberado o no. Luego, para avanzar otros 10 pasos el botón deberá ser oprimido nuevamente o liberado y vuelto a oprimir.

En caso de haberse ejecutado anteriormente la función DETERMINAR CERO y estando por ello trabajando con el SRG, se encontrará activa la PCCP. Por ello cuando se hiciera avanzar el posicionador en una dirección y este alcanzara el límite de la zona de trabajo el Arduino detendrá el avance. Para estos casos esta interfaz no cuenta con indicadores luminosos que indiquen que el posicionador se encuentra en uno de los imites y no se bloquearán los botones direccionales. Deberá suponerse que esto ha ocurrido simplemente porque el posicionador no se moverá en la dirección indicada, y podrá verificarse visualizando en alguna de las otras interfaces si efectivamente esto ha sucedido.

En caso de no haberse ejecutado la función DETERMINAR CERO y por ello no encontrarse activa la PCCP, el posicionador avanzará cada vez que el operario se lo indique, sin importar la posición en la que se encuentre. Por ello debe tenerse especial cuidado cuando se esté trabajando en la cercanía de la estructura. De cualquier manera una colisión se evitaría mediante los Fines de Carrera de Protección, pero ello implicaría la desconexión de la alimentación de los drivers.



Estando utilizando una de estas funciones podrá también guardarse la posición actual en una de las dos memorias existentes. Para ello se utilizarán los botones GUARDAR EN MEM 1 y GUARDAR EN MEM 2 según cual memoria quisiera utilizarse. Al oprimir alguno de estos botones se envía la orden al Arduino el cual confirma la acción devolviendo a la aplicación un caracter en particular. En este caso, mediante esta confirmación se hará cambiar brevemente el color del botón, indicando que el guardado de la posición en la memoria se realizó correctamente.



Como se explicó anteriormente, en caso de haberse guardado una posición en una de las memorias y se deseara guardar una nueva posición en esta misma, al oprimir el botón de memorizar la nueva posición sobrescribirá a la anterior guardada. Además, si se guardase una posición estando utilizando el SRI, esta no se perderá al empezar a trabajar con el SRG ya que las coordenadas almacenadas se convertirán de un sistema a otro, pudiéndose así volver a el mismo punto guardado antes de realizar el cambio de sistema de referencia.

Para finalizar esta función se deberá oprimir el botón SALIR.

19. BOTONERA

Esta interfaz consiste en un circuito electrónico compuesto por pulsadores y luces tipo LED, la cual permitirá el manejo del posicionador de forma reducida y permitirá seguir el proceso del mismo. El mismo se encuentra conectado directamente al Arduino por lo que estará siempre disponible, a diferencia de la Aplicación de LabVIEW y la APP las cuales pueden o no estar conectadas.

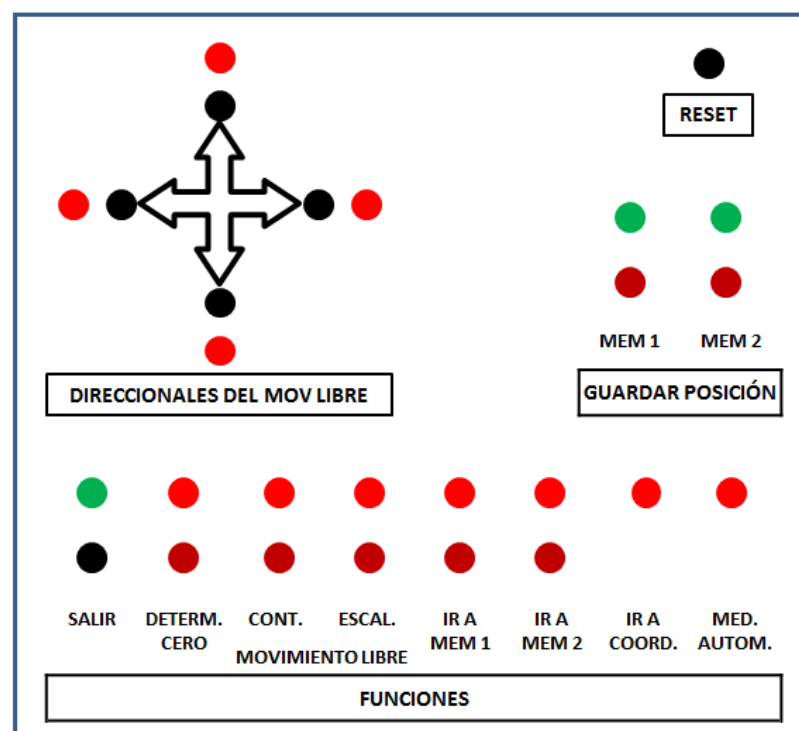
19.1 DESCRIPCIÓN DE LA BOTONERA

En la parte inferior se encuentran los botones para iniciar las funciones acompañados encima de ellos por luces LED que indicarán cuando la función correspondiente se encuentre en ejecución. Como se explicó anteriormente las funciones IR A COORDENADAS y MEDICIÓN AUTOMÁTICA requieren ingreso de datos numéricos por parte del usuario, lo cual no puede realizarse desde esta interfaz, por lo que en esta no se encuentran botones para iniciar las mismas, aunque si se encuentran los LEDs correspondiente para indicar cuando estas se encuentran en ejecución.

En la parte superior izquierda se encuentran los botones direccionales de las funciones MOVIMIENTO LIBRE acompañadas cada uno de un LED el cual indicará cuando el posicionador se encuentra en alguno de los límites de la zona de trabajo cuando se esté trabajando con en SRG.

A la derecha de estos se encuentran los botones para guardar la posición actual en alguna de las memorias, cada uno acompañado de su correspondiente luz LED utilizado para confirmar el guardado de la posición.

En la parte superior derecha de la interfaz se encuentra el botón RESET el cual resetea el Arduino en caso de que sea necesario.





19.2 MANEJO DE LA BOTONERA

▪ HABILITACIÓN DE LA BOTONERA

La botonera se alimentará eléctricamente desde el Arduino, por lo tanto, siempre que este último se encuentre alimentado la botonera estará en condiciones de ser utilizada.

▪ VARIABLES DE AJUSTE

Como se indicó al explicar este mismo ítem en la aplicación de LabVIEW, las variables de ajuste requieren el ingreso de estos datos numéricos por parte del usuario. Estas variables son requeridas obligadamente al inicio de la programación del Arduino, por lo que el posicionador no podrá funcionar hasta tanto no se defina el valor de estas variables. Debido a esto, si no se pudiera contar con la interfaz de la aplicación de LabVIEW para ingresar el valor de las variables, estas deberían poder ser ingresadas de otra manera. Es por ello que los últimos valores utilizados en estas variables serán almacenados en la memoria no volátil EEPROM del Arduino, y por lo tanto al iniciarse la programación el usuario deberá optar por ingresar nuevos valores a estas variables mediante la Aplicación de LabVIEW o bien habilitar el uso de los valores almacenados en la EEPROM.

Esta habilitación del uso de los últimos valores se podrá realizar desde la botonera, para ello, ni bien se alimente eléctricamente la placa Arduino o la misma sea reseteada, deberán oprimirse de forma simultánea los botones SALIR y MOVIMIENTO LIBRE CONTINUO. En este punto, para que el operario pueda confirmar que las variables de ajuste fueron correctamente cargadas, las luces que indican la ejecución de las funciones se encenderán en secuencia y se volverán a apagar. De esta manera el operario sabrá que las variables de ajuste ya fueron cargadas y el posicionador está listo para ser utilizado.

IMPORTANTE: En esta interfaz las variables de ajuste se utilizarán principalmente para la PCCP ya que mediante la misma no se puede visualizar las coordenadas del posicionador. Por ello es de suma importancia que el posicionador no haya sido modificado estructuralmente o se le hayan agregado nuevos elementos desde el momento en que las variables utilizadas y guardadas en la EEPROM fueron determinadas, ya que de otra manera los valores utilizados en estas variables serían obsoletos pudiendo quedar inutilizado en PCCP por trabajar con valores incorrectos en estas variables.

▪ CONEXIÓN AL ARDUINO

Esta interfaz se encuentra montada en la misma caja donde se ubica el Arduino por lo que la conexión con este es permanente y no requiere ningún tipo intervención por parte del operador.

▪ SELECCIÓN DE FUNCIONES

En la parte inferior de la botonera se encuentran pulsadores rojos con los cuales se podrá seleccionar la función que se desea ejecutar. En este caso solo se encontrarán los pulsadores para iniciar la función DETERMINAR CERO, las funciones de MOVIMIENTO LIBRE y las de IR A MEMORIA, ya que IR A COORDENADAS y MEDICIÓN AUTOMÁTICA no pueden ejecutarse desde esta interfaz. Se encuentra además un pulsador negro SALIR para finalizar las funciones de MOVIMIENTO LIBRE, teniendo en cuenta que las otras finalizarán de forma automática sin intervención del operador.



Al seleccionar una función el pulsador accionado pondrá en HIGH el pin digital correspondiente, indicando así cual función se debe ejecutar. Luego Arduino confirmará la ejecución de la función encendiendo el LED rojo ubicado encima del pulsador que fue oprimido, el cual permanecerá encendido hasta que la función finalice. A diferencia de las otras interfaces, aquí al estar ejecutándose una función los pulsadores que inician las otras no podrán bloquearse, pero de todas maneras su accionamiento no produciría ningún efecto.

Mientras no se esté ejecutando ninguna función permanecerá encendido el LED verde ubicado encima del pulsador de SALIR.

▪ FUNCIÓN MOVIMIENTO LIBRE

Para realizar el desplazamiento del posicionador se deberán utilizar los pulsadores direccionales ubicados en la parte superior izquierda. Al igual que en las otras interfaces, en caso de trabajando en el modo CONTINUO, al accionar uno de estos pulsadores el posicionador avanzará en la dirección correspondiente de forma continua hasta que el mismo sea liberado.

Si se trabajara en el modo ESCALONADO de 10 pasos, al accionar los pulsadores direccionales el posicionador se desplazará únicamente 10 pasos en la dirección, independientemente de que el pulsador haya sido o no liberado. Luego, para avanzar otros 10 pasos el pulsador deberá ser accionado nuevamente o liberado y vuelto a accionar.

En caso de haberse ejecutado anteriormente la función DETERMINAR CERO y estando por ello trabajando con el SRG, se encontrará activa la PCCP. Por ello cuando se hiciera avanzar el posicionador en una dirección y este alcanzara el límite de la zona de trabajo el Arduino detendrá el posicionador y se encenderá el LED rojo ubicado junto al botón direccional correspondiente. Y en caso de no haberse ejecutado la función DETERMINAR CERO y por ello encontrarse desactivada la PCCP el posicionador avanzará cada vez que el operario se lo indique, sin importar la posición en la que se encuentre.

En esta interfaz la posición actual podrá guardarse en las memorias accionando alguno de los pulsadores ubicados a la derecha de los botones direccionales. Al oprimir alguno de estos pulsadores se envía la orden al Arduino el cual confirma la acción encendiendo brevemente el LED verde ubicado encima del pulsador, indicando así que el guardado de la posición fue realizado.

Para finalizar esta función se deberá oprimir el pulsador de SALIR.



20. COMUNICACIÓN ENTRE ARDUINO Y LAS INTERFACES

Durante el funcionamiento, Arduino y las interfaces intercambiarán permanentemente información. Para poder realizar este intercambio de manera ordenada y que la información no se mezcle se requiere de un código de comunicaciones. Este le permitirá a cada interfaz clasificar y ordenar la información recibida para su posterior utilización dentro de los procesos.

Este código consiste en enviar caracteres específicos o cadena de caracteres desde un dispositivo al otro, los cuales, significarán diferentes cosas según el momento de la programación en que sean enviados o recibidos. Así, según cuál sea el carácter o caracteres recibidos por el dispositivo receptor este ejecutará una acción determinada.

20.1 DESDE LAS INTERFACES HACIA ARDUINO

La información enviada desde las interfaces al Arduino podrán ser órdenes o datos.

20.1.1 ÓRDENES

Una orden indicaría al Arduino que este debe realizar una acción en particular como por ej., iniciar una función, guardar la posición en una memoria, avanzar en una dirección determinada o detener el avance, etc., Es decir, ordenarían al Arduino que se realice una de las acciones que tiene programado. Estas órdenes se enviarán desde las interfaces al Arduino mediante el envío de un único carácter, el cual, según el momento del programa en que sea enviado, indicaría realizar una acción u otra. Así, por ej, si no se estuviera ejecutando ninguna función, el envío del carácter '4' daría la orden de ejecutar la función IR A MEMORIA 1. En cambio si estuviera ejecutándose la función MOVIMIENTO LIBRE el envío de este mismo carácter daría la orden de avanzar en sentido horizontal negativo.

▪ SUB PROCESO SELECCIÓN DE FUNCIÓN

No estándose ejecutando ninguna función, desde las interfaces deberá enviarse al Arduino el carácter indicando la función que se desea iniciar, para ello desde las mismas se enviarán los siguientes caracteres según sea en caso:

Aplicación de LabVIEW	App	Orden al Arduino
1	1	Iniciar función DETERMINAR CERO
2	2	Iniciar función MOVIMIENTO LIBRE CONTINUO
3	3	Iniciar función MOVIMIENTO LIBRE ESCALONADO
4	-	Iniciar función IR A MEMORIA 1
5	-	Iniciar función IR A MEMORIA 2
6	-	Iniciar función MEDICIÓN AUTOMÁTICA
7	-	Iniciar función IR A COORDENADA

▪ FUNCIONES MOVIMIENTO LIBRE

En este momento el usuario podrá ordenar el desplazamiento del posicionador hacia alguna de las direcciones, guardar la posición en alguna de las memorias o bien finalizar la función. Por otra parte la aplicación de LabVIEW ordenará de forma automática el envío de las coordenadas actuales. Cada una de estas órdenes serán representadas por los siguientes caracteres:

Aplicación de LabVIEW	App	Orden al Arduino
8	8	Avanzar en dirección vertical positiva
5	5	Avanzar en dirección vertical negativa
6	6	Avanzar en dirección horizontal positiva
4	4	Avanzar en dirección horizontal negativa
7	7	Detener el avance
1	1	Guardar posición en memoria 1
2	2	Guardar posición en memoria 2
P	-	Solicitud de coordenadas actuales
0	0	Salir de la función

Arduino al recibir uno de estos caracteres lo almacenará en la variable A, luego, según el carácter almacenado en esta variable se realizará la acción correspondiente. Por ello en la función MOV LIBRE CONTINUO al dar la orden de avance este lo hará de forma permanente hasta que no llegue el carácter indicando que se detenga (7) y sobrescriba a la variable A.

En caso de querer almacenar la posición en una memoria esta acción se deberá ejecutar una única vez, por lo que desde las interfaces de LabVIEW y la APP se enviará la orden mediante los caracteres 1 o 2, y la misma programación del Arduino asignará a la variable A el valor 7 luego de guardar la posición.

La solicitud de las coordenadas actuales se realizará durante todo el uso de estas funciones. Para ello el carácter P se enviará automáticamente cada 4 ciclos de la aplicación de LabVIEW (80 ms).

▪ FUNCIÓN IR A COORDENADA

En este caso, antes de enviar las coordenadas, se enviará un carácter indicando el tipo de coordenadas que se enviarán a continuación. Esto lo realizará LabVIEW de manera automática en el ciclo anterior al envío de las coordenadas, es por esto que este carácter no forma parte del String de datos de esta función (lo cual se verá más adelante). Los caracteres con los cuales se indicará el tipo de variables a enviar o bien se dará la orden de finalizar la función, serán los siguientes:

Aplicación de LabVIEW	Orden al Arduino
A	Indica que se enviarán las coordenadas en pasos
B	Indica que se enviarán las coordenadas en pasos
P	Solicitud de coordenadas actuales
0	Salir de la función

Aquí la solicitud de las coordenadas se realizará únicamente mientras el posicionador se desplaza de una posición a otra. Para ello el carácter P se enviará automáticamente cada 4 ciclos de la aplicación de LabVIEW (80 ms).

▪ FUNCIÓN MEDICIÓN AUTOMÁTICA

En este caso las únicas órdenes que se enviarán y los caracteres que las representan serán los indicados a continuación:

Aplicación de LabVIEW	Orden al Arduino
I	Ordena iniciar las mediciones con los datos anteriormente enviados
M	Indica que se van a modificar los datos enviando una nueva cadena
0	Salir de la función

20.1.2 DATOS

Las órdenes indicarán al Arduino que debe realizar una acción en particular, pero en ocasiones para poder realizarlas Arduino necesitará información que deberá ser ingresada por el usuario, como por ej., las coordenadas deseadas en la función IR A COORDENADAS. Esta información por lo tanto no será otra cosa que datos numéricos, los cuales serán enviados hacia el Arduino en forma de cadena de caracteres o Strings.

Esto implica a su vez, que los datos solo podrán ser enviados al Arduino desde la aplicación de LabVIEW, ya que solo esta interfaz permite el ingreso de datos numéricos

Arduino necesitará la recepción de estos datos en 3 momentos, al inicio para indicarle las variables de ajuste, al ejecutarse la función MEDICIÓN AUTOMÁTICA para indicarle los números de mediciones, distancias, etc, y al ejecutarse la función IR A COORDENADAS para indicarle las coordenadas deseadas. Por ello, existirán 3 tipos de String según los datos que se vayan a enviar.

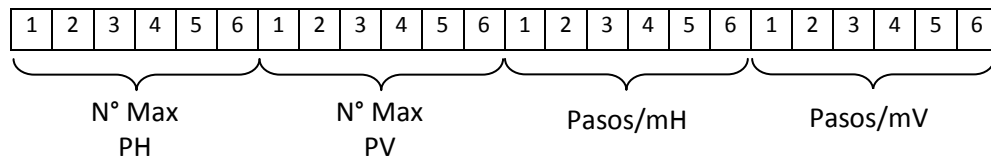
Los Strings estarán compuestos entonces por las variables que necesitará el Arduino según el momento. Los valores de estas se ubicarán uno a continuación del otro dentro del String y la extensión en número de caracteres de cada variable será constante. Así, según la cadena que se espera recibir, Arduino sabrá de cuantos elementos se compone la misma y en que ubicación dentro de esta se encuentra cada variable. De esta manera se podrá fragmentar la cadena y separar los valores guardándolos a cada uno en su variable correspondiente.



Los tres tipos de Strings que se enviarán desde la aplicación de LabVIEW hacia el Arduino serán los siguientes:

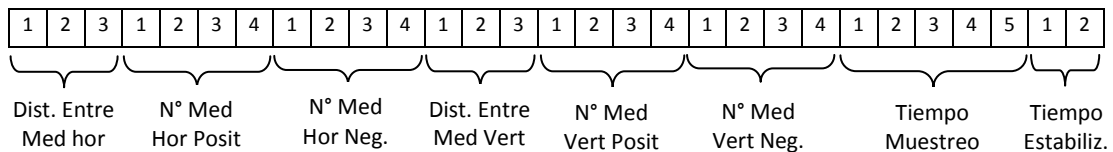
▪ STRING DE VARIABLES DE AJUSTE

Esta cadena de datos se enviará al oprimir el botón INICIO. La pista estará compuesta por 4 variables distribuidas en 24 caracteres, siendo la posición y extensión de cada variable la indicada a continuación:



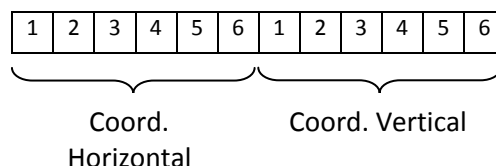
▪ STRING DE DATOS PARA FUNCIÓN MEDICIÓN AUTOMÁTICA

Esta cadena de datos se enviará al oprimir el botón ENVIAR VALORES. La pista estará compuesta por 8 variables distribuidas en 29 caracteres, siendo la posición y extensión de cada variable la indicada a continuación:



▪ STRING DE DATOS PARA FUNCIÓN IR A COORDENADA

Esta cadena de datos se enviará al oprimir el botón IR. La pista estará compuesta por 2 variables distribuidas en 12 caracteres, siendo la posición y extensión de cada variable la indicada a continuación:



En caso de que las coordenadas sean ingresadas en mm, se enviarán los valores ingresados multiplicados por 10.

20.2 DESDE ARDUINO HACIA LAS INTERFACES

En este caso la información enviada desde Arduino a las interfaces podrán ser confirmaciones e indicaciones o datos.



20.2.1 CONFIRMACIONES E INDICACIONES

Las confirmaciones serán enviadas por Arduino a las interfaces luego de que el mismo haya ejecutado o se encuentre en ejecución una orden recibida desde estas últimas. Una indicación en cambio avisa a la interfaz sobre una situación particular, como por ej, que se ha alcanzado uno de los límites de la zona de trabajo mientras se está utilizando la función MOVIMIENTO LIBRE, o que la programación se encuentra en espera de recibir órdenes o datos dentro de una función. Luego, la interfaces utilizarán estas confirmaciones o indicaciones para bloquear, desbloquear y cambiar de color los botones de las mismas según cual sea el caso. Esto, además de servir didácticamente, impide que se opriman botones que no pertenecen a la función en ejecución, evitando así que se envíen caracteres hacia el Arduino que generen inconvenientes.

Si bien en el caso de una orden el bloqueo o desbloqueo de botones puede realizarse directamente mediante la programación interna de cada interfaz según la orden que se envíe, el hacerlo recién luego de recibir una confirmación proveniente desde el Arduino sirve para confirmar que la orden enviada fue recibida correctamente por el Arduino y que la acción ordenada se ha realizado o se encuentra en ejecución.

Tanto las confirmaciones como las indicaciones serán enviadas a la App mediante un único carácter, y a la aplicación de LabVIEW mediante una cadena de caracteres de extensión constante. En el caso de las cadenas enviadas a la aplicación de LabVIEW todas serán precedidas por el símbolo @, lo que indicará que a continuación se envía una orden o indicación. Luego, la orden o indicación en particular estará representada por un numero encerrado entre letras iguales, siendo estos caracteres 'F' si se trata de una orden respecto a la ejecución de funciones, 'L' si es una orden proveniente de la función MOVIMIENTO LIBRE, 'M' si fuera de la función MEDICIÓN AUTOMÁTICA o 'I' si fuera de la función IR A COORDENADAS.

▪ INICIO DE EJECUCIÓN DE LAS FUNCIONES:

Una vez Arduino haya recibido el carácter indicando la función que el usuario desea ejecutar y la misma ya se encuentre en uso, o bien se haya finalizado una función que se encontraba en ejecución, este enviará los siguientes caracteres o cadena de ellos a modo de confirmación o indicación:

Aplicación de LabVIEW	App	Confirmación o indicación hacia las interfaces
@F1F	1	Confirma la ejecución de la función DETERMINAR CERO
@F2F	2	Confirma la ejecución de la función MOVIMIENTO LIBRE CONTINUO
@F3F	3	Confirma la ejecución de la función MOVIMIENTO LIBRE ESCALONADO
@F4F	4	Confirma la ejecución de la función IR A MEMORIA 1
@F5F	4	Confirma la ejecución de la función IR A MEMORIA 2
@F6F	4	Confirma la ejecución de la función MEDICIÓN AUTOMÁTICA
@F7F	4	Confirma la ejecución de la función IR A COORDENADA

@F0F	0	Indica que no se encuentra en ejecución ninguna función y que se está trabajando con el SRG
@F8F	8	Indica que no se encuentra en ejecución ninguna función y que se está trabajando con el SRI

■ FUNCIONES MOVIMIENTO LIBRE

En este caso Arduino enviará a la aplicación de LabVIEW indicaciones cuando el posicionador se encuentre en alguno de los límites de la zona de trabajo, o confirmaciones en caso de que el usuario haya ordenado guardar una posición en una memoria. Para esto se enviará un dato compuesto por 6 caracteres (4 para los límites y 2 para las memorias) pudiendo estos tomar valor 0 o 1 según lo que se desee indicar o confirmar. En el caso de la APP, a está solo se le confirmará el guardado de la posición en las memoria.

Aplicación de LabVIEW	App	Confirmación o indicación hacia las interfaces
@L X ₁ X ₂ X ₃ X ₄ X ₅ X ₆ L	-	Cadena compuesta por caracteres X ¹ donde cada uno de estos tomará valor 0 o 1 según sea el caso. X ₁ = 1 Indica que el posicionador se encuentra en el límite vertical superior X ₂ = 1 Indica que el posicionador se encuentra en el límite vertical inferior X ₃ = 1 Indica que el posicionador se encuentra en el límite horizontal superior X ₄ = 1 Indica que el posicionador se encuentra en el límite horizontal inferior X ₅ = 1 Confirma el guardado de la posición en la Memoria 1 X ₆ = 1 Confirma el guardado de la posición en la Memoria 2
-	1	Confirma el guardado de la posición en la Memoria 1
-	2	Confirma el guardado de la posición en la Memoria 1

Cuando el usuario haya ordenado avanzar en alguna dirección, la confirmación de esto se deberá realizar observando la modificación de las coordenadas del posicionador en el cuadro POSICIÓN.

¹ Aquí se indica a las variables como X₁, X₂, etc, solo como simplificación ya que en la programación real poseen nombres más extensos.

▪ FUNCIÓN MEDICIÓN AUTOMÁTICA

Estándose ejecutando esta función se enviarán las siguientes confirmaciones o indicaciones para saber en qué parte de la función se encuentra la programación:

Aplicación de LabVIEW	Confirmación o indicación hacia la interfaz
@M1M	Indica que la función se encuentra en espera de los datos para la medición automática
@M2M	Indica que no se produjeron desbordes y la función se encuentra en espera de la orden de iniciar las mediciones, modificar los valores o salir de la función
@M3M	Indica que se produjeron desbordes y la función se encuentra en espera de la orden de modificar los valores o salir de la función
@M4M	Confirma que las mediciones automáticas se encuentran en proceso
@M5M	Indica el inicio del muestreo en un punto
@M6M	Indica el final del muestreo en un punto
@M7M	Indica que las mediciones automáticas finalizaron
@M8M	Indica que la función MEDICIÓN AUTOMÁTICA finalizó

▪ FUNCIÓN IR A COORDENADAS

En esta función solo se indicará o confirmará las dos siguientes acciones:

Aplicación de LabVIEW	Confirmación o indicación hacia la interfaz
@I1I	Indica que la función se encuentra en espera de las coordenadas a las cuales ir o la orden de salir de la función
@I2I	Confirma que el posicionador se está dirigiendo a las coordenadas solicitadas

20.2.2 DATOS

Así como el envío de datos desde la aplicación de LabVIEW a Arduino, el envío de datos desde el Arduino se realizará únicamente con destino a esta interfaz.

A diferencia de las cadenas de datos enviadas desde la aplicación de LabVIEW hacia el Arduino, las cuales poseerán una extensión en caracteres constante, la longitud de las cadenas de datos enviadas



desde Arduino no tendrán un número de elementos fijo, sino que variará según los datos que se envíen, por ej, en caso de ser datos numéricos dependerá de cuantas cifras posean los mismos.

Por ello será necesario indicar dentro de la cadena el inicio y el final de cada dato enviado. Para ello cada dato será encerrado entre caracteres especiales para indicar dónde empieza y termina el mismo (similar a las cadenas de órdenes o indicaciones). A su vez toda la cadena será precedida por otro carácter especial el cual indicará que tipo de información contiene la cadena que se enviará luego. Así, según cual sea el carácter inicial, LabVIEW podrá clasificar la cadena y guardarla en el lugar que corresponda.

▪ VARIABLES DE AJUSTE

Esta cadena se enviara al inicio del programa en el void setup () luego de recibir las variables de ajuste. El envío es para confirmar que los valores ingresados por el usuario son los realmente utilizados en el programa.

Esta cadena contendrá 7 datos y tendrá la siguiente forma:

'?' 'A' ... 'A' 'B' ... 'B' 'C' ... 'C' 'D' ... 'D' 'G' ... 'G' 'H' ... 'H'

Donde cada (...) indica una variable numérica. Así, los datos enviados entre cada par de caracteres indicaran lo siguiente:

'?'	Caracter inicial que indica el tipo de datos contenido en la cadena
'A' ... 'A'	Nº Max PH
'B' ... 'B'	Nº Max PH
'C' ... 'C'	Pasos/mH
'D' ... 'D'	Pasos/mV
'G' ... 'G'	Coordenada horizontal máxima en mm
'H' ... 'H'	Coordenada vertical máxima en mm

Luego desde la aplicación se separará cada dato para mostrarlo en su indicador correspondiente

▪ FUNCIÓN MEDICIÓN AUTOMÁTICA

Al enviarse los valores desde la interfaz hacia el Arduino este los recibirá y procesará. Luego enviará a la aplicación una cadena de datos confirmando los valores recibidos y agregando otros valores calculados. Esta cadena contendrá 14 datos y tendrá la siguiente forma:

'/' 'A' ... 'A' 'B' ... 'B' 'C' ... 'C' 'D' ... 'D' 'E' ... 'E' 'G' ... 'G' 'H' ... 'H' 'I' ... 'I' 'J' ... 'J' 'K' ... 'K' 'N' ... 'N' 'O' ... 'O' 'Q' ... 'Q' 'R' ... 'R'

Donde cada (...) indica una variable numérica. Así, los datos enviados entre cada par de caracteres indicaran lo siguiente:

'/'	Caracter inicial que indica el tipo de datos contenido en la cadena
'A' ... 'A'	Se envían 4 caracteres que pueden tomar valor '0' o '1'. Un '1' indicaría que existe un desborde y un '0' que no lo existe. El 1° de estos caracteres corresponde al desborde horizontal positivo, el 2° al horizontal negativo, el 3° al vertical positivo y el 4° al vertical negativo
'B' ... 'B'	Distancia deseada entre mediciones horizontales
'C' ... 'C'	Número de mediciones horizontales positivas
'D' ... 'D'	Número de mediciones horizontales negativas
'E' ... 'E'	Distancia deseada entre mediciones verticales
'G' ... 'G'	Número de mediciones verticales positivas
'H' ... 'H'	Número de mediciones verticales negativas
'I' ... 'I'	Tiempo de muestreo
'J' ... 'J'	Distancia real entre mediciones horizontales
'K' ... 'K'	Cantidad de pasos entre mediciones horizontales
'N' ... 'N'	Distancia real entre mediciones verticales
'O' ... 'O'	Cantidad de pasos entre mediciones verticales
'Q' ... 'Q'	Duración total del proceso de medición automática
'R' ... 'R'	Tiempo de estabilización

Durante el proceso de medición, Arduino enviará cadenas en las cuales estarán contenidos las coordenadas del punto actual de medición y el porcentaje de realización del proceso. Estos valores se mostrarán uno a continuación del otro en el indicador PROCESO, por lo que la cadena estará compuesta de un solo dato el cual se enviará desde en Arduino ya con el formato definitivo para ser visualizado. Si bien la cadena no requerirá ser dividida, si se deberá indicarse el inicio y final del dato. Esta cadena tendrá la siguiente forma:

'Z' 'Z' ... 'Z'

Donde (...) indica una oración compuesta tanto por texto como por variables numéricas. Así, los datos enviados en esta cadena indicaran lo siguiente:

'Z'	Caracter inicial que indica el tipo de datos contenido en la cadena
'Z' ... 'Z'	Coordenada del punto de medición respecto del SRR y porcentaje de realización del proceso. Este dato tendrá la forma: "(X ; Y) 100%"

▪ FUNCIÓN IR A COORDENADA

En caso de que la coordenada ingresada por el usuario se encuentre fuera de la zona de trabajo, Arduino enviara un texto indicando que la coordenada esta fuera de rango y cuál es el rango de valores



que puede ingresarse en cada coordenada, ya sea que se esté trabajando en pasos o mm. Todo este texto se trabajará como un único dato, el cual se mostrará en pantalla mediante una ventana emergente. La cadena en este caso tendrá la siguiente forma:

'+' 'X' ... 'X'

Donde (...) indica una oración compuesta tanto por texto como por variables numéricas. Así, los datos enviados en esta cadena indicaran lo siguiente:

'+'	Caracter inicial que indica el tipo de datos contenido en la cadena
'X' ... 'X'	Texto a mostrar en la ventana emergente. El mismo tendrá la forma: "Coordenadas fuera de rango. Ingrese valores Horizontales entre (0 - ...) pasos y valores Vertical entre (0 - ...) pasos" o bien, "Coordenadas fuera de rango. Ingrese valores Horizontales entre (0 - ...) mm y valores Vertical entre (0 - ...) mm"

▪ POSICIÓN

Esta cadena se enviará desde la función ImprimirPosicion () al final de la programación. La misma contendrá 2 valores y tendrá la siguiente forma:

'#' 'A'...'A' 'B'...'B'

Donde cada (...) indica una variable numérica. Así, los datos enviados entre cada par de caracteres indicarán lo siguiente:

'#'	Caracter inicial que indica el tipo de datos contenido en la cadena
'A' ... 'A'	Coordenada vertical en pasos
'B' ... 'B'	Coordenada horizontal en pasos

Solamente se enviarán las coordenadas en pasos, siendo LabVIEW luego la encargada de convertir estas en mm para su visualización. Esto se realiza de esta manera para reducir la duración del envío de coordenadas ya que, como esta esta se realiza durante el avance de los motores, el pequeño tiempo que implica el envío de estos datos es suficiente para retrasar el tiempo entre dos pasos consecutivos del motor, lo cual puede apreciarse en el sonido que emiten los motores como una leve vibración u oscilación.

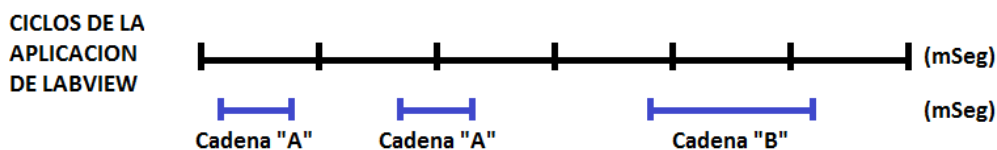


21. DURACIÓN DEL ENVÍO DE CADERAS DE INFORMACIÓN

Al enviarse cadenas o String de datos, ordenas, confirmaciones o indicaciones entre la aplicación de LabVIEW y el Arduino, estas deben recibirse completamente por el elemento receptor para poder ser utilizadas, ya que si esta información al momento de ser procesada estuviera incompleta se producirían errores o mal funcionamiento de sistema.

En el caso de las órdenes enviadas desde la aplicación de LabVIEW a Arduino, estas no presentan problemas ya que las mismas están compuestas por un único carácter. En el caso de las cadenas de datos esto tampoco conlleva inconvenientes, ya que como se indicó anteriormente, Arduino sabe los momentos en los cuales debe recibir cada tipo de dato y la extensión de los mismos. Por ello, cuando llegado dicho momento la programación simplemente espera que el dato llegue por completo contando para ello la cantidad de bytes recibidos por el puerto serie, para recién luego continuar con la ejecución del sketch y procesar dicho dato.

Por otro lado, tanto los datos como las confirmaciones e indicaciones son enviados desde Arduino a LabVIEW mediante cadenas, las cuales según el tipo que sean poseerán extensión constante o variable. A su vez, la programación de la aplicación de LabVIEW está formada principalmente por un bucle que se ejecuta continuamente sin interrupciones. Es por esto que la recepción de una cadena por puerto serie puede producirse enteramente dentro de un mismo ciclo de la aplicación, o estar dividido entre dos o tres ciclos, como se muestra a continuación.



Puede darse el caso de que la extensión de la cadena sea corta y por ello el tiempo de recepción de la misma sea menor al tiempo de duración de cada ciclo de la aplicación, es el caso de las cadenas tipo "A" indicados en el gráfico. Estas cadenas podrían ser recibidas enteramente y procesadas dentro de un mismo ciclo con lo cual no habría inconvenientes ya que se procesaría la cadena completa. En cambio, podría darse el caso en el cual una parte de la cadena fuera recibida al final de un ciclo de la aplicación y la parte restante de la cadena en el ciclo siguiente, aun cuando la cadena posea solo un par de caracteres de extensión. Esto produciría errores en la aplicación ya que en ninguno de los dos ciclos la cadena se encontraría completa para su procesamiento.

Podría también darse el caso en que la extensión de la cadena sea lo suficientemente larga como para que el tiempo de recepción de la misma sea mayor a la duración de cada ciclo de la aplicación. Aquí el procesamiento de la cadena nunca podría ser correcto ya que esta nunca se encontraría completa durante alguno de los ciclo. En este caso, suponiendo que la duración de la recepción de la cadena sea mayor a un ciclo de la aplicación y menor a dos ciclos, la misma podría quedar dividida entre dos o tres ciclos como se indica en el grafico para la cadena tipo "B".

Para resolver este inconveniente, luego de recibirse el caracter inicial de cada cadena, se deberá almacenar en una memoria todo lo recibido por el puerto serie durante el ciclo en curso y al mismo se deberá añadir luego lo recibido en el ciclo siguiente (o los ciclos siguientes). De esta manera se asegura que la cadena completa sea almacenada en la memoria para recién luego ser procesada desde allí. A su vez, estas memorias mantendrán almacenadas las cadenas o se vaciaran según sea necesario para el sistema.

Para almacenar estas cadenas recibidas desde el puerto serie se utilizarán como memorias los Shift Registers de LabVIEW. Existirá uno de estos por cada tipo de dato, confirmación o indicación. Estos, al

recibir el carácter inicial de la cadena vaciarán la memoria correspondiente y almacenarán la información recibida desde el puerto serie junto con la recibida en el o los ciclos siguientes.

Para saber cuántos ciclos deben almacenarse en cada memoria debe determinarse primeramente la extensión máxima en número de bytes de cada tipo de dato, confirmación o indicación y el tiempo correspondiente que tardaría el mismo en ser recibido.

21.1 CARACTERÍSTICAS DE LA FUNCIÓN SERIAL.PRINT()

Para enviar las cadenas de datos o String desde Arduino a LabVIEW se utilizará la función `Serial.print(Dato, Formato)`, la cual imprime los datos por puerto serie como texto ASCII. De esta forma los números se imprimen dígito por dígito mediante su correspondiente valor ASCII. Así, un número de 4 dígitos se imprimirá mediante un paquete de 4 bytes donde cada uno de estos contendrá el valor ASCII del dígito correspondiente. El parámetro Formato de la función se deja vacío, ya que por defecto se envían los valores numéricos en formato decimal (DEC). Las Variables tipo float se imprimen también en este caso dígito por dígito, incluyendo la coma y dos decimales.

De la misma manera al enviar texto mediante esta función, el mismo se envía carácter por carácter, debiendo para ello estar el texto encerrado entre comillas. En el caso de que el texto estuviera compuesto por un único carácter, el mismo deberá encerrarse entre comillas simples ('), mientras que si el mismo estuviera conformado por dos o más caracteres deberá estar encerrado entre comillas inglesas ("). En ambos casos estas comillas no se transmitirán por el puerto serie.

21.2 VELOCIDAD DE TRANSMISIÓN DE BYTES DE INFORMACIÓN Y DURACIÓN DE CICLOS DE LabVIEW

El protocolo serial es la forma o cantidad de bits necesarios para transmitir un byte de información.

Para la comunicación del Arduino con la aplicación de LabVIEW se utilizará una velocidad de transmisión de 19200 Baudios (Bits por segundos), mientras que con la APP se utilizará una velocidad de 9600 Baudios.

Teniendo en cuenta que la trama de bits para transmitir cada byte de información de Arduino a LabVIEW es de la forma:

- 1 bit de inicio
- 8 bites de información
- 0 bit de paridad
- 1 bit de finalización

Por ello, cada byte de información enviado desde Arduino estará compuesto por 10 bits. Por lo que, con una velocidad de transmisión de 19200 bites por segundo, el tiempo de envío de cada byte de información a la aplicación de LabVIEW será de:

$$T_{Byte} = \frac{1 \text{ seg}}{19200 \text{ bit}} * 10 \text{ bit} = 0,521 \text{ mSeg}$$

Por su parte, la aplicación de LabVIEW está formada principalmente por un bucle que se ejecuta cada 20mSeg. Este tiempo se adopta ya que es lo suficientemente corto como para reducir las pausas entre acciones realizadas dentro de la aplicación.



21.3 DURACIÓN DE TRANSMISIÓN DE CADENAS

■ CONFIRMACIONES O INDICACIONES

Cada una de estas cadenas poseerán extensión constante y siempre serán precedidas por el símbolo '@'. De los diferentes tipos de cadenas de confirmaciones o indicaciones las más extensas de estas tendrán la forma "@LX1 X2 X3 X4 X5 X6L" y serán enviadas durante las funciones de movimiento libre, donde cada variable X será un valor booleano y por lo tanto solo podrá tomar valor 0 o 1 (aquí se indica a las variables como X1, X2, etc, solo como simplificación ya que en la programación real poseen nombres más extensos). Esta cadena poseerá una extensión de 9 bytes de información (está compuesta por 9 caracteres). Luego, el tiempo de transmisión máxima para una confirmación o indicación será de:

$$T = 0,521 \frac{mSeg}{Byte} \cdot 9 Bytes = 4,7 mSeg$$

Este tiempo es menor a un ciclo de 20 mSeg de la aplicación de LabView, por lo que la cadena podría repartirse, a lo sumo, entre 2 ciclos del mismo. De allí que la aplicación de LabVIEW al detectar el carácter inicial '@' recibido desde el puerto serie, guardaría en el Shift Register correspondiente todo lo recibido durante el ciclo en curso y durante el ciclo siguiente.

■ VARIABLES DE AJUSTE

Estas cadenas como se vio tendrán la siguiente forma:

'?' 'A' ... 'A' 'B' ... 'B' 'C' ... 'C' 'D' ... 'D' 'E' ... 'E' 'G' ... 'G' 'H' ... 'H'

Donde cada (...) indica una variable numérica. Las primeras 5 de estas se ingresarán desde la aplicación de LabVIEW mediante los controles correspondientes. Luego los valores máximos que puedan tomar estas variables dependen de los valores máximos que puedan ingresarse en cada control (configurado en la programación de la aplicación). Por otra parte las últimas 2 variables son función de las anteriores.

La extensión máxima de la cadena se da para los valores máximos indicados en la siguiente tabla:

Cadena		Valor máximo	N° Bytes de información
'?'	Caracter inicial	?	1
'A' ... 'A'	N° Max PH	A999999A	8
'B' ... 'B'	N° Max PV	B999999B	8
'C' ... 'C'	Pasos/mH	C400000C	8
'D' ... 'D'	Pasos/mV	D400000D	8
'G' ... 'G'	Coordenada horizontal máxima en mm	G999999000G	11
'H' ... 'H'	Coordenada vertical máxima en mm	H999999000H	11
Extensión Total máxima de la cadena			55



Luego, la duración máxima para el envío de este tipo de cadenas será:

$$T = 0,521 \frac{mSeg}{Byte} \cdot 55 Bytes = 28,7 mSeg$$

Este tiempo es mayor a un ciclo de 20 mSeg de la aplicación de LabVIEW y menor a dos ciclos, por lo que la cadena podría repartirse, a lo sumo, entre tres ciclos de la misma. De allí que la aplicación de LabVIEW al detectar el carácter inicial '?' recibido desde el puerto serie, guardaría en el Shift Register correspondiente todo lo recibido durante el ciclo en curso y los dos ciclos siguiente.

■ DATOS FUNCIÓN MEDICIÓN AUTOMÁTICA

Estas cadenas tendrán la siguiente forma:

'/' 'A'...'A' 'B'...'B' 'C'...'C' 'D'...'D' 'E'...'E' 'G'...'G' 'H'...'H' 'I'...'I' 'J'...'J' 'K'...'K' 'N'...'N' 'O'...'O' 'Q'...'Q' 'R'...'R'

Donde cada (...) indica una variable numérica. Algunas de estas variables se ingresarán desde la aplicación de LabVIEW mediante los controles correspondientes, por lo tanto los valores máximos que puedan tomar estas variables dependerán de los valores máximos que puedan ingresarse en cada control. Otras variables tomarán su valor máximo según los valores ingresados en las anteriores.

La extensión máxima de la cadena se da para los valores máximos indicados en la siguiente tabla:

Cadena		Valor máximo	N° Bytes de información
'/'	Caracter inicial	/	1
'A' ... 'A'	Desbordes	A1111A	6
'B' ... 'B'	Distancia entre mediciones horizontales	B250B	5
'C' ... 'C'	Número de med. horizontales positivas	C2000C	6
'D' ... 'D'	Número de med. horizontales negativas	D2000D	6
'E' ... 'E'	Distancia entre mediciones verticales	E250E	5
'G' ... 'G'	Número de med. verticales positivas	G2000G	6
'H' ... 'H'	Número de med. verticales negativas	H2000H	6
'I' ... 'I'	Tiempo de muestreo	I60000I	7
'J' ... 'J'	Distancia real entre med. Horizontales	J250,02J	8
'K' ... 'K'	Cantidad de pasos entre med. Horizontal	K100000K	8
'N' ... 'N'	Distancia real entre med. Verticales	N250,02N	8
'O' ... 'O'	Cantidad de pasos entre med. Verticales	O100000O	8
'Q' ... 'Q'	Duración total del proceso de med. autom.	Q74965469Q	10
'R' ... 'R'	Tiempo de estabilización	R60R	4

Extensión Total máxima de la cadena		94
-------------------------------------	--	----

Luego, la duración máxima para el envío de este tipo de cadenas será:

$$T = 0,521 \frac{mSeg}{Byte} \cdot 94 Bytes = 49,0 mSeg$$

Este tiempo es mayor a dos ciclos de 20 mSeg de la aplicación de LabView y menor a tres ciclos, por lo que la cadena podría repartirse, a lo sumo, entre cuatro ciclos de la misma. De allí que la aplicación de LabVIEW al detectar el carácter inicial '/' recibido desde el puerto serie, guardaría en el Shift Register correspondiente todo lo recibido durante el ciclo en curso y los tres ciclos siguientes.

▪ PROCESO DE MEDICIÓN AUTOMÁTICA

Estas cadenas tendrán la siguiente forma:

$$"\text{¿Z(X ; Y) 100\%Z}"$$

Donde X e Y son las coordenadas relativas del punto de medición. Por lo tanto la extensión máxima de esta cadena se dará cuando la misma tenga la forma " $\text{¿Z(-2000 ; -2000) 100\%Z}$ ". Es decir su extensión máxima será de 25 bytes (contando los espacios). Luego, la duración máxima para el envío de la misma será:

$$T = 0,521 \frac{mSeg}{Byte} \cdot 25 Bytes = 13,1 mSeg$$

Este tiempo es menor a un ciclo de 20 mSeg de la aplicación de LabView, por lo que la cadena podría repartirse, a lo sumo, entre dos ciclos del mismo. De allí que la aplicación de LabVIEW al detectar el carácter inicial '¿' recibido desde el puerto serie, guardaría en el Shift Register correspondiente todo lo recibido durante el ciclo en curso y durante el ciclo siguiente.

▪ FUNCIÓN IR A COORDENADA

Estas cadenas tendrán la siguiente forma:

$$'+' 'X' \dots 'X'$$

Donde, encerrado entre los caracteres 'X' se ubicará el mensaje "Coordenadas fuera de rango. Ingrese valores Horizontales entre (0 - ...) y valores Verticales entre (0 - ...)". Por lo tanto la mayor extensión de esta cadena se producirá cuando la misma tenga la forma "+XCoordenadas fuera de rango. Ingrese valores Horizontales entre (0 - 999999) pasos y valores Verticales entre (0 - 999999) pasos". Siendo 999999 el número máximo de pasos que puede realizar el posicionador, lo que queda determinado por el valor máximo que pueden ingresarse en los controles correspondientes de la aplicación. En este caso la extensión máxima será de 129 bytes (contando los espacios). Luego, la duración máxima para el envío de la misma será:

$$T = 0,521 \frac{mSeg}{Byte} \cdot 129 Bytes = 67,2 mSeg$$



Este tiempo es mayor a tres ciclos de 20 mSeg de la aplicación de LabView y menor a cuatro ciclos, por lo que la cadena podría repartirse, a lo sumo, entre cinco ciclos del mismo. De allí que la aplicación de LabVIEW al detectar el carácter inicial '+' recibido desde el puerto serie, guardaría en el Shift Register correspondiente todo lo recibido durante el ciclo en curso y los cuatro ciclos siguientes.

▪ POSICIÓN

Estas cadenas tendrán la siguiente forma:

'#' 'A' ... 'A' 'B' ... 'B'

Donde cada (...) indica una variable numérica. La extensión máxima de la cadena se dará para los valores máximos indicados en la siguiente tabla:

Cadena		Valor máximo	N° Bytes de información
'#'	Caracter inicial	#	1
'A' ... 'A'	Coordenada vertical en pasos	A999999A	8
'B' ... 'B'	Coordenada horizontal en pasos	B999999B	8
Extensión Total máxima de la cadena			17

Luego, la duración máxima para el envío de este tipo de cadenas será:

$$T = 0,521 \frac{mSeg}{Byte} \cdot 17 Bytes = 8,9 mSeg$$

Este tiempo es menor a un ciclo de 20 mSeg de la aplicación de LabView por lo que, al detectar el carácter inicial '#' recibido desde el puerto serie, la misma guardaría en el Shift Register correspondiente todo lo recibido durante el ciclo en curso y durante el ciclo siguiente.



4TA PARTE

ELECTRICIDAD Y ELECTRÓNICA

22. DRIVERS, FUENTES DE ALIMENTACIÓN Y CABLEADO

22.1 DRIVERS

▪ DRIVER DEL MOTOR VERTICAL

En este caso a la velocidad de funcionamiento el motor utilizado entregaría un torque un poco menor a 2,1 Nm con una corriente bipolar de 5 A/fase, mientras que el torque requerido será de 1,12 Nm. Ya que el torque del motor será considerablemente mayor al requerido se optará por reducir la corriente en el motor para evitar resonancias y reducir el consumo de energía.

Si se tienen en cuenta por un lado que existe una relación lineal entre la corriente circulante por las bobinas y el torque estático del motor, y por otro lado que al trabajar a bajas velocidades el motor entregaría un torque próximo al torque estático, podría considerarse que el torque entregado a la velocidad de trabajo poseerá también una relación lineal con la corriente en las bobinas. Luego, si se desprecia el par de detención (por no contarse con su valor), si la corriente se redujera en un 25% a un valor de 3,75 A, el torque entregado sería un poco inferior a 1,57 Nm lo cual sería suficientemente superior al requerido.

Luego, el motor vertical trabajará con una corriente pico de 3,75 A/fase, por ello se selecciona el siguiente driver:

- Marca: LEADSHINE
- Modelo: DM542E
- Corriente máxima de salida: 4,2 A (3A RMS)
- Tensión de alimentación: 20 – 50 VDC
- Tensión de alimentación típica : 24 – 48 VDC
- Tensión de señal de pulso: 4 – 5 V para HIGH, 0 – 0,5 V para LOW

En función de los valores de regulación de corriente que permite el driver, el mismo deberá regularse en 3,76 A.

Según lo determinado en el capítulo 4.8 el torque necesario de retención deberá ser de 0,374 Nm, lo cual es menor a la mitad del torque necesario total (1,12 Nm), por ello en el driver podrá activarse la función de reducción de corriente cuando el motor está detenido, reduciendo la corriente al 50 % del valor regulado, evitando así el calentamiento innecesario del motor y reduciendo el consumo de energía. Con esta reducción de corriente se reducirá también el par en un 50 %, pero el mismo seguiría siendo superior al par de retención necesario para mantener la carga en su posición.

Las características más importantes del manual de operaciones del driver utilizado se indican en el anexo A.11.

▪ DRIVER DE LOS MOTORES HORIZONTALES

En este caso a la velocidad de funcionamiento el motor utilizado entregaría un torque un poco menor a 2,9 Nm con una corriente bipolar de 6 A/fase, mientras que el torque requerido será de 1,88 Nm. Ya que el torque del motor será considerablemente mayor al requerido se optará aquí también por reducir la corriente en el motor.



A partir de las mismas consideraciones realizadas para el motor vertical si la corriente se redujera en un 25 % a un valor de 4,5 A, el torque entregado sería un poco inferior a 2,17 Nm lo cual sería suficientemente superior al requerido.

Luego, el motor vertical trabajará con una corriente pico de 4,5 A/fase, por ello se selecciona el siguiente driver:

- Marca: LEADSHINE
- Modelo: DM556E
- Corriente máxima de salida: 5,6 A (4A RMS)
- Tensión de alimentación: 20 – 50 VDC
- Tensión de alimentación típica : 24 – 48 VDC
- Tensión de señal de pulso: 4 – 5 V para HIGH, 0 – 0,5 V para LOW

En función de los valores de regulación de corriente que permite el driver el mismo deberá regularse en 4,9 A.

Según lo determinado en el capítulo 4.9 se requerirá par de retención cuando el motor se encuentre detenido. Este par podrá ser significativamente menor al par necesario para el desplazamiento, por lo que en el driver podrá activarse la función de reducción de corriente cuando el motor se encuentre detenido.

Las características más importantes del manual de operaciones del driver utilizado se indican en el anexo A.12.

22.2 FUENTES DE ALIMENTACIÓN

▪ ALIMENTACIÓN DE DRIVER - MOTOR VERTICAL

Debido a que el motor trabajará a baja velocidad se deberá utilizar una fuente de baja tensión dentro del rango admisible del driver, evitando así la posibilidad de recalentamiento. Se utilizará en este caso fuentes conmutadas ya que fuentes lineales del valor de corriente requerido son difíciles de conseguir, a no ser que la misma sea construida a partir de sus componentes. En este caso se alimentará el driver con una fuente de 24 VDC.

Ya que el driver se configurará para entregar 3,76 A/fase, y teniendo en cuenta un aumento del 25 % por estarse trabajando con fuentes conmutadas se requerirá de una fuente de al menos 10 A.

▪ ALIMENTACIÓN DE DRIVERS - MOTORES HORIZONTALES

En este caso se utilizarán también fuentes de 24 VDC debido a que el motor trabajará a baja velocidad y las mismas serán del tipo conmutadas por su fácil comercialización.

Ya que el driver se configurará para entregar 4,9 A/fase, y teniendo en cuenta un aumento del 25 % por estarse trabajando con fuentes conmutadas se requerirá de una fuente de al menos 13 A para cada driver.



▪ ALIMENTACIÓN DEL CIRCUITO DE COMANDO

Por prolijidad se incluye aquí la selección de la fuente de alimentación del circuito de comando, aunque los elementos necesarios para su dimensionamiento se seleccionan en los capítulos siguientes.

En el caso de los contactores SIEMENS 3RT20 23-1BB40 y SCHNEIDER LC1 D09BD (seleccionados en el capítulo 24.3), los mismos disiparán en retención una potencia de 5,9 W y 5,4 W respectivamente, lo que corresponde a corrientes en el circuito de comando de 250 mA y 225 mA.

Por su parte cada sensor de posición (seleccionados en el capítulo 23.1) poseerá un consumo de 30 mA para su funcionamiento y por su circuito de salida circulará 5 mA en función de las resistencias 4K7 y 1K conectadas en serie a cada una.

En cuanto a los 3 diodos led, las resistencias conectadas en serie a los mismos se calcularon para limitar la corriente en estos a 11 mA.

Por su parte el Arduino mega tiene un consumo de más de 100mA a 12VDC.

Se tendrá así un consumo aproximado en el circuito de comando de 1A. Se selecciona por seguridad para alimentar al mismo una fuente conmutada de 24 VDC 3 A protegida con fusible de 2,5 A.

El circuito de comando se presenta en el anexo B.4.

22.3 CABLEADO

▪ TIPOS DE CABLES A UTILIZAR

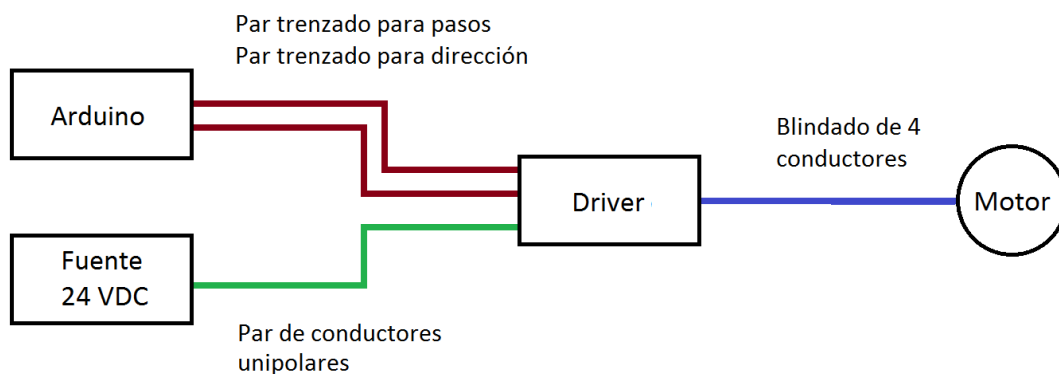
Para los cables que alimentarán los motores deberá utilizarse cables blindados para evitar que introduzcan ruido en los cables de señal, lo que podría generar errores en el posicionamiento o inestabilidad del sistema. Se utilizará en este caso conductores marca Marlew modelo XF_WNe/NBR1 (Potencia) o FX_WNe/NBR1 (Comando), ambos con blindaje mediante cinta de cobre recocido con cobertura 100 %.

Para los cables de señal (pasos y dirección) deberá utilizarse cables trenzados en pares. Se utilizará en este caso conductores marca Marlew modelo AR Miliar (instrumentación electrónica) con blindaje mediante cinta helicoidal de aluminio-poliéster y sin armadura.

Las alimentaciones de los drivers desde las fuentes DC se realizarán mediante conductores unipolares IRAM 247-3 marca Prysmian modelo Superastic Flex.

Las características de los distintos cables se adjuntan en el anexo A.13.

La configuración del cableado entre los distintos componentes será la siguiente:



▪ ALIMENTACIÓN DE MOTORES

Como los cables de alimentación de los motores horizontales soportarán una corriente regulada de 4,9 A y los del motor vertical una corriente regulada de 3,76 A, podrán utilizarse para su alimentación los siguientes conductores:

- Marca: MARLEW
- Modelo: XF_WNe/NBR1 (Potencia)
- Código: XF 0415 WNe/NBR1
- Formación y sección: 4x1,5 mm²
- Diámetro exterior: 12,3 mm
- Corriente admisible: 20A.
- Peso: 214 gr/m

O bien:

- Marca: MARLEW
- Modelo: FX_WNe/NBR1 (Comando)
- Código: FX 0510 WNe/NBR1
- Formación y sección: 5x1 mm²
- Diámetro exterior: 12,3 mm
- Corriente admisible: No se especifica, pero para conductores de 1,5 mm² es de 20A.

Peso: 211 gr/m

En el caso de los motores horizontales, estos se encontrarán sobre los ejes horizontales, por lo que ninguna parte de este cableado deberá viajar por alguna cadena portacables, por ello los drivers podrán ubicarse fuera del túnel e ingresar únicamente los conductores que alimentan a los motores.

En el caso del motor vertical este se encontrará sobre el eje vertical, por lo que una parte del cableado se viajará por la cadena portacables horizontal que conecta el eje horizontal con el vertical. Esta cadena se seleccionó en el capítulo 8.1 y poseerá un radio de curvatura de 38 mm en su centro y de aproximadamente 47 mm en la cara externa de la sección útil. Teniendo en cuenta que los cables blindados para la alimentación del motor poseen un radio de curvatura mayor al permitido por la cadena, en este caso se opta por ubicar el driver del motor en el eje vertical (detrás de la chapa de protección), de



esta manera por la cadena portacables viajarán los pares trenzados de señal y los de alimentación del driver ya que estos poseerán un menor radio de curvatura.

▪ SEÑALES

Se decide utilizar dos cables de un par en lugar de un cable de tres conductores ya que los primeros permiten un menor radio de curvatura a igual sección de los conductores, por lo que soportaría mejor los esfuerzos al montarse dentro de la cadena portacables.

Se utilizarán entonces como cables de señal 2 conductores de un par cada uno del tipo:

- Marca: MARLEW
- Modelo: AR MILIAR
- Código: AR 5200
- Formación: Par blindado
- Diámetro exterior: 6,3 mm
- Radio de curvatura (7xDiam): 44,1 mm
- Conductor: 7 hilos AWG 16
- Peso: 61 gr/m

IMPORTANTE: Al posicionador, debajo de la chapa de protección horizontal, ingresarán los conductores de alimentación de los dos motores horizontales y los conductores de alimentación y señal de los driver del motor vertical. Los conductores de alimentación de los motores horizontales y los de señal del vertical deberán mantenerse alejados en esta parte (en lo posible más de 10 cm) para evitar la inducción de ruido en estos últimos.

▪ ALIMENTACIÓN DE DRIVERS

Para su selección se tuvo en cuenta la caída de tensión en los conductores que alimentan a los drivers. El caso más desfavorable por su longitud se presentará en la alimentación del driver del motor vertical, ya que el mismo se ubicará sobre el eje vertical del posicionador.

La longitud real de los conductores quedará definida al momento de construir y montar el posicionador y su sistema de control. Por seguridad para el cálculo de caída de tensión se consideraron conductores de 10 m de longitud cada uno.

La corriente máxima a suministrar por la fuente de alimentación del driver del motor vertical será de 10 A, lo que obliga a utilizar conductores IRAM 247-3 de al menos 1 mm², los cuales soportarán más de los 11,5 A indicados en el catálogo, ya que los mismos no se encontrarán canalizados rígidamente.

Luego, para conductores de 1 mm² se tendrá una caída de tensión de:

$$\Delta V = 2 \cdot I \cdot R \cdot L = 2 \cdot 10 \text{ A} \cdot 0,0195 \frac{\text{Ohm}}{\text{m}} \cdot 10 \text{ m} = 3,9 \text{ V}$$

Por ello que se tendría una tensión de entrada al driver de (24 - 3,9) V = 20,1 V, la cual se encontraría muy próxima a los 20 V, límite inferior de la alimentación del driver. Por ello se selecciona para la alimentación de los drivers conductores de 2,5 mm² con los cuales se tendrá una caída de tensión de 1,56 V y una tensión en bornes del driver de (24 - 1,56) V = 22,44 V.



Por su parte, considerando conductores de cuerdas flexibles, sin armadura y diámetro exterior menor a 25 mm, el radio de curvatura de los mismos será de 4 veces su diámetro. Por tanto, el radio de curvatura para un conductor de $2,5 \text{ mm}^2$ con 3,6 mm de diámetro será de 22,44 mm, inferior a los 38 mm de radio de curvatura de la cadena portacables horizontal por donde viajará.



23. UBICACIÓN DEL PUNTO DE REFERENCIA

23.1 SENSORES DE POSICIÓN

Ya que estos sensores de posición tendrán como finalidad determinar la ubicación del Cero Global los mismos deberán tener una elevada precisión de manera de reducir al mínimo el error al determinar este punto en diferentes usos del posicionador.

Al estarse ejecutando la función MOVIMIENTO LIBRE mientras se trabaja con el SRI, el posicionador podrá seguir desplazándose aun luego de ser detectado por los sensores. Por otro lado, estos sensores no deberán interferir en el desplazamiento del posicionador al momento de detectarlo, evitando así colisiones con el mismo.

Por lo anterior se utilizarán como sensores de posición Barreras fotoeléctricas en horquilla modelo OPU200 de la marca IFM cuyo catálogo se adjunta en el anexo A.14. Las principales características de las mismas se indican a continuación:

- Marca y modelo: IFM OPU200
- Diámetro del objeto detectable más pequeño: 0,2 mm
- Tipo de luz: Luz infrarroja
- Tensión de alimentación: 10 a 30 VDC
- Consumo de corriente: 30 mA
- Tipo de salida: PNP/NPN
- Caída de tensión Max de la salida: 1,5V
- Corriente máxima de la salida: 100 mA

La precisión en la detección vendrá dada por el diámetro del objeto detectable más pequeño (0,2 mm) el cual como puede observarse es lo suficientemente chico para asegurar una alta exactitud en la detección del punto Cero Global.

23.2 TIPO DE CONEXIÓN

Mediante este circuito el Arduino leerá el estado de los Sensores de Posición para así determinar la posición del punto Cero Global cuando se ejecute la función DETERMINAR CERO.

Los sensores seleccionados permitirán elegir su modo de funcionamiento en modo luz (salida activada cuando el receptor detecta la luz, es decir sensor no obstruido) y modo oscuridad (salida activada cuando el receptor no detecta la luz, es decir sensor obstruido). Por otra parte el sensor permitirá alimentación PNP ó NPN. En el caso de usar alimentación NPN debería utilizarse una configuración PULL UP y para alimentación PNP configuración PULL DOWN.

Por su parte el sensor deberá alimentarse con tensiones entre los 10 y los 30 VDC, mientras que las entradas digitales del Arduino son de 5V. Esto obligará a utilizar un divisor de tensiones para adaptar el nivel de tensión de funcionamiento del sensor al nivel a tensión admitida por el Arduino.

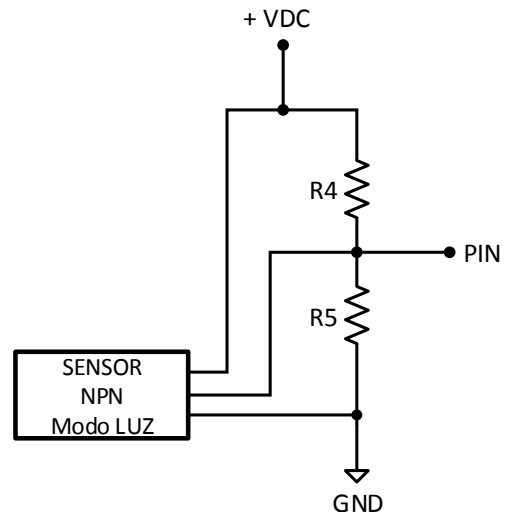
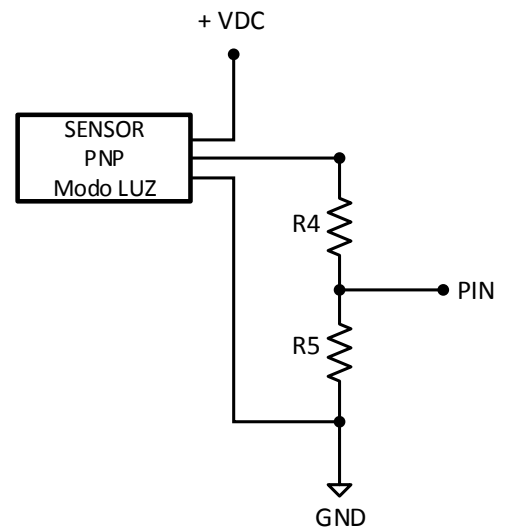
En caso de utilizar alimentación PNP el circuito PULL DOWN utilizado se muestra en la figura. Aquí, si primeramente consideramos que el sensor se encuentra en modo sombra, cuando el sensor no se encuentre obstruido (posicionador no detectando) la salida estará desactivada por lo que el pin del Arduino estará en estado BAJO (0V). Cuando el sensor este obstruido (posicionador detectado) la salida se activará, por lo que el pin del Arduino estará en estado ALTO (5V). (Divisor de tensiones compuesto por las resistencias R4 y R5). En caso de una falla en los cables que van al sensor, o en el mismo sensor, la salida del mismo no podría activarse por lo que el Arduino leería un estado BAJO (0V) y en caso de falla en la fuente el Arduino también leerá estado BAJO (0V). Esto se resume en:

- Posicionador no detectado: 0 V
- Posicionador detectado: 5 V
- Falla en cableado o sensor: 0 V
- Falla en la fuente: 0 V

Con el mismo razonamiento si el sensor trabajara en modo luz, se invertirían las tensiones vistas por el Arduino, ya que aquí al no detectar Arduino recibiría 5V y al detectar recibiría 0V. Por su parte ante los fallos seguiría recibiendo 0V en ambos casos.

En caso de utilizar alimentación NPN el circuito PULL UP utilizado se muestra en la figura. Si aquí también consideramos primeramente al sensor en modo sombra, cuando el mismo no se encuentre obstruido (posicionador no detectando) la salida estará desactivada por lo que el pin del Arduino estará en estado ALTO (5V) (Divisor de tensión compuesto por R4 y R5). Cuando el sensor este obstruido (posicionador detectado) la salida se activará, por lo que el pin del Arduino estará en estado BAJO (0V). En caso de una falla en los cables que van al sensor, o en el mismo sensor, la salida del mismo no podría activarse por lo que en este caso el Arduino leería un estado ALTO (5V). Y en caso de falla en la fuente el Arduino leerá estado BAJO (0V). Esto se resume en:

- Posicionador no detectado: 5 V
- Posicionador detectado: 0 V
- Falla en cableado o sensor: 5 V
- Falla en la fuente: 0 V



Con el mismo razonamiento si el sensor trabajara en modo luz, se invertirían las tensiones vistas por el Arduino, ya que aquí al no detectar Arduino recibiría 0V y al detectar recibiría 5V. Por su parte ante los fallos seguiría recibiendo 5V si la falla fuera en el cableado o el sensor y 0V si lo fuera en la fuente.

Resumiendo lo anterior se tiene el siguiente cuadro:

	PNP		NPN	
	Luz	Sombra	Luz	Sombra
Posic. no detectado	5	0	0	5
Posic. Detectado	0	5	5	0
Falla en cables o sensor	0	0	5	5
Falla en fuente	0	0	0	0

Como se explica en el capítulo 15.2, la programación de la función DETERMINAR CERO, leerá inicialmente el estado del sensor, si el mismo se encontrara detectando al posicionador se producirá un avance continuo hasta que finalice la detección. Si luego de avanzar determinada cantidad de pasos el sensor aún permaneciera en estado de detección se considerará que existe una falla en el circuito. Esta consideración solo podría darse si el estado de detección y ambos tipos de fallas, enviaran al Arduino la misma señal, lo que ocurrirá solamente cuando si trabajara con alimentación PNP en modo luz, como puede verse en el cuadro comparativo anterior. Por lo tanto se utilizará los sensores en modo Luz alimentados en formato PNP configuración PULL DOWN.

Debido a la gran extensión del circuito y la caída de tensión en el mismo conviene utilizar una tensión + VDC elevada. Siendo 30 V la tensión máxima soportada por el sensor se adopta utilizar una fuente de 24 VDC debido a su mayor comercialización.

A partir del diagrama indicativo de la configuración PULL DOWN la tensión recibida en el PIN del Arduino será:

$$V_{PIN} = \frac{R_5}{R_4 + R_5} \cdot 24V$$

Los pines digitales del Arduino soportan una tensión máxima de 5V, siendo la tensión umbral alrededor de los 2,5V (tensiones superiores las lee como HIGH y las inferiores como LOW). De aquí, para que la tensión V_{PIN} tome valores en el rango entre 2,5 y 5V, R_4 deberá ser entre 8,6 y 3,8 veces mayor que R_5 respectivamente¹.

Teniendo en cuenta que el sensor podrá proporcionar una corriente máxima de salida de 100 mA y siendo la fuente de alimentación de 24V, la suma de R_4 y R_5 debe ser siempre superior a 240 Ohm. Por su parte, cuanto mayor el valor de R_4 y R_5 menor será la influencia de la resistencia propia de los cables y menor la potencia disipada, lo cual es importante si se tiene en cuenta que la salida del sensor estará activa prácticamente todo el tiempo.

Se adopta entonces resistencias $R_4 = 4K7$ y $R_5 = 1K$, con las cuales se obtendría una tensión V_{PIN} de 4,2 V y una corriente de 4,2 mA.

El circuito de conexión para la detección del punto de referencia se presenta en el anexo B.4.

NOTA: En caso de alimentar este circuito con 12 VDC, R_4 debería ser entre 4,4 y 1,4 veces R_5 , pudiéndose utilizar en este caso $R_4 = 1K8$ o 2,2K y $R_5 = 1K$.

¹ La relación real entre estas resistencias no debería ser muy cercana a alguno de estos límites para que la tensión en el PIN no sea muy cercana a la tensión máxima o a la tensión umbral.



24. PROTECCIONES CONTRA COLISIONES MEDIANTE FINES DE CARRERA Y CONTRA DESENERGIZADO DE DRIVERS

24.1 PROTECCIÓN CONTRA COLISIONES MEDIANTE FINES DE CARRERA

Esta protección contra colisiones será un circuito eléctrico utilizado como mecanismo de seguridad para cortar la alimentación eléctrica de los drivers en caso de que el posicionador se aproxime demasiado a la estructura, evitando así su colisión con esta.

Para ello, entre cada driver y su respectiva fuente de alimentación se ubicará un contactor mediante el cual se alimentará o interrumpirá la alimentación al driver. A su vez, se montarán fines de carrera en cada eje del posicionador, los cuales se encargarán de energizar las bobinas de los contactores.

En el eje vertical se utilizarán dos fines de carrera, uno en el extremo superior y otro en el extremo inferior, los cuales serían accionados mediante elementos diseñados para tal fin que estarán sujetos al portainstrumentos. En el caso del eje horizontal se montará un único fin de carrera en el soporte central del eje inferior el cual se accionará mediante elementos ubicados en los soportes extremos de este eje.

Los fines de carrera poseerán contactos NC los cuales se conectarán en serie para energizar las bobinas de los contactores. Además se contará con un pulsador NO en paralelo al circuito de los fines de carrera de manera que, si algún fin de carrera fuera accionado, mediante este pulsador se pueda accionar momentáneamente los contactores y energizar los drivers para desplazar el posicionador a una ubicación segura.

Se utilizarán para esto fines de carrera marca ABB serie LS20 cuyo catálogo se adjunta en el anexo A.15. Se adoptó en este caso fines de carrera con embolo de roldana y acción lenta, cuyas características principales se indican a continuación:

- Marca y Serie: ABB Serie LS20
- Código: LS21 M12D11-P01 y LS21 M14D11-P01
- Actuador: Embolo de roldana
- Desplazamiento para accionamiento seguro: 5,9 mm
- Desplazamiento máximo: 8,7 mm
- Contactos integrados 1NA/1NC (acción lenta)
- Máxima velocidad de actuación: 0,5 m/seg

Se utilizarán dos fines de carrera modelo LS21 M12D11-P01 (dirección actuador 0°) para el eje vertical y un fin de carrera modelo LS21 M14D11-P01 (dirección actuador 90°) para el eje horizontal.

Podrían utilizarse también los fines de carrera del mismo modelo pero con ruptura brusca cuyos códigos serán LS21 M12B11-P01 y LS21 M14B11-P01. En este caso el desplazamiento para accionamiento seguro será de 6,9 mm por lo que deberá tomarse los cuidados correspondientes durante el ensamble del sistema.

24.2 PROTECCIÓN CONTRA DESENERGIZADO DE DRIVERS

En caso de que se interrumpa la alimentación eléctrica de alguno de los drivers, como por ejemplo por falla en alguna de las fuentes o contactores, el proceso debería ser detenido por seguridad, en especial si



se interrumpiera la alimentación en uno de los drivers de un motor horizontal, ya que el funcionamiento de un solo motor horizontal estando el otro detenido, podría producir roturas en algunos elementos mecánicos.

Para ello en paralelo a la alimentación eléctrica de cada driver se colocará un relé con contactos NA. Las bobinas de los tres contactores se conectarán en paralelo y se alimentarán a través de la conexión en serie de los contactos NA de estos tres relés. Así, si la alimentación de algún driver se interrumpiera también lo hará la alimentación de su relé en paralelo, produciendo por ello la apertura simultáneamente de los tres contactores.

Teniendo en cuenta que mediante estos relés se alimentará solo las bobinas de los 3 contactores podrán utilizarse relés de estado sólido de potencia o relés miniatura de potencia enchufables. Estos relés, al estar conectado en paralelo a la alimentación de los drivers, deberán accionarse con 24VDC. En este caso se utilizarán relés marca Phoenix Contact sistema Refline complete con contactos de potencia, cuyo catálogo se adjunta en el anexo A.16. Podrán utilizarse en este caso:

- Relés de estado sólido de potencia de 3A modelo OPT-24DC/24DC/2 soldados en placa de circuito impreso o montados en zócalo RIF-0 de 1 contacto NA modelo RIF-0-BSC/1.
- Relés miniatura con contactos de potencia de 6A modelo REL-MR-24DC/21 soldados en placa de circuito impreso o montados en zócalo RIF-0 de 1 contacto conmutado modelo RIF-0-BSC/21.

24.3 CONTACTORES DE ALIMENTACIÓN DE DRIVERS

Los contactores deberán alimentar eléctricamente a los Drivers de los motores paso a paso. Se instalará uno entre cada fuente de alimentación y su driver respectivo. Las fuentes de alimentación tendrán que poder entregar 10 y 13 A según lo determinado en el capítulo 22.1, por lo que los contactores deberán poder soportar estas corrientes en categoría AC-1.

En cuanto a la tensión de mando por seguridad se prefiere utilizar una tensión de 24 V. Luego, por simplicidad se utilizará la misma fuente de alimentación y cableado de los sensores de posición, por lo que la tensión de mando será de 24 VDC.

Para indicar si los contactores se encuentran abiertos y los drivers desenergizados se colocará una señal luminosa conectada a través del contacto NC de alguno de los contactores. Las bobinas de los 3 contactores se conectarán en paralelo como se indicó en el capítulo anterior, por lo tanto se podrá utilizar el contacto NC de solo uno de ellos.

Con estas características puede utilizarse por ej los siguientes modelos de contactores:

- | | | | |
|----------------------------|----------------------|------------|-------------------|
| - SIEMENS - 3RT20 15-1BB42 | Ie: 18A (AC-1 <690V) | Us: 24 VDC | Cont Aux: 1NC |
| - SCHNEIDER - LC1 D09BD | Ie: 25A (AC-1 440V) | Us: 24 VDC | Cont Aux: 1NA+1NC |

En ambos casos no se verifica el número de ciclos de maniobra por ser sumamente bajo.

El circuito de conexión de la protección contra colisiones mediante fines de carrera y la protección contra desenergizado de drivers se presenta en el anexo B.4.



5TA PARTE

OTROS



25. ANÁLISIS ECONÓMICO

25.1 COMPUTO DE MATERIALES

El diseño del posicionador se realizó teniendo en cuenta dos aspectos fundamentales:

- Facilidad para conseguir los componentes.
- Contar con datos completos de los componentes para su correcta selección.

Estas dos condiciones no siempre se cumplen simultáneamente, ya que hay marcas reconocidas que proveen toda la documentación y características de sus productos para su correcta selección, pero los mismos son difíciles de conseguir. En cambio, se consiguen con mayor facilidad productos similares pero de marcas alternativas no reconocidas, las cuales no proveen catálogos completos con las características de sus productos.

En esos casos se seleccionaron los productos de marcas reconocidas que proveen toda la información necesaria de sus productos, pero luego para el cómputo de materiales se consideraron los elementos de marcas no reconocidas pero que se consiguen con mayor facilidad. Esto se realiza para obtener una referencia aproximada del costo total del posicionador, teniendo siempre presente que los elementos de marcas alternativas serán siempre de menor costo que los de primera marca seleccionados.



MATERIALES	CANT.	UNID.	COSTO [\$]		PROVEEDOR
			UNIT	TOTAL	

ESTRUCTURA

BARRA ACERO TREFILADO SAE 1010 16 mm					
ALTERNATIVA: BARRA ACERO TREFILADO SAE 1040 17 mm	4	m	1420	5680	HIGINIO BERTONCINI Y CIA S.A.
BARRA ACERO TREFILADO SAE 1010 28 mm					
ALTERNATIVA: BARRA ACERO TREFILADO SAE 1040 1 1/8"	10	m	4060	40600	HIGINIO BERTONCINI Y CIA S.A.
BARRA DE GRILON 50 X 50 X 1000 mm	1	un	46170	46170	INSUMOS CAC
PLANCHA DE POLICARBONATO COMPACTO 4mm ESPESOR 2400 X1200 mm	1	un			
ALTERNATIVA: PLANCHA DE POLICARBONATO COMPACTO 4mm ESPESOR 3000 X 2000 mm	1	un	63728	63728	ALUMAR SRL
PLANCHUELA ALUMINIO 100 x 10 x 3000 mm	1	un	42963	42963	ALUMINA ARGENTINA SRL
PLANCHUELA ALUMINIO 80 x 10 x 3000 mm	1	un	34360	34360	ALUMINA ARGENTINA SRL
PLANCHUELA ALUMINIO 12,7 x 5 x 6000 mm	1	un	5327	5327	ALUMINA ARGENTINA SRL
PLANCHUELA ALUMINIO 20 x 15 x 1000 mm	1	un			
ALTERNATIVA: BARRA CUADRADA 19,05 x 1000 mm	1	un	6065	6065	ALUMINA ARGENTINA SRL
PERFIL ANGULO 25x1,2 mm ALUMINIO	1	un	2015	2015	ALUMAR SRL
CADENA PORTACABLES vertical Fesma 18.37.28 (util 37x18 total 51x23)	1	m			LINEALTEC
ALTERNATIVA: TECMOTION 37x18 R48	1	m	9800	9800	CNC INSUMOS SRL
CADENA PORTACABLES horizontal Fesma 18.37.38 (util 37x18 total 51x23)	1.3	m			LINEALTEC
ALTERNATIVA: TECMOTION 37x18 R48	1.3	m	9800	12740	CNC INSUMOS SRL
CARROS ROMA 164-A	8	un	1060	8474	GRAMABI SRL
RIEL PORTACARRO ROMA PLANO 164 x 6 m	2	un	6069	12138	GRAMABI SRL
MANGUERA PVC TRANSPARENTE 6 mm EXT 3 mm INT	4	m	50	200	
OTROS (Soportes, buloneria, ejes, etc)	1	un	45000	45000	



MOTOR Y TRANSMISIÓN

MOTOR NEMA 23 2,1 Nm LEADSHINE D57CM21	1 un	11535	11535	ELINDAR SRL
MOTOR NEMA 23 3,1 Nm LEADSHINE D57CM31	2 un	15450	30900	ELINDAR SRL
REDUCTOR PLANETARIO 5:1 NEMA 23 MICRON NemaTRUE NT23-005	1 un			
ALTERNATIVA: TECMOTION RPL23	1 un	90320	90320	CNC INSUMOS SRL
REDUCTOR PLANETARIO 10:1 NEMA 23 MICRON NemaTRUE NT23-010	2 un			
ALTERNATIVA: TECMOTION RPL23	2 un	90320	180640	CNC INSUMOS SRL
ACOPLAMIENTO COMINTEC GAS/SG ST 00 M1-M1 8 B1 10 B1 SG 64 Sh-D	2 un			
ALTERNATIVA: TECMOTION 2030 8x10 mm	2 un	4230	8460	CNC INSUMOS SRL
CORREA SINCRONA VERTICAL ABIERTA GATES POWER GRIP GT3-5MGT-15	3.5 m			
ALTERNATIVA: GATES ABIERTA HTD LL-5M-15	4 m	3696	14783	JUAN CARLOS MILESI
CORREA SINCRONA HORIZONTAL ABIERTA GATES POWER GRIP GT4-8MGT	10 m			
ALTERNATIVA: GATES SIN FIN HTD 2600-8MGT-20	4 un	11568	46272	JUAN CARLOS MILESI
POLEA SINCRONA VERTICAL GT3 24-5MR-15-6F ACERO	2 un			
ALTERNATIVA: HTD 24-5M-15-6F ACERO	2 un	4482	8964	JUAN CARLOS MILESI
POLEA SINCRONA HORIZONTAL HTD 22-8M-20-5F ACERO	4 un	5251	21005	JUAN CARLOS MILESI
CASQUILLO CONICO 1008-12	4 un			
ALTERNATIVA: CASQUILLO CONICO 1008-09	4 un	4940	19760	JUAN CARLOS MILESI
RODAMIENTO POLEA VERTICAL SKF 628/8-2Z	2 un	6595	13190	LA ESPERANZA RODAMIENTOS
RODAMIENTO POLEA HORIZONTAL SKF 609-2Z	8 un	4133	33064	LA ESPERANZA RODAMIENTOS
OTROS	1 un	33000	33000	

ELECTRÓNICA Y CONTROL

DRIVER MOTOR VERTICAL LEADSHINE DM542E	1 un	16736	16736	ELINDAR SRL
DRIVER MOTOR HORIZONTAL LEADSHINE DM556E	2 un	20722	41444	ELINDAR SRL



FUENTE CONMUTADA DRIVER MOTOR VERTICAL 24VDC 10A	1 un	9000	9000	LIBERCAM
FUENTE CONMUTADA DRIVER MOTOR HORIZONTAL 24VDC 15A	2 un	12000	24000	LIBERCAM
FUENTE CONMUTADA CIRCUITO DE COMANDO 24VDC 3A	1 un	6000	6000	LIBERCAM
CONVERSOR DC/DC 24/12V 5A	1 un	6210	6210	ENERKIT
ARDUINO MEGA 2560 R3	1 un	23710	23710	TECNOLIVEUSA
MODULO BLUETOOTH HC-05 Ó HC-06	1 un	1602	1602	TECNOLIVEUSA
TABLERO METALICO ESTANCO 300x450x225 mm (GENROD 099203)	1 un	18719	18719	LIESA SA
CONTACTOR 25A AC1 BOB. 24VDC AUX. 1NC (SCHNEIDER LC1D09BD)	3 un	15120	45360	LIESA SA
RELE DE ESTADO SOLIDO DE POTENCIA 3A Un:24VDC (PHOENIX CONTACT OPT-24DC/24DC/2)	3 un	7328	21983	LIESA SA
FINES DE CARRERA ACTUADOR TIPO ROLDANA DE EMBOLO ABB LS21 M12D11-P01	2 un			
ALTERNATIVA: SCHNEIDER XCKP2106G11	2 un	9853	19706	LIESA SA
FINES DE CARRERA ACTUADOR TIPO ROLDANA DE EMBOLO ABB LS21 M14D11-P01	1 un			
ALTERNATIVA: SCHNEIDER XCKP2106G11	1 un	9853	9853	LIESA SA
SENSOR DE POSICION BARRERA FOTOELECTRICA IFM OPU200	2 un	49357	98714	IFM ELECTRONIC SRL
CABLE DE CONEXIÓN DE SENSOR CONECTOR M8 X 5M IFM EVC142	2 un	5342	10684	IFM ELECTRONIC SRL
CABLE DE POTENCIA BLINDADO MARLEW XF 0415 WNe/NBR1	16 m			
ALTERNATIVA: CABLE DE AUTOMATIZACION INDUSTRIAL BLINDADO MARLEW EC 0415	16 m	1439	23024	LIESA SA
CABLE DE SEÑAL BLINDADO MARLEW AR 5200	12 m	563	6756	LIESA SA
CABLE UNIPOLAR IMSA 2,5 mm	20 m	194	3880	LIESA SA
OTROS	1 un	30000	30000	
TOTAL DE MATERIALES		\$ 1.264.533		

Cotización en pesos teniendo el dólar un valor de \$ 160,00 según precio de venta del Banco de la Nación Argentina para billetes.

En cuanto a la mano de obra se considera que el posicionador podría construirse en el taller de la Facultad de Ingeniería de la UNNE, por lo que no tendría un costo directo.



25.2 PROVEEDORES

A continuación se indican los datos de contacto de los proveedores de los principales componentes:

- HIGINIO BERTONCINI Y CIA S.A. Tel: 0362 – 4463 333
Dir: Santiago del Estero 2415 – Resistencia. Chaco.
Web: bertoncinionline.com.ar

- JUAN CARLOS MILESI Tel: 011 – 4605 4606
Dir: Cosquin 5555 – CABA.
Web: -

- ALUMAR S.R.L. Tel: 0362 – 4435 400
Dir: RN 16 Km 13,5 – Resistencia. Chaco.
Web: alumarsrl.com.ar

- ALUMINA ARGENTINA S.R.L. Tel: 011 – 4750 3680
Dir: Av Bartomomé Mitre 3380 – Caseros. Bs As.
Web: alumina-argentina.com.ar

- LINEARTEC Tel: 011 – 4469 4800
Dir: Ombú 3737 – Ciudadela. Bs. As.
Web: lineartec.com.ar

- CNC INSUMOS S.R.L. Tel: 011 – 4629 7689
Dir: J. M. de Rosas 145 – Castelar. Bs. As.
Web: cncinsumos.com

- ELINDAR S.R.L. Tel: 011 – 4262 1613
Dir: Av. José de San Martín 2407 – Lanús Oeste. Bs. As.
Web: elindar.com.ar

- INSUMOS CAC Tel: 011 – 7544 0105
Dir: Av. J. M. de Rosas 793 – Lomas del Mirador. Bs. As.
Web: insumoscac.mitiendanube.com

- LA INDUSTRIAL ELÉCTRICA S.A. Tel: 011 – 4686 5777
Dir: Montiel 3060 – CABA.
Web: liesa.com.ar



- IFM ELECTRONIC S.R.L.

Tel: 011 – 5353 3436

Dir: Lola Mora 421, Piso 10, Of 3 – Puerto Madero Bs. As.

Web: ifm.com

26. POSIBLES MEJORAS

- Actualmente cuando se desee ir a una coordenada determinada el posicionador avanzará o retrocederá la cantidad de pasos necesarios para alcanzar las coordenadas de destino. En este caso la posición final real variará ligeramente dependiendo de si el posicionador debiera avanzar o retroceder para llegar hasta dicho punto. Esta ligera variación será producto de la deformación propia de la correa y del juego que existe entre los engranajes de los reductores de velocidad, ya que estos absorberán parte del desplazamiento cuando se produzca un cambio en el sentido de giro del motor.

Para asegurar entonces que al ir hasta una coordenada en particular, el desplazamiento finalice siempre en el mismo lugar, el movimiento para llegar hasta ese punto debería realizarse siempre en un mismo sentido. Por ello si por ejemplo se quisiera avanzar desde una coordenada menor a una mayor, podría realizarse el avance directamente desde una coordenada a la otra en el sentido positivo. En cambio, si se deseara ir de una coordenada mayor a una menor, debería primeramente retrocederse en exceso, para luego avanzar en sentido positivo hasta la coordenada deseada. Así, en ambos casos se llegaría finalmente a la coordenada deseada mediante un desplazamiento positivo y por ello el error producido por la deformación de la correa y el juego entre los engranajes del reductor sería siempre el mismo y en el mismo sentido, por lo que la posición final sería siempre la misma.

Esto podría hacerse manualmente por el operador teniendo el cuidado de llevar al posicionador a la coordenada deseada avanzando siempre en una misma dirección, o bien podría corregirse de forma automática mediante la programación, haciendo por ejemplo, que el posicionador retroceda determinada cantidad de pasos en exceso para luego avanzar nuevamente esa cantidad de pasos.

- Al introducir objetos en el túnel la relación de bloqueo de estos, definido como la relación entre su superficie expuesta (superficie proyectada sobre un plano transversal al túnel) y la superficie transversal del túnel, no debe ser superior al 5 u 8 %. En este caso la superficie total del túnel es de 43200 cm^2 , mientras que la superficie expuesta total del posicionador sería de 6412 cm^2 , por lo que la relación de bloqueo del posicionador sería del 14,8 %, valor fuera del rango aceptable. Por su parte, si se considera solamente la parte central del posicionador, sin considerar la superficie expuesta de los ejes horizontales superior e inferior, la superficie del túnel se reduciría a 38304 cm^2 , mientras que la superficie expuesta del posicionador se reduciría a 1516 cm^2 , por lo que la relación de bloqueo del posicionador sería en este caso de 3,96 %. Las acciones a tomar respecto a esto deberán analizarse una vez construido el posicionador

En primer lugar se deberá evaluar la necesidad o no de colocar las chapas de protección horizontal superior e inferior para proteger del viento a las correas y la cadena portacables ya que, al encontrarse estos muy próximos al techo y piso del túnel respectivamente, la baja velocidad del viento por efecto de la capa límite en estos sectores puede hacer innecesaria su implementación. De esta forma se reduciría considerablemente la superficie expuesta y con ello la relación de bloqueo.

En caso de ser necesario montar las chapas de protección horizontales, se podría analizar la implementación de perfiles aerodinámicos en la parte posterior del posicionador, a fin de que el aire fluya a través de este de forma suave reduciendo así la turbulencia producida por el mismo.

Puede también analizarse la posibilidad de alejar el posicionador de manera que la deformación del flujo producida por este no genere influencia aguas arriba sobre modelo. En este caso se deberá prolongar el soporte del portainstrumento.

Podría también evaluarse utilizar como elementos de transmisión tornillos de transmisión o tornillos de bola recirculaste, ya que por la rigidez de estos no sería necesario protegerlos del viento.

- Debido a la gran variabilidad de precios de los distintos componentes, al momento de construir el posicionador podría realizarse una nueva comparación de costos entre los distintos tipos de transmisión (tornillos de transmisión, tornillos de bolas recirculantes y correas síncronas), sus distintos



accesorios y tamaño de motores requeridos, realizando luego el rediseño estructural correspondiente si fuese necesario.

- Podría modificarse el mecanismo de sujeción rígida entre el eje horizontal superior y los rieles, de manera que permita avances axiales continuos en lugar de avances discretos. Para ello podría utilizarse un sistema de sujeción a presión por el riel en lugar de la sujeción actual mediante bulones.
- Si un driver se desenergizara, mediante los relés de estado sólido se producirá la apertura de los tres contactores con lo cual se desenergizarían todos los drivers. En el funcionamiento actual esto detendría a los motores pero la programación continuaría ejecutándose normalmente, ya que una luz piloto en el tablero de drivers indicará que los contactores se encuentran abiertos pero no habrá una señal que indique en la pantalla de la aplicación de LabVIEW que se producido una falla.
Esto podría solucionarse utilizando la misma señal del contacto NC de un contactor para alimentar (mediante un divisor de tensión) una entrada digital tipo interrupción del Arduino, la cual al activarse interrumpiría la acción realizada en ese momento y enviaría una señal a la aplicación de LabVIEW para que esta indique al operador que se han detenido los motores.
- El ingreso de las variables N°MaxPH y N°MaxPV por parte del usuario podría eliminarse si el rango de funcionamiento del posicionador fuera determinado automáticamente mediante un par extra de sensores ubicados en el extremo superior de cada desplazamiento. De esta manera, el posicionador luego de determinar el CG podría avanzar continuamente hasta ser detectado por estos sensores, luego las variables N°MaxPH y N°MaxPV tomarían los valores de los contadores nhg y nvh respectivamente.
El problema en este caso se presentaría si existiera alguna pieza o elemento fijo (como por ej. parte del objeto a ensayar o su soporte) que obstaculice la zona de trabajo determinada por estos sensores. En estos casos el posicionador colisionaría contra estos elementos antes de ser detectado por estos sensores extras. En cambio, para el caso actual en que el número máximo de pasos es ingresado por el operario, este podrá darle los valores adecuados para evitar colisionar con estos obstáculos.
En la realidad, el que convenga utilizar una forma u otra de establecer los límites dependerá en todo caso del ensayo a realizar y las distintas características del mismo, por lo que podría modificarse el sistema de manera que el operario decida si se van a ingresar estos límites manualmente o de forma automática.
En el presente trabajo se decidió ingresar los valores solo de forma manual ya que su procedimiento no es laborioso, más aun teniendo en cuenta que las demás variables de ajuste se obtienen de forma similar y por lo tanto no implicaría un mayor esfuerzo realizarlo de este modo. Por otra parte se estaría ahorrando también el costo de los sensores extra.
- La interfaz APP para celular podría mejorarse incluyendo un indicador de coordenadas y botones para poder ejecutar las funciones IR A MEMORIA. Esto no se realizó en este momento para no sobrecargar la pantalla de esta con botones e indicadores que no se consideran necesarios según la finalidad de esta interfaz.
Para agregar los botones de las funciones IR A MEMORIA simplemente habría que repetir lo realizado con los botones de inicio de las demás funciones. En cuanto a la visualización de las coordenadas, desde Arduino debería enviarse una cadena de caracteres con las mismas, similar a la enviada a la Aplicación de LabVIEW, solo que en este caso cada coordenada en lugar de ir encerrada entre caracteres específicos deberán estar separados mediante un mismo carácter denominado separador de campos. Luego mediante el bloque "recorta.texto" se dividirá el String obteniendo las coordenadas por separado para asignarlas a su indicador correspondiente.



- Ya que el sistema de control de los motores paso a paso se realizará mediante lazo abierto, resultaría conveniente montar un sistema para verificar que los motores horizontales avancen de forma pareja y dar aviso en caso de que esto no ocurriera. De esta manera podría detectarse si alguno de estos motores perdiera pasos produciendo inclinación en el eje vertical. Esta pérdida de pasos podría ocurrir por ej. en caso de no distribuir el peso del eje vertical proporcionalmente sobre los dos ejes horizontales durante el montaje del posicionador, lo cual produciría mayor torque resistente sobre uno de los motores y este podría no avanzar correctamente.
- Sería conveniente realizar una simulación de flujo para determinar el efecto del viento sobre las chapas de protección, obteniendo así valores más precisos de las cargas producidas por el viento sobre las mismas, para luego compararlos con los utilizados en este proyecto, los cuales fueron obtenidos mediante diferentes consideraciones y simplificaciones.
Estas simulaciones se intentaron realizar sin éxito mediante el software Ansys Fluent. Para ello en primer lugar se realizó la simulación sobre un perfil alar cuyos coeficientes de arrastre y sustentación son conocidos, a modo de validar el procedimiento utilizado. En las numerosas simulaciones realizadas los coeficientes obtenidos fueron levemente inferiores a la mitad de los valores de los coeficientes reales, por lo que el procedimiento de simulación no pudo considerarse confiable y se decidió continuar el dimensionamiento obteniendo el valor de las cargas de forma analítica mediante tablas.
- Podría resultar de utilidad incorporar una nueva función que permita realizar mediciones automáticas con coordenadas circulares, indicando para ellos el número y la distancia entre mediciones en una misma dirección y el ángulo que abarcará la zona de muestreo.



27. FUENTES CONSULTADAS

- Stepping motors and their microprocessor controls, Takashi Kenjo, 1984.
- Manual “Sistema de motores paso a paso Sure Step”, Automation Direct, 2007.
- Resistencia de materiales, P. A. Stiopin
- Mecánica de los Fluidos, M. Potter y D. Wiggert, 3ra Ed.
- Diseño en ingeniería mecánica de Shigley, Richard G. Budynas, 8va Ed.
- Tecnología Mecánica Tomo I, Pascual Pezzano, 11va Ed, 1992
- Catálogo de correas síncronas Gates
- Catálogo de reductores planetarios Micron modelo TRUE
- Catalogo Rodamientos SKF – Enero 2019
- Información y datos proporcionados por el personal del Laboratorio de Aerodinámica de la UNNE.



ANEXO A

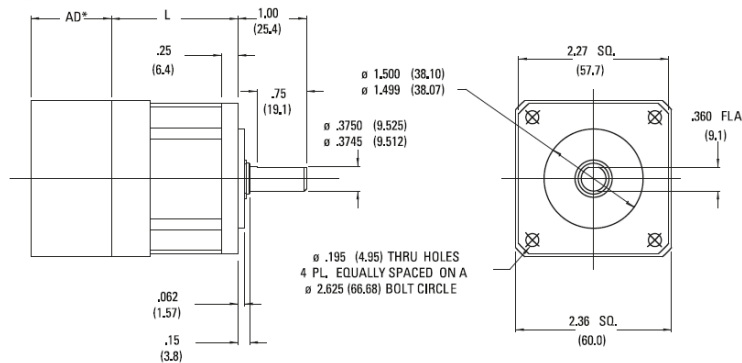
CATÁLOGOS DE COMPONENTES UTILIZADOS

A.1 REDUCTOR PLANETARIO - MICRON

NemaTRUE™ Size 23/60

True Planetary Gearheads

English - Dimensions are in. (mm)



Ratio¹	NT23 Dimension 'L' in. [mm]	NT60 Dimension 'L' in. [mm]	Backlash [arc-min]		Weight lb. [kg]	Efficiency
			Precision	High Precision		
3:1 to 10:1	1.92 [48.8]	2.07 [52.6]	13 max	8 max	1.5 [0.7]	93%
15:1 to 100:1	2.61 [66.3]	2.76 [70.1]	15 max	9 max	1.9 [0.9]	88%

AD** = Adapter length
Adapter length will vary depending
on motor.
Efficiency is calculated at 100% of the
rated torque.

Performance Specifications

Part Number		Ratio ¹	2000 Hour Life				Tpeak in.-lb. [Nm]	J in.-lb.-sec²x10 ⁻⁴ [kg-cm²]	Torsional Stiffness		
English	Metric		T _r (1000 rpm) in.-lb. [Nm]	T _r (2000 rpm) in.-lb. [Nm]	T _r (3000 rpm) in.-lb. [Nm]	T _r (4000 rpm) in.-lb. [Nm]			in.-lb./arc-min [Nm/arc-min]	NT	NTP
NT23-003	NT60-003	3:1	98 [11,1]	80 [9,0]	70 [7,9]	65 [7,3]	150 [16,9]	0.67 [0,076]	6.3 [0,71]	15.8 [1,8]	
NT23-004	NT60-004	4:1	107 [12,1]	90 [10,2]	80 [9,0]	73 [8,2]	250 [28,3]	0.22 [0,024]	6.3 [0,71]	15.8 [1,8]	
NT23-005	NT60-005	5:1	105 [11,9]	88 [9,9]	78 [8,8]	71 [8,0]	250 [28,2]	0.22 [0,025]	6.3 [0,71]	15.8 [1,8]	
NT23-007	NT60-007	7:1	100 [11,3]	83 [9,4]	74 [8,4]	66 [7,5]	250 [28,3]	0.22 [0,024]	6.3 [0,71]	15.8 [1,8]	
NT23-010	NT60-010	10:1	163 [7,1]	58 [6,6]	54 [6,1]	52 [5,9]	250 [28,2]	0.14 [0,016]	6.3 [0,71]	15.8 [1,8]	
NT23-015	NT60-015	15:1	124 [14,0]	110 [12,4]	105 [11,9]	100 [11,3]	250 [28,3]	0.21 [0,024]	6.3 [0,71]	15.8 [1,8]	
NT23-020	NT60-020	20:1	180 [20,3]	147 [16,6]	130 [14,7]	120 [13,6]	275 [31,1]	0.21 [0,024]	6.3 [0,71]	15.8 [1,8]	
NT23-025	NT60-025	25:1	175 [9,8]	142 [16,0]	125 [14,1]	115 [13,0]	250 [28,3]	0.21 [0,024]	6.3 [0,71]	15.8 [1,8]	
NT23-030	NT60-030	30:1	207 [23,4]	182 [20,6]	157 [17,7]	147 [16,6]	275 [31,1]	0.13 [0,015]	6.3 [0,71]	15.8 [1,8]	
NT23-040	NT60-040	40:1	207 [23,4]	182 [20,6]	157 [17,7]	147 [16,6]	275 [31,1]	0.13 [0,015]	6.3 [0,71]	15.8 [1,8]	
NT23-050	NT60-050	50:1	202 [22,8]	175 [19,8]	152 [17,2]	142 [16,0]	275 [31,1]	0.13 [0,015]	6.3 [0,71]	15.8 [1,8]	
NT23-070	NT60-070	70:1	197 [22,3]	172 [19,4]	147 [16,6]	137 [15,5]	275 [31,1]	0.13 [0,015]	6.3 [0,71]	15.8 [1,8]	
NT23-100	NT60-100	100:1	85 [9,6]	79 [8,9]	74 [8,9]	73 [8,2]	275 [31,1]	0.13 [0,015]	6.3 [0,71]	15.8 [1,8]	

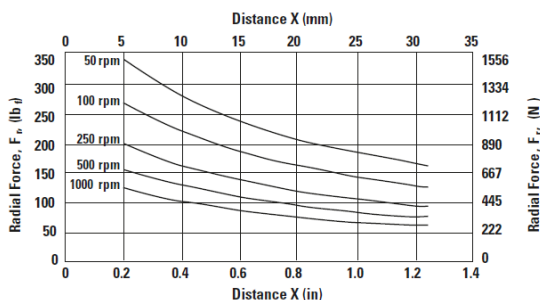
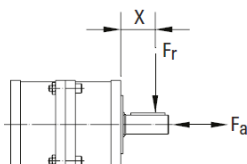
All ratios are available to ship in 24 hours through the Gearhead Express Program.

¹ Ratios are exact, higher ratios and other custom options are also available. consult factory.

T_{peak} = Allowable momentary peak torque for emergency stop or heavy shock loading.
 J = Mass moment of inertia reflected to the input shaft (including pinion assembly).

T_r = Rated output torque at rated speed for specific hours of life.

NT23, NTP23, NT60



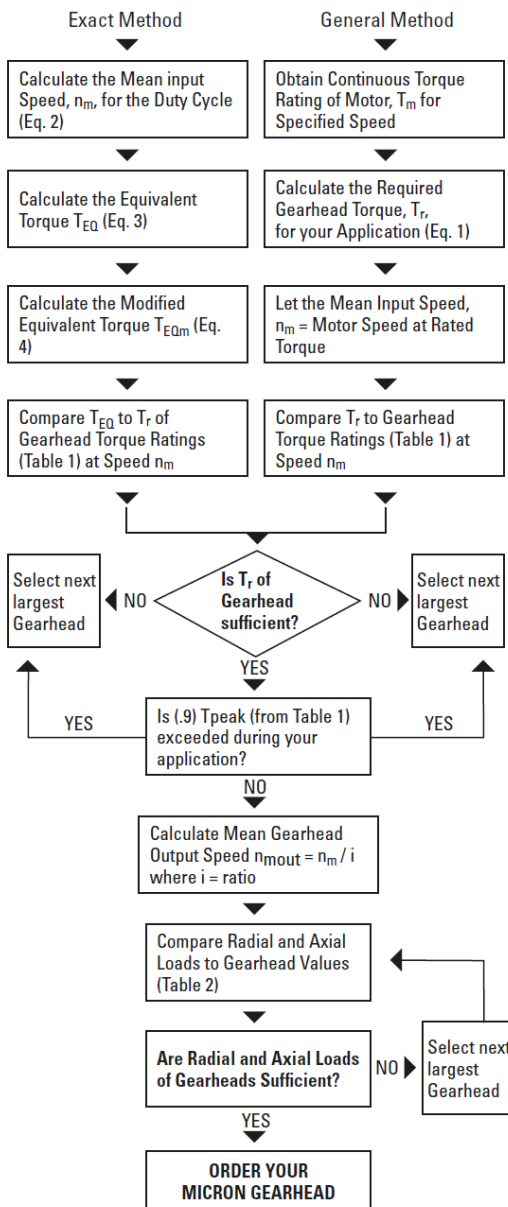
Speed rpm	Axial Load, F_a lb _f [N]
250	310 [1379]

Sizing and Selection

Step 1: Select the required precision class and gearhead configuration (in-line or right angle).

Step 2: Select the proper gearhead using exact or general method.

For continuous duty applications, please contact Applications Engineering.



General Method:

Required Gearhead Torque(T_r)

$$(1) T_r = T_M^* \times i \times e$$

where:

T_M^* = continuous torque of motor

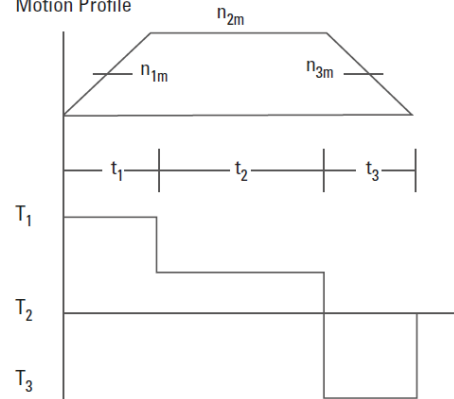
i = Gearhead ratio

e = efficiency of Gearhead

* Since many motors are capable of exceeding their continuous torque rating for extended lengths of time, the value for T_M will only provide a starting point for Gearhead selection. Only use the general method if the continuous motor rating is not exceeded in the application.

Exact Method

Motion Profile



t_n = time period n

n_{nm} = mean speed during time period t_n

T_n = torque during time period t_n

Mean input speed (n_m)

$$(2) n_m = \frac{n_{1m}t_1 + n_{2m}t_2 + n_{3m}t_3 + \dots + n_{nm}t_n}{t_t}$$

where $t_t = t_1 + t_2 + t_3 + \dots + t_n$

Equivalent torque (T_{EQ})

$$(3) T_{EQ} = 8.7 \sqrt{\frac{T_1^{8.7} n_{1m} t_1 + T_2^{8.7} n_{2m} t_2 + T_3^{8.7} n_{3m} t_3 + \dots + T_n^{8.7} n_{nm} t_n}{n_m t_t}}$$

Modified equivalent torque (T_{EQm})

$$(4) T_{EQm} = T_{EQ} / Q$$

where Q is:

Q	# of cycles/hr
1,0	>0
0,9	>1000
0,7	>2500
0,5	>5000

For applications > 10,000 cycles/hour or for continuous duty operation, please contact application engineering.

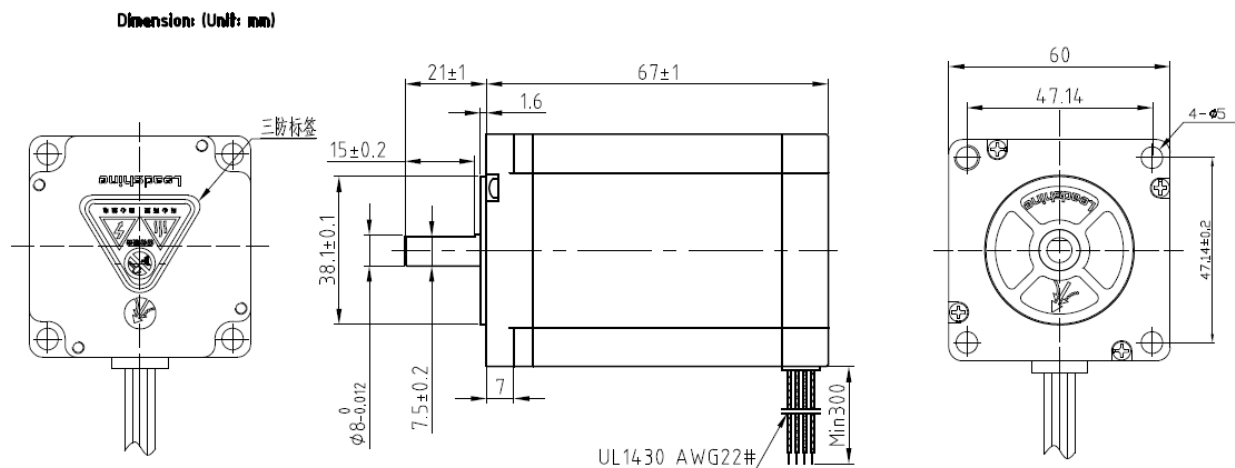


A.2 MOTOR EJE VERTICAL - LEADSHINE



Technical Data:-

Model:	D57CM21
Frame Size:	NEMA 23
Phase:	2
Step Angle:	1.8°
Accuracy:	±0.9°
Length:	67 mm (2.64 inch)
Number of Wires:	4
Holding Torque:	2.1N.m (298 oz-in)
Phase Current:	5 A/Phase
Voltage:	1.26 V
Resistance:	0.21 Ω
Inductance:	0.75 mH
Rotor Inertia:	0.57 kg-cm ² (0.00807 oz-in-sec ²)
Shaft Size:	8 mm (0.315 inch)
Shaft Type:	Single
Weight:	1.2 kg (2.64 lb)
Insulation Class:	B
Insulation resistance:	Min 100 M Ω , 500 VDC)
Operating Temperature:	-10 - 50°C (14 - 122°F)
Temperature Rise (Max):	80K
Radial Play:	Max 0.02 mm with 450g load
Axial Play:	Max 0.08 mm with 900g load





A.3 TRANSMISIÓN EJE VERTICAL - GATES



Correas Industriales - Informe del calculo

Diseño Flex

Calculado para:		Realizado por: Marcelo Joaquin Obregon - juaco0@hotmail.com - Teléfono	
Aplicación: Cálculo #1			
DATOS			
Info. transmisión		Motriz	Conducida
Ratio velocidad:	1,00 Estructura del: 187,5 Reductor: 5,00 RPM:	37,50	37,50 +/-4%
Pot. del motor:	42 W, Eficiencia: 100,00 % Velocidad periférica máxima:	33 m / s	33 m / s
Reductor:	39,06 W, Eficiencia: 93,00 %		
Factor servicio:	1,3		
Potencia cálculo:	51 W		
Distancia entre ejes:	1500 mm +/-25%	Casquillos evaluados:	Taper L., Taladro único mínimo
Estándar motor:	Motor eléctrico	Correas evaluadas:	PowerGrip GT4, PowerGrip GT3, PowerGrip GTX, PowerGrip HTD
TRANSMISIÓN ELEGIDA			
Tipo de correa: PowerGrip GT3 - 5MGT		Correa	Motriz
Ref. Pieza:		2440-5MGT3-15	24-5MR-15
Ref. Producto:		9400-55292	
Ratio velocidad:	1.0	Diámetro primit:	38,20 mm
RPM conducida:	37,50	RPM:	37,50
Pot. nominal:	84 W	Velocidad de correa:	0,1 m / s
Fuerza tracción:	670 N	Ancho superior:	30,00 mm
Distancia entre ejes:	1160,0 mm	Ref. casquillo:	Agujero mínimo
Rango instal. / retens.	1137,6 mm a 1161,0 mm	Agujero:	Agujero mínimo
		Peso:	150 g
TENSIÓN			
Correa nueva		Correa usada	Cuando vaya a reinstalar correas usadas, mida y anote la tensión antes de desmontar y vuelva a instalar con la misma tensión.
Tensión estática (por canal/ramal):	316 a 348 N	221 a 253 N	
Tensión estática (tracción total de):	632 a 695 N	442 a 506 N	
Deflexión por canal/ramal:	23,00 mm	23,00 mm	
Fuerza por canal/ramal:	3,3 a 3,5 kgf	2,5 a 2,8 kgf	
Tensímetro sónico:	316 a 348 N	221 a 253 N	
Frecuencia de la correa:	31 a 32 Hz	26 a 28 Hz	
Datos Sonic 507C/508C: Masa 4,1g / m, Ancho: 15 mm/#R, Long. ramal: 1160			
NOTAS			
- Puede ser necesario poner flancos en ambas poleas para guiar la correa adecuadamente. - La alineación es crítica para larga distancias centro. - Verifique la capacidad de la Carga OverHung (OHL) de la caja de engranajes. - La polea motriz debe mecanizarse de acuerdo con el diámetro solicitado. Prevea un plazo de entrega adecuado. - El taladro de la polea motriz debe cumplir con el ajuste H8. El eje debe introducirse al 100%. - Se debe mecanizar el agujero de la polea conducida al tamaño solicitado. Prevea un plazo de entrega adecuado. - El taladro de la polea conducida debe cumplir con un ajuste H8. El eje debe introducirse al 100%.			
Este informe: (1) sólo se aplica a los productos de Gates; (2) contiene información confidencial; (3) solamente puede ser revelada a apoyar la venta o el mantenimiento de nuestros productos; y (4) no es una garantía de rendimiento. Productos Gates no están diseñados, fabricados o probados para su uso en aplicaciones aeronáuticas, incluidas las hélices de aviones o sistemas de accionamiento de rotor, y todas las aplicaciones en el aire con o sin tripulación de cualquier tipo. Sistemas de elevación y el frenado tienen consideraciones especiales. El comprador es el único responsable de la selección y prueba de productos para cualquier uso previsto. El presente informe y ningún producto mencionado en este informe están sujetos a Condiciones puertas estándar y condiciones de venta, incluyendo todas las renunciaciones, exclusiones y limitaciones de las garantías, expresas o implícitas. Estos términos pueden ser encontrados en ww2.gates.com/termsofsale.			

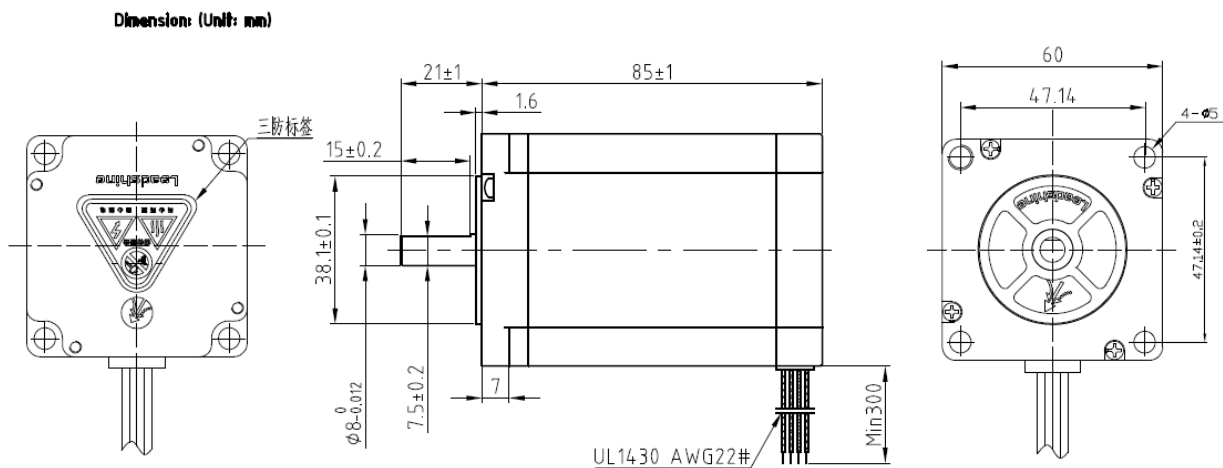


A.4 MOTOR EJE HORIZONTAL - LEADSHINE



Technical Data:-

Model:	D57CM31
Frame Size:	NEMA 23
Phase:	2
Step Angle:	1.8°
Accuracy:	±0.9°
Length:	88 mm (3.46 inch)
Number of Wires:	4
Holding Torque:	3.1N.m (439 oz-in)
Phase Current:	6 A/Phase
Voltage:	1.56 V
Resistance:	0.26 Ω
Inductance:	1.18 mH
Rotor Inertia:	0.84 kg-cm ² (0.01189 oz-in-sec ²)
Shaft Size:	8 mm (0.315 inch)
Shaft Type:	Single
Weight:	1.4 kg (3.08 lb)
Insulation Class:	B
Insulation resistance:	Min 100 MΩ, 500 VDC)
Operating Temperature:	-10 - 50°C (14 - 122°F)
Temperature Rise (Max):	80K
Radial Play:	Max 0.02 mm with 450g load
Axial Play:	Max 0.08 mm with 900g load





A.5 TRANSMISIÓN EJE HORIZONTAL - GATES



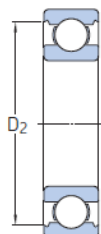
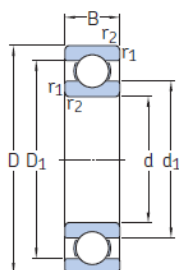
Correas Industriales - Informe del calculo

Diseño Flex

Calculado para:	Realizado por:	Marcelo Joaquin Obregon - juaco0@hotmail.com - Teléfono		
Aplicación: Cálculo #1				
DATOS				
Info. transmisión		Motriz	Conducida	
Ratio velocidad:	1,00 Estructura del: 187,5 Reductor: 10,00 RPM:	18,75	18,75 +/-4%	
Pot. del motor:	61 W, Eficiencia: 100,00 % Velocidad periférica máxima:	33 m / s	33 m / s	
Reductor:	56,73 W, Eficiencia: 93,00 %			
Factor servicio:	1,3			
Potencia cálculo:	74 W			
Distancia entre ejes:	2250 mm +/-25%	Casquillos evaluados:	Taper L., Taladro único mínimo	
Estándar motor:	Motor eléctrico	Correas evaluadas:	PowerGrip GT4, PowerGrip GT3, PowerGrip GTX, PowerGrip HTD	
TRANSMISIÓN ELEGIDA				
Tipo de correa: PowerGrip GT4 - 8MGT		Correa	Motriz	Conducida
	Ref. Pieza:	4400-8MGT4-20	22-8M-20	22-8M-20
Ratio velocidad:	1,0	Ref. Producto:	9579-50392	
RPM conducida:	18,75	Diámetro primit:	--	56,02 mm
Pot. nominal:	90 W	RPM:	0,7499	18,75
Fuerza tracción:	1326 N	Velocidad de correa:	0,1 m / s	0,1 m / s
Distancia entre ejes:	2112,0 mm	Ancho superior:	--	40,00 mm
Rango instal. / retens.	2073,9 mm a 2113,3 mm	Ref. casquillo:	--	1008
		Agujero:	--	9 mm - 25,4 mm
		Perno de par	--	6,2 Nm
		Peso:	427 g	--
TENSIÓN				
Tensión estática (por canal/ramal):	Correa nueva	Correa usada	Cuando vaya a reinstalar correas usadas, mida y anote la tensión antes de desmontar y vuelva a instalar con la misma tensión.	
Tensión estática (tracción total de):	626 a 688 N	438 a 501 N		
Deflexión por canal/ramal:	1252 a 1377 N	876 a 1001 N		
Fuerza por canal/ramal:	42,00 mm	42,00 mm		
Tensímetro sónico:	6,3 a 6,8 kgf	4,8 a 5,3 kgf		
Frecuencia de la correa:	626 a 688 N	438 a 501 N		
	19 a 20 Hz	16 a 17 Hz		
NOTAS				
- Puede ser necesario poner flancos en ambas poleas para guiar la correa adecuadamente.				
- La alineación es crítica para larga distancias centro.				
- Verifique la capacidad de la Carga OverHung (OHL) de la caja de engranajes.				
Este informe: (1) sólo se aplica a los productos de Gates; (2) contiene información confidencial; (3) solamente puede ser revelada a apoyar la venta o el mantenimiento de nuestros productos; y (4) no es una garantía de rendimiento.				
Productos Gates no están diseñados, fabricados o probados para su uso en aplicaciones aeronáuticas, incluidas las hélices de aviones o sistemas de accionamiento de rotor, y todas las aplicaciones en el aire con o sin tripulación de cualquier tipo. Sistemas de elevación y el frenado tienen consideraciones especiales. El comprador es el único responsable de la selección y prueba de productos para cualquier uso previsto.				
El presente informe y ningún producto mencionado en este informe están sujetos a Condiciones puertas estándar y condiciones de venta, incluyendo todas las renunciaciones, exclusiones y limitaciones de las garantías, expresas o implícitas. Estos términos pueden ser encontrados en ww2.gates.com/termsofsale .				

A.6 RODAMIENTOS PARA POLEAS HORIZONTALES - SKF

1.1 Rodamientos rígidos de una hilera de bolas d 9 – 10 mm



2Z



2RSL



2RS1

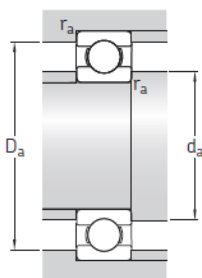


2RS1



2RSH

Dimensiones principales			Capacidad de carga básica		Carga límite de fatiga	Velocidades nominales		Masa	Designaciones	
d	D	B	dinámica C	estática C ₀	P _u	Velocidad de referencia	Velocidad límite ¹⁾		Rodamiento abierto o tapado en ambos lados	tapado en un lado ¹⁾
mm			kN		kN	r. p. m.		kg	–	
9 cont.	24	7	3,9	1,66	0,071	–	19 000	0,015	▶ 609-2RSH	609-RSH
	24	7	3,9	1,66	0,071	70 000	34 000	0,014	▶ 609-2RSL	609-RSL
	24	7	3,9	1,66	0,071	70 000	34 000	0,015	▶ 609-2Z	609-Z

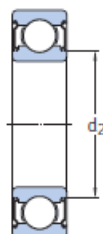
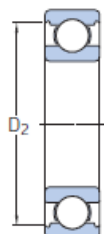
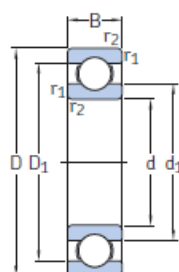


Dimensiones						Dimensiones de resaltes y radios de acuerdo				Factores de cálculo	
d	d ₁	d ₂	D ₁	D ₂	r _{1,2} min.	d _a mín.	d _a máx.	D _a máx.	r _a máx.	k _r	f ₀
mm						mm				–	
9 cont.	–	12,8	–	21,2	0,3	11	12,5	22	0,3	0,025	13
	–	12,8	–	21,2	0,3	11	12,5	22	0,3	0,025	13
	14,4	–	–	21,2	0,3	11	14,3	22	0,3	0,025	13



A.7 RODAMIENTOS PARA POLEA DE RETORNO VERTICAL - SKF

1.1 Rodamientos rígidos de una hilera de bolas d 7 – 9 mm



2Z



2Z



2RSL



2RZ



2RS1

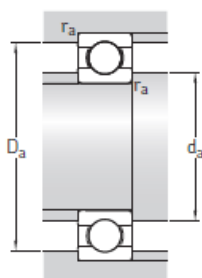


2RS1



2RSH

Dimensiones principales			Capacidad de carga básica		Carga límite de fatiga	Velocidades nominales		Masa	Designaciones	
d	D	B	dinámica C	estática C ₀	P _u	Velocidad de referencia	Velocidad límite ¹⁾		Rodamiento abierto o tapado en ambos lados	tapado en un lado ¹⁾
mm			kN		kN	r. p. m.		kg	–	
8	16	4	0,819	0,3	0,012	90 000	56 000	0,003	618/8	–
	16	5	1,33	0,57	0,024	–	26 000	0,0036	628/8-2RS1	–
	16	5	1,33	0,57	0,024	90 000	45 000	0,0036	628/8-2Z	–



Dimensiones						Dimensiones de resaltes y radios de acuerdo				Factores de cálculo	
d	d ₁	d ₂	D ₁	D ₂	r _{1,2} min.	d _a min.	d _a máx.	D _a máx.	r _a máx.	k _r	f ₀
mm						mm				–	
8	10,5	–	13,5	–	0,2	9,4	–	14,6	0,2	0,015	7,5
	10,1	–	–	14,2	0,2	9,4	9,4	14,6	0,2	0,015	11
	10,1	–	–	14,2	0,2	9,4	10	14,6	0,2	0,015	11

A.8 CADENAS PORTA CABLES - FESMA



Série 18 - Altura Interna 18mm

Própria para proteção de cabos com até 15mm de diâmetro, as esteiras da série 18 são pequenas, porém muito fortes e robustas.



- Modelo FB com abertura destacável, tipo dobradiça



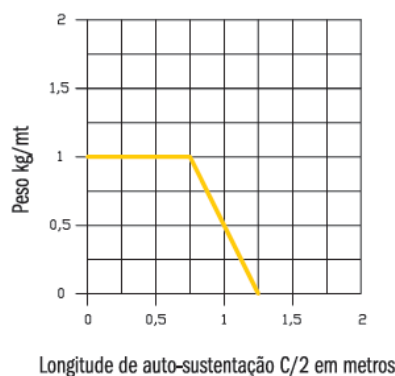
- 4 opções de curvatura



- 4 diferentes opções de largura



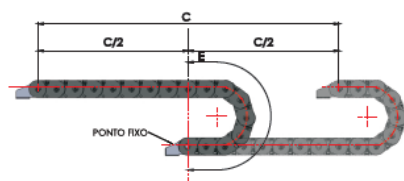
- Disponível separador vertical
para os modelos FB



Características:	FA/FB.18.15	FA/FB.18.18	FA/FB.18.26	FA/FB.18.37
Elos por metro	32	32	32	32
Temperatura de trabalho	-30° / + 120 C°	-30° / + 120 C°	-30° / + 120 C°	-30° / + 120 C°
Material	Nylon	Nylon	Nylon	Nylon
Peso por metro	256 gr.	288 gr.	320 gr.	384 gr.
Cor padrão	Preto	Preto	Preto	Preto

Calculando o comprimento

Todas nossas esteiras são fornecidas por metro. Para calcular o comprimento, você deverá primeiramente saber qual é o curso total de deslocamento (C) do equipamento, ou seja, a distância percorrida de um extremo ao outro. Será determinante para calcular o comprimento da esteira o posicionamento do ponto fixo, que pode ser:



1 - Pelo Centro do Curso: Quando o ponto fixo for instalado no centro do curso (forma mais econômica), teremos o comprimento total da esteira somando $C/2 + E$.



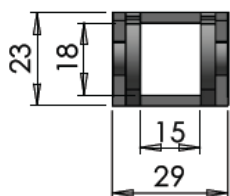
Série 18 - Esteira Pequena e Robusta



R	28	38	75	100
H	79	99	173	223
D	58	66	118	151
E	152	183	300	378

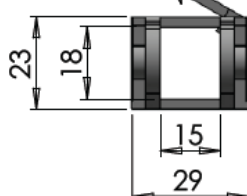
Para especificar o comprimento total da esteira, consulte a página 02

FA

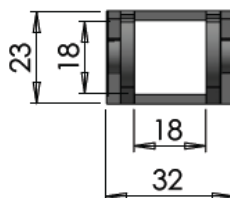


Raio (R)	Código
28	FA.18.15.28
38	FA.18.15.38
75	FA.18.15.75
100	FA.18.15.100

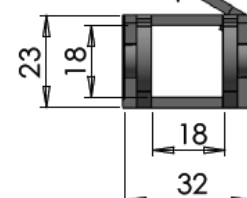
FB



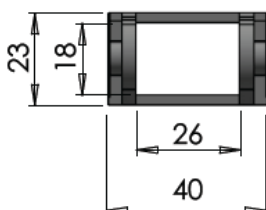
Raio (R)	Código
28	FB.18.15.28
38	FB.18.15.38
75	FB.18.15.75
100	FB.18.15.100



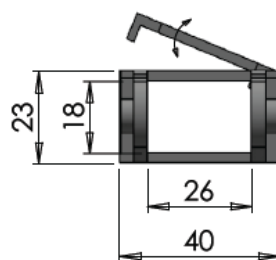
Raio (R)	Código
28	FA.18.18.28
38	FA.18.18.38
75	FA.18.18.75
100	FA.18.18.100



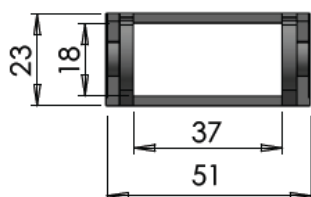
Raio (R)	Código
28	FB.18.18.28
38	FB.18.18.38
75	FB.18.18.75
100	FB.18.18.100



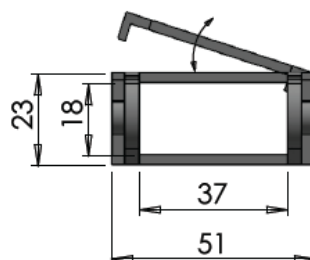
Raio (R)	Código
28	FA.18.26.28
38	FA.18.26.38
75	FA.18.26.75
100	FA.18.26.100



Raio (R)	Código
28	FB.18.26.28
38	FB.18.26.38
75	FB.18.26.75
100	FB.18.26.100



Raio (R)	Código
28	FA.18.37.28
38	FA.18.37.38
75	FA.18.37.75
100	FA.18.37.100

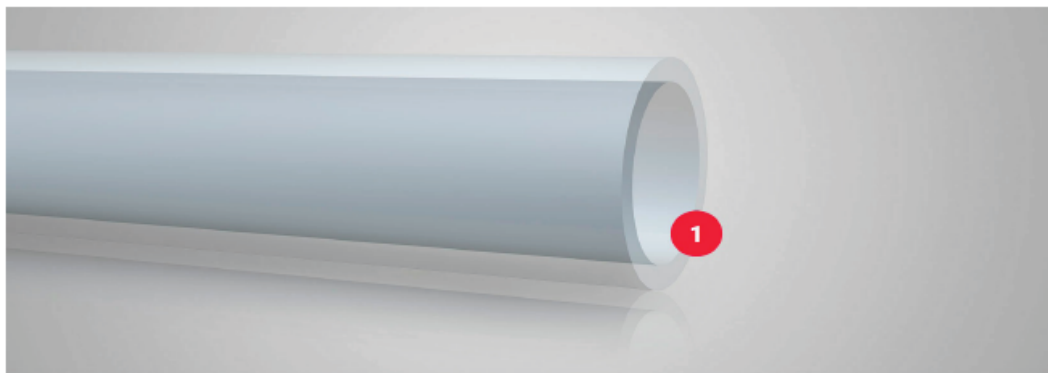


Raio (R)	Código
28	FB.18.37.28
38	FB.18.37.38
75	FB.18.37.75
100	FB.18.37.100

A.9 MANGUERAS DE PVC - ESPIROFLEX



Espirocristal



Tubo flexible unicapa fabricado por extrusión en PVC transparente plastificado

Descripción

Atóxica y de uso alimentario según reglamentos europeos CE 1935/2004 y UE 10/2011

Gran flexibilidad

Alta resistencia a la rotura por alargamiento

Soporta temperaturas entre -10°C y $+60^{\circ}\text{C}$

Tubería con buena resistencia química, asociada a las propiedades habituales del PVC

Aplicaciones

Trasiego de líquidos alimentarios que requieran simulantes alimentarios A, B, y C en condiciones OM2 según reglamento UE 10/2011, tales como vinos, mostos, cervezas, vinagres y líquidos alcohólicos hasta 20°

Industria en general y como nivel en el sector de la construcción

Instalaciones sanitarias

Datos técnicos

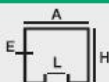
Ø int (mm) Ø ext (mm) Peso (gr/m) Radio de curvatura (mm)

5	7	24	15
5	8	39	16
5	9	54	18
5	13	139	28
6	8	28	16
6	9	45	18
6	9,5	52	20
6	10	62	20
6	12	104	25
7	9	32	20
7	10	49	20
7	11	70	22
7	12	92	25
7	14	142	30
7,5	9,5	34	20

A.10 CARRO Y RIEL PORTA CARRO - ROMA

CORREDIZO SUPERIOR

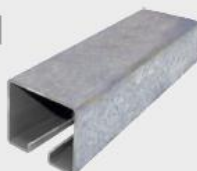
Rieles y guías



Mecanizados con tecnología italiana de excelente calidad

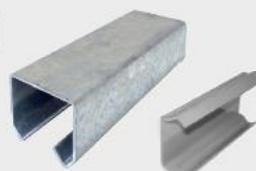
Riel Plano

ART	H	A	E	L
164	30	36	1,10	10
168	24	30	0,90	8
170	38	44	1,60	13
172	47	50	1,80	13
174	55	55	2,00	13



Riel Biselado

ART	H	A	E	L
140 *	53	21	2,10	
142 *	28	14	1,10	
168 Bis	25	29	0,90	10
164 Bis	30	38	1,10	12
170 Bis	38	43	1,60	13
180 Bis	65	57	2,00	13



Riel 140 */ 142 *

Guía

ART	H	A	E
12	13	14	0,90
21	18	23	0,90
84	45	35	1,60



Acero inoxidable
Hierro galvanizado

Carros



ART. 142-A
Ø 25 - 30 kg*
ART. 142-N
Ø 25 - 30 kg*



ART. 144-A
Ø 25 - 30 kg*
ART. 144-N
Ø 25 - 30 kg*



ART. 140-A
Ø 32 - 100 kg*

Acoples y soportes para rieles



Acople Soporte 170
(250 kg)



Acople 164 bis
(300 kg)



Soporte 164 bis
(250 kg)



Soporte 168
(250 kg)



ART. 170-A
Ø 32 - 150 kg*

ART. 164-A
Ø 25 - 80 kg*

ART. 168-A
Ø 20 - 60 kg*



ART. 172-A
Ø 38 - 200 kg*



ART. 174-A
Ø 45 - 300 kg*



ART. 170-A-Bis
Ø 32 - 150 kg*

ART. 164-A-Bis
Ø 25 - 80 kg*

ART. 168-A-Bis
Ø 20 - 60 kg*



ART. 180-A-Bis-BL
Ø 55 - 400 kg*



ART. 170-BL
Ø 28 - 200 kg*

ART. 172-BL
Ø 32 - 250 kg*

ART. 174-BL
Ø 40 - 350 kg*

Referencias:

A= Acero

Bis= Biselado

BL= Rulmán blindado

* Peso soportado por 2 unidades



BL - Ruedas montadas
en rulmanes blindados



A.11 DRIVER MOTOR VERTICAL - LEADSHINE

User Manual

DM542E

2-Phase Digital Stepper Drive



2. Specifications

2.1 Electrical Specifications

Parameters	DM542E			
	Min	Typical	Max	Unit
Output Current	1.0	-	4.2 (3.0 RMS)	A
Supply Voltage	20	24 - 48	50	VDC
Logic signal current	7	10	16	mA
Pulse input frequency	0	-	200	kHz
Minimal pulse width	2.5	-	-	μS
Minimal direction setup	5.0	-	-	μS
Isolation resistance	500			MΩ

2.2 Environment

Cooling	Natural Cooling or Forced cooling	
Operating Environment	Environment	Avoid dust, oil fog and corrosive gases
	Ambient Temperature	0°C — 65°C (32°F - 149°F)
	Humidity	40%RH — 90%RH
	Operating Temperature	0°C — 50°C (32°F - 122°F)
	Vibration	10-50Hz / 0.15mm
Storage Temperature	-20°C — 65°C (-4°F - 149°F)	
Weight	Approx. 227g (8 oz)	



3.1 Connector P1 Configurations

Pin Function	Details
PUL+	<u>Pulse signal:</u> Pulse active at rising edge; 4-5V when PUL-HIGH, 0-0.5V when PUL-LOW. Minimal pulse width of 2.5 μ s. Add a resistor for current-limiting at +12V or +24V input logic voltage (1K for +12V, 2k for +24V). The same as DIR and ENA signals.
PUL-	
DIR+	<u>DIR signal:</u> This signal has low/high voltage levels to represent two directions of motor rotation.. Minimal direction setup time of 5 μ s. Also swapping the connection of two wires of a coil (e.g. A+ and A-) to the drive will reverse motor direction.
DIR-	
ENA+	<u>Enable signal:</u> This signal is used for enabling/disabling the drive. High level +5V (NPN control signal) for enabling the drive and low level for disabling the drive. PNP and Differential control signals are on the contrary, namely Low level for enabling. By default it is left UNCONNECTED (ENABLED).
ENA-	

3.2 Connector P2 Configurations

Pin Function	Details
GND	Power supply ground connection.
+V	Power supply positive connection. Suggest 24-48VDC power supply voltage
A+, A-	Motor Phase A connections. Connect motor A+ wire to A+ Pin; motor A- wire to A-
B+, B-	Motor Phase B connections. Connect motor B+ wire to B+ Pin; motor B- wire to B-

7.1 Microstep Resolution Configurations

Microstep resolution is set by SW5, 6, 7, 8 of the DIP switches as shown in the following table:

Microstep	Steps/rev.(for 1.8°motor)	SW5	SW6	SW7	SW8
2	400	OFF	ON	ON	ON
4	800	ON	OFF	ON	ON

7.2.1 Dynamic Current Configurations

Peak Current	RMS Current	SW1	SW2	SW3
1.00A	0.71A	ON	ON	ON
1.46A	1.04A	OFF	ON	ON
1.91A	1.36A	ON	OFF	ON
2.37A	1.69A	OFF	OFF	ON
2.84A	2.03A	ON	ON	OFF
3.31A	2.36A	OFF	ON	OFF
3.76A	2.69A	ON	OFF	OFF
4.20A	3.00A	OFF	OFF	OFF

A.12 DRIVER MOTOR HORIZONTAL - LEADSHINE

User Manual

DM556E

2-Phase Digital Stepper Drive



2. Specifications

2.1 Electrical Specifications

Parameters	DM556E			
	Min	Typical	Max	Unit
Output Current	1.8	-	5.6(4.0 RMS)	A
Supply Voltage	20	24 - 48	50	VDC
Logic signal current	7	10	16	mA
Pulse input frequency	0	-	200	kHz
Minimal pulse width	2.5	-	-	μS
Minimal direction setup	5.0	-	-	μS
Isolation resistance	500			MΩ

2.2 Environment

Cooling	Natural Cooling or Forced cooling	
Operating Environment	Environment	Avoid dust, oil fog and corrosive gases
	Ambient Temperature	0°C — 65°C (32°F - 149°F)
	Humidity	40%RH—90%RH
	Operating Temperature	0°C — 50°C (32°F - 122°F)
	Vibration	10-50Hz / 0.15mm
Storage Temperature	-20°C — 65°C (-4°F - 149°F)	
Weight	Approx. 300g (10.6oz)	



3.1 Connector P1 Configurations

Pin Function	Details
PUL+	<u>Pulse signal:</u> Pulse active at rising edge; 4-5V when PUL-HIGH, 0-0.5V when PUL-LOW. Minimal pulse width of 2.5 μ s. Add a resistor for current-limiting at +12V or +24V input logic voltage (1K for +12V, 2k for +24V). The same as DIR and ENA signals.
PUL-	
DIR+	<u>DIR signal:</u> This signal has low/high voltage levels to represent two directions of motor rotation.. Minimal direction setup time of 5 μ s. Also swapping the connection of two wires of a coil (e.g. A+ and A-) to the drive will reverse motor direction.
DIR-	
ENA+	<u>Enable signal:</u> This signal is used for enabling/disabling the drive. High level +5V (NPN control signal) for enabling the drive and low level for disabling the drive. PNP and Differential control signals are on the contrary, namely Low level for enabling. By default it is left UNCONNECTED (ENABLED) .
ENA-	

3.2 Connector P2 Configurations

Pin Function	Details
GND	Power supply ground connection.
+V	Power supply positive connection. Suggest 24-48VDC power supply voltage
A+, A-	Motor Phase A connections. Connect motor A+ wire to A+ Pin; motor A- wire to A-
B+, B-	Motor Phase B connections. Connect motor B+ wire to B+ Pin; motor B- wire to B-

7.1 Microstep Resolution Configurations

Microstep resolution is set by SW5, 6, 7, 8 of the DIP switches as shown in the following table:

Microstep	Steps/rev.(for 1.8°motor)	SW5	SW6	SW7	SW8
2	400	OFF	ON	ON	ON
4	800	ON	OFF	ON	ON

7.2.1 Dynamic Current Configurations

Peak Current	RMS Current	SW1	SW2	SW3
1.8A	1.3A	ON	ON	ON
2.1A	1.5A	OFF	ON	ON
2.7A	1.9A	ON	OFF	ON
3.2A	2.3A	OFF	OFF	ON
3.8A	2.7A	ON	ON	OFF
4.3A	3.1A	OFF	ON	OFF
4.9A	3.5A	ON	OFF	OFF
5.6A	4.0A	OFF	OFF	OFF



A.13 CABLES – MARLEW Y PRYSMIAN



AR MILIAR®
Cuadrete blindado

AR MILIAR®

Instrumentación Electrónica

Par, terna y cuadrete blindado - 300 Volt - 105°C - PLTC - UL 13

Aplicaciones

Circuitos de instrumentación electrónica, señales digitales y analógicas (4-20mA).
Circuitos de seguridad intrínseca. Detección de pérdidas de gas y/o fluidos.
Medición y monitoreo de presión, temperatura, volumen. Monitoreo de señales de alarma.

Características

Temperatura máxima: 105°C de servicio.
Tensión nominal: 300 Volt.
Norma constructiva: UL 13 tipo PLTC - UL 2250 tipo ITC.
Norma de fuego: UL 1685.
Norma de conductores: ASTM B8 clase B.
Código NEC: Art. 725 PLTC - Art. 727 ITC - Art. 800 -Comunicaciones - Art. 501 áreas clasificadas CI1 Div.2 y CI2 Div.2.
Norma hidrocarburos: NFC 32-200 - ASTM D 1239.

Descripción

Conductor: Cobre electrolítico recocido en formación de 7 hilos, clase B.
Aislación: PVC.
Paso del trenzado: 50mm (20 torsiones por metro).
Par sin blindar: Encintado de poliéster.
Blindaje: Cinta helicoidal de aluminio-poliéster más conductor de drenaje de cobre estañado en formación de 7 hilos, clase B.
Cubierta: PVC negro (azul para seguridad intrínseca), no propagante del incendio, resistente a la luz solar e hidrocarburos.
Desgarre: Hilo de poliamida debajo de la cubierta.

Instalación

Montaje: Radio mínimo de curvatura igual a 7 x diámetro exterior del cable para cables sin armar y de 12 x diámetro exterior del cable para cables armados.
Tracción máxima: 5daN/mm² aplicados sobre los conductores de cobre. En cables armados con alambres se puede usar 10daN/mm² aplicados sobre los alambres de la armadura.

Dimensiones y Pesos

Tipo	Calibre AWG		Estándar		
	Conductor	Drenaje	Código	Diámetro exterior mm	Peso kg/km
Par blindado	20	22	AR 5000	5,0	33
Par blindado	18	20	AR 5100	5,7	45
Par blindado	16	18	AR 5200	6,3	61
Par blindado	14	18	AR 5300	7,8	89



XF _ WNe/NBR1

Potencia



**Unipolar o Multipolar blindado - XLPE/PVC - 0,6/1kV - 90°C - ABNT NBR 6251
ABNT NBR 7287**

Aplicaciones

Cuenta con protección electromagnética adicional. Apto para instalaciones aéreas en bandeja, enterrados en ductos o directamente enterrados. Redes fijas de distribución de energía en: fábricas, depósitos, salas de máquinas, distribuidoras de energía, etc.

Características

Temperatura máxima: 90°C de servicio, 130°C de sobrecarga y 250°C de cortocircuito.

Tensión nominal: 600/1000 Volt CA, máxima 1200 Volt CA.

Norma constructiva: ABNT NBR 6251 - ABNT NBR 7287.

Norma de fuego: IEC 60332-3-24 / NBR NM IEC 60332-3-24 - No propagación de incendio.

Norma de conductores: IEC 60228 / NBR NM 280.

Descripción

Conductor: Cobre electrolítico recocido en formación clase 5.

Aislación: XLPE (polietileno reticulado).

Identificación: Negros numerados con identificación alfanumérica.

Blindaje: Cinta de cobre recocido, aplicada helicoidalmente con cobertura 100% y solape del 20%.

Cubierta: PVC negro, no propagante del incendio.

Instalación

Montaje: Radio mínimo de curvatura igual a 12 x diámetro exterior del cable.

Tracción máxima: 5daN/mm² aplicados sobre los conductores de cobre.

Dimensiones y Pesos

Conductores x sección N° x mm ²	Código	Diámetro bajo pantalla mm	Diámetro exterior mm	Peso kg/km
4x1.5	XF 0415 WNe/NBR1	9.1	12.3	214
4x2.5	XF 0425 WNe/NBR1	10.2	13.6	269



XF _ WNe/NBR1®
Multipolar blindado clase 5
(XLPE)

Características eléctricas

Sección nominal mm ²	Resistencia eléctrica a 90°C en CA. Ohm/km	Reactancia inductiva a 50hz		Intensidad admisible en aire (2)			Intensidad admisible en tierra (3)		
		Unipolares (1) Ohm/km	Multipolares Ohm/km	1x (1) Amper	2x Amper	3x (4) Amper	1x (1) Amper	2x Amper	3x (4) Amper
1,5	16,96	0,157	0,100	19,0	22,8	20,0	32,3	32,3	27,6
2,5	10,18	0,145	0,093	25,7	31,4	27,6	42,8	43,7	37,1

(1) Tres Cables unipolares en un plano en contacto entre si.

(2) Cables instalados en aire a 40°C según IEC 364-5-523.

(3) Cables directamente enterrados con temperatura del terreno igual a 25°C y resistividad térmica específica del terreno igual a 1K.m/W.

(4) Los valores informados también corresponden a cables tetrapolares y pentapolares.



FX _ WNe/NBR1

Comando

**Multipolar blindado - XLPE/PVC -0,6/1KV -90°C - ABNT NBR 6251
ABNT NBR 7290**

Aplicaciones

Cuenta con protección electromagnética adicional. Equipos de control, señalización, medición, protección y comandos eléctricos de instalaciones industriales. Apto para instalaciones aéreas en bandeja, enterrados en ductos o directamente enterrados.

Características

Temperatura máxima: 90°C de servicio, 130°C de sobrecarga, 250°C de cortocircuito.

Tensión nominal: 500V c.a. para 1mm² / 1000 V c.a. para demás secciones.

Norma constructiva: ABNT NBR 6251 - ABNT NBR 7290.

Norma de fuego: IEC 60332-3-24 / NBR NM IEC 60332-3-24 - No propagación de incendio.

Norma de conductores: IEC 60228 / NBR NM 280.

Descripción

Conductor: Cobre electrolítico recocido en formación clase 5.

Aislación: XLPE (Polietileno reticulado).

Identificación: Color negro con identificación alfanumerica.

Blindaje: Cinta de cobre recocido, aplicada helicoidalmente con cobertura 100% y solape del 20%.

Cubierta: PVC negro, no propagante del incendio.

Instalación

Montaje: Radio mínimo de curvatura igual a 12 x diámetro exterior del cable.

Tracción máxima: 5daN/mm aplicados sobre los conductores de cobre.

Dimensiones y Pesos

Formación del cable N° Cond. x sección (mm ²)	Código	Diámetro bajo pantalla mm	Diámetro exterior mm	Peso ?kg/km
5 x 1	FX 0510 WNe/NBR1	9.2	12.3	211
7 x 1	FX 0710 WNe/NBR1	10.1	13.2	233

Propiedades Generales

Sección nominal mm ²	Resist. eléctrica 90°C en CA. Ohm/km	Intensidad admisible en aire (*)							
		2x Amper	4x Amper	7x Amper	10x Amper	19x Amper	30x Amper	40x Amper	48x Amper
1	24,86	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1,5	16,96	22,8	20,0	16,0	10,0	10,0	9,0	8,0	7,0
2,5	10,18	31,4	27,6	22,0	13,8	13,8	12,4	11,0	9,6
4	6,31	42,8	36,1	28,9	18,1	18,1	16,2	14,4	12,6

(*) En aire a 40°C, según IEC 364-5-523 y tabla 310, 15 del NEC 2008.



FX _ WNe/NBR1®
Multipolar blindado clase 5



Cable Flexible para tendidos en cañerías; tipo H07V-K

450 / 750 V

IRAM NM 247-3

Características técnicas

Sección nominal	Diámetro máx. de alambres del conductor	Espesor de aislación nominal	Diámetro exterior aprox.	Masa aprox.	Intensidad de corriente admisible en cañerías (3)		Caída de tensión (4)	Resistencia Eléctrica máxima a 20°C y c.c.
mm ²	mm	mm	mm	kg/km	(1) A	(2) A	V/A km	ohm/km
0,75	0,21	0,6	2,3	11	9	8	50	26
1,0	0,21	0,6	2,5	15	11,5	10,5	37	19,5
1,5	0,26	0,7	3,0	20	15	13	26	13,3
2,5	0,26	0,8	3,6	31	21	18	15	7,98
4	0,31	0,8	4,1	45	28	25	10	4,95
6	0,31	0,8	4,7	63	36	32	6,5	3,30
10	0,41	1,0	6,0	107	50	44	3,8	1,91
16	0,41	1,0	7,0	167	66	59	2,4	1,21
25	0,41	1,2	9,6	268	88	77	1,54	0,78
35	0,41	1,2	10,8	361	109	96	1,20	0,554
50	0,41	1,4	12,8	511	131	117	0,83	0,386
70	0,51	1,4	14,6	698	167	149	0,61	0,272
95	0,51	1,6	16,8	899	202	180	0,48	0,206
120	0,51	1,6	19,7	1175	234	208	0,39	0,161

(1) 2 conductores cargados + PE en cañerías embutidas en mampostería, temperatura ambiente 40° C.

(2) 3 conductores cargados + PE en cañerías embutidas en mampostería, temperatura ambiente 40° C.

(3) Para Instalaciones en aire (no contempladas en el Regl. de Instalaciones en Inmuebles de la AEA) considerar los valores (1) y (2)

(4) Cables en contacto en corriente alterna monofásica 50 Hz., $\cos \phi = 0,8$.

RADIOS DE CURVATURA

Los radios de curvatura indicados en la siguiente tabla, son los radios mínimos que el cable puede adoptar en su posición definitiva de servicio y son aplicables a todos los cables de la norma IRAM 2178.

Cables sin Armadura ni blindaje	Cables sin Armadura ni blindaje	Cables sin Armadura ni blindaje	Cables Armados y/o con blindaje cualquiera sea el tipo
Diámetro exterior del cable (mm)	Diámetro exterior del cable (mm)	Diámetro exterior del cable (mm)	
Menos de 25	De 25 a 50	Más de 50	
Radio mínimo de curvatura expresado en múltiplos del diámetro del cable (cuerdas flexibles)			
4 D	5 D	6 D	10 D
Radio mínimo de curvatura expresado en múltiplos del diámetro del cable (cuerdas rígidas)			
6 D	8 D	10 D	10 D



A.14 SENSORES DE POSICIÓN TIPO BARRERA FOTOELÉCTRICA EN HORQUILLA - IFM

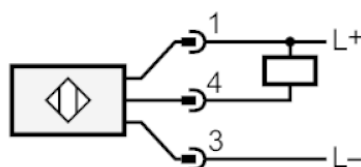
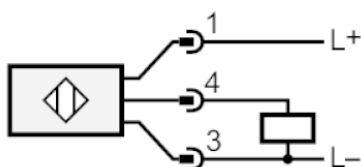
OPU200



Barrera fotoeléctrica en horquilla

OPU-FCKG/IO-LINK/AS

Características del producto		
Tipo de luz		luz infrarroja
Interfaz de comunicación		IO-Link
Dimensiones [mm]		25 x 10 x 45
Datos eléctricos		
Tensión de alimentación [V]		10...30 DC; (tensión de alimentación "supply class 2" según cULus)
Consumo de corriente [mA]		< 30
Clase de protección		III
Protección contra inversiones de polaridad		sí
Tipo de luz		luz infrarroja
Longitud de onda [nm]		860
Salidas		
Alimentación		PNP/NPN; (Detección automática de carga PNP/NPN)
Función de salida		modo luz / oscuridad; (seleccionable)
Caída de tensión máx. de la salida de conmutación DC [V]		1,5
Corriente máxima permanente de la salida de conmutación DC [mA]		100
Frecuencia de conmutación DC [Hz]		10000
Protección contra cortocircuitos		sí
Tipo de protección contra cortocircuitos		pulsada
Resistente a sobrecargas		sí
Rango de detección		
Diámetro del objeto detectable más pequeño [mm]		0,2
Datos mecánicos		
Peso [g]		56
Dimensiones [mm]		25 x 10 x 45
Profundidad de la horquilla D [mm]		17
Ancho de la horquilla W [mm]		10
Materiales		Carcasa: zinc conformado a presión negro anodizado; Lentes: vidrio
Conexión eléctrica		
Conector: 1 x M8		





A.15 FINES DE CARRERA DE PROTECCIÓN - ABB

LS2..M.. Limit Switches

Metal Casing IP67 - 30 mm Width
Pre-Wired

Movement to be detected:

Cable: 5 x 0.75 mm² / 5 x AWG 18
Length: 1 m
(Other lengths see ordering details)

On end



30° Cam Translation



On end



Actuator

Plain plunger

Roller plunger

Cross roller plunger

Steel plain plunger
with fixing nuts

⊕ (N.C. contact with positive opening operation)
Maximum actuation speed
Min. force / torque: - actuation
- positive opening operation

0.5 m/s
15 N
30 N

0.1 m/s
10 N
30 N

0.1 m/s
10 N
30 N

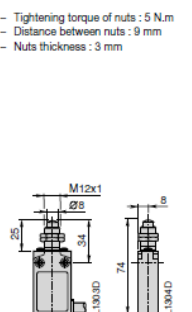
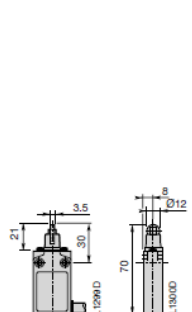
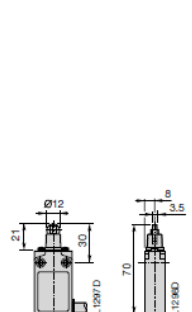
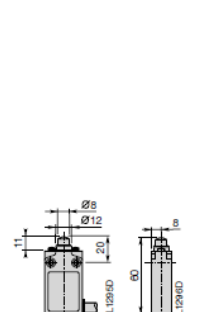
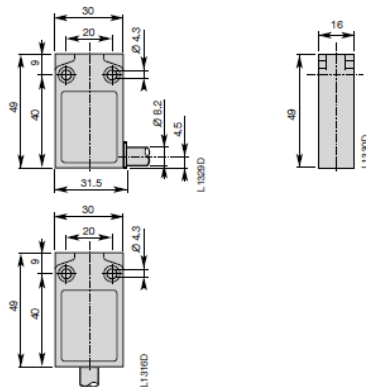
0.5 m/s
15 N
30 N

Additional Technical Data

Cable output left / right code..... ☐ 0				
Cable output bottom code..... ☐ 1				
IEC 20/22 II PVC cable code ☐ P				
UL 62 PVC cable code..... ☐ U				
Type to be completed with head code ☐	Metal plunger ☐ 1 Metal plunger ☐ 6 (with dust protection cup)	Metal roller ☐ 2 Plastic roller ☐ 3 —	Metal roller ☐ 4 Plastic roller ☐ 5 —	— — —
Snap action contacts	Type	LS2 ☐ M1 ☐ B11- ☐ 01	LS2 ☐ M1 ☐ B11- ☐ 01	LS2 ☐ M1 ☐ B11- ☐ 01
13 BN 21 BK 14 BU 22 BK Zb	Operation diagram	0 1.0 1.9 4.0 5.0 mm 21-22 13-14 21-22 13-14	0 1.7 3.3 6.9 8.7 mm 21-22 13-14 21-22 13-14	0 1.7 3.3 6.9 8.7 mm 21-22 13-14 21-22 13-14
Non-overlapping Slow action contacts	Type	LS2 ☐ M1 ☐ D11- ☐ 01	LS2 ☐ M1 ☐ D11- ☐ 01	LS2 ☐ M1 ☐ D11- ☐ 01
13 BN 21 BK 14 BU 22 BK Zb	Operation diagram	0 1.9 3.4 5.0 mm 21-22 13-14 3.2	0 3.3 5.9 8.7 mm 21-22 13-14 5.5	0 3.3 5.9 8.7 mm 21-22 13-14 5.5
Weight (packing per unit)	kg	0.175	0.180	0.180
				0.190

■ Closed contact / □ Open contact

Dimensions (mm)



- Tightening torque of nuts : 5 N.m
- Distance between nuts : 9 mm
- Nuts thickness : 3 mm

A.16 RELES - PHOENIX CONTACT

Sistema de relés industriales: RIFLINE complete

Relés de estado sólido enchufables

Relés de estado sólido enchufables aptos para zócalo de relé RIF-0 y PLC-INTERFA-CE.

Las ventajas:

- Corriente de conmutación hasta 3 A
- Resistentes a RT III (resistentes al lavado)
- Resistentes a vibraciones y choques
- Sin desgaste y de larga duración
- Conmutador de tensión nula en la salida de AC
- Soldables en placa de circuito impreso

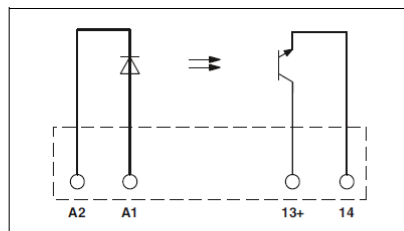
Observaciones:

Para los esquemas dimensionales y taladros de montaje véase la página 401

Al montar relés sobre zócalos de carril DIN o sobre la placa de circuito impreso pueden darse datos limitados, sobre todo en la corriente constante límite y/o el rango de temperatura ambiente. Véase el apartado "Generalidades" en el capítulo "Principios de la técnica de relés" en la página 272



Relé de estado sólido,
salida DC máx. 3 A



Datos técnicos

Datos de entrada		①
Margen admisible (referido a U_N)		0,8 - 1,2
Nivel de conmutación	Señal 1 ("H") [V DC] $\geq$ Señal 0 ("L") [V DC] $\leq$	16 10
Corriente de entrada típica con U_N	[mA]	7
Tiempo de cierre típico con U_N	[μ s]	20
Tiempo de apertura típico con U_N	[μ s]	300
Frecuencia de transmisión $f_{límite}$	[Hz]	300
Datos de salida		
Tensión máxima de ruptura		33 V DC
Tensión de conmutación mínima		3 V DC
Corriente constante límite		3 A (véase curva derating)
Corriente de carga mínima		-
Corriente de cierre máxima		15 A (10 ms)
Corriente de fuga en estado desconectado		-
Ángulo de desfase (cos ϕ)		-
Circuito de salida		2 conductores sin masa
Integral de carga límite		-
Circuito de salida		Protección contra inversión de polaridad, protección contra sobretensiones
Caída de tensión a la corriente continua límite máxima		≤ 150 mV
Datos generales		
Tensión transitoria de dimensionamiento		Aislamiento básico
Tensión de prueba entrada/salida		2,5 kV (50 Hz, 1 min)
Temperatura ambiente (servicio)		-25 °C ... 60 °C
Tipo de funcionamiento nominal		Tiempo de trabajo 100 %
Normas/especificaciones		IEC 60664, EN 50178
Grado de suciedad/categoría de sobretensiones		2 / III
Posición de montaje/montaje		Discrecional/alineables sin separación
Dimensiones	An / Al / Pr	5 mm/28 mm/15 mm

Datos de pedido

Descripción	Tensión de entrada U_N	Tipo	Código	Emb.
Relés de estado sólido enchufables				
Relés de estado sólido de potencia	① 24 V DC	OPT-24DC/ 24DC/ 2	2966595	10
Relés de estado sólido enchufables				
Relé de estado sólido de entrada	① 24 V DC			

Sistema de relés industriales: RIFLINE complete

Relés miniatura de potencia enchufables

Relés enchufables de un conmutador aptos para los zócalos de relé RIF-0 y PLC-INTERFACE.

Las ventajas:

- Contactos de potencia hasta 6 A
- Contacto dorado multicapa o de potencia
- Elevado índice de protección RT III (resistentes al lavado) o RT II para relé con un conmutador con accionamiento manual
- Separación segura según DIN EN 50178 entre bobina y contacto
- Soldables en placa de circuito impreso

Observaciones:

Al sobrepasar los valores máximos indicados para relés de contacto universal multicapa se destruye la capa de oro. En el servicio ulterior, serán válidos los valores máximos del relé con contacto de potencia. En este caso los valores de vida útil pueden ser más bajos que en el contacto de potencia puro.

Para los esquemas dimensionales y taladros de montaje véase la página 400

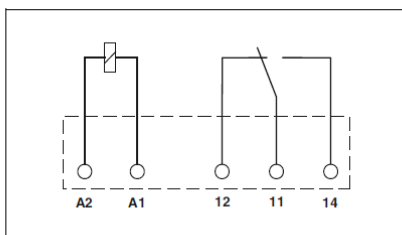
Al montar relés sobre zócalos de carril DIN o sobre la placa de circuito impreso pueden darse datos limitados, sobre todo en la corriente constante límite y/o el rango de temperatura ambiente. Véase el apartado "Generalidades" en el capítulo "Principios de la técnica de relés" en la página 272



Relés con un conmutador,
máx. 6 A

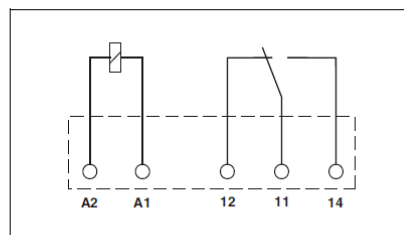


Relés con un conmutador,
con accionamiento manual,
máx. 6 A



Datos técnicos

①	②
Véase el diagrama	
14	7
5	5
2,5	2,5
1 contacto conmutado	
AgSnO	
250 V AC/DC	
5 V (para 100 mA)	
6 A	
10 A (4 s)	
10 mA (para 12 V)	
1 contacto conmutado	
AgSnO, dorado duro	
30 V AC/36 V DC	
100 mV (para 10 mA)	
50 mA	
50 mA	
1 mA (con 24 V)	



Datos técnicos

①	②
Véase el diagrama	
14	7
5	5
2,5	2,5
1 contacto conmutado	
AgSnO	
250 V AC/DC	
5 V (para 100 mA)	
6 A	
10 A (4 s)	
10 mA (para 12 V)	
1 contacto conmutado	
AgSnO, dorado duro	
30 V AC/36 V DC	
100 mV (para 10 mA)	
50 mA	
50 mA	
1 mA (con 24 V)	

Datos de entrada	
Margen admisible (referido a U_N)	
Corriente de entrada típica con U_N	[mA]
Tiempo típico de reacción para U_N	[ms]
Tiempo típico de apertura para U_N	[ms]
Datos de salida	
Tipo de contacto	
Material del contacto	
Tensión máxima de ruptura	
Tensión de conmutación mínima	
Corriente constante límite	
Corriente de cierre máxima	
Corriente de conmutación mínima	
Datos generales	
Tensión de prueba (bobina/contacto)	
Temperatura ambiente (servicio)	
Tipo de funcionamiento nominal	
Vida útil mecánica	
Normas/especificaciones	
Posición de montaje/montaje	
Dimensiones	An / Al / Pr

4 kV AC (50 Hz, 1 min)	
-40 °C ... 85 °C	
Tiempo de trabajo 100 %	
2x 10 ⁷ periodicidades de cambio de estado	
IEC 60664 , EN 50178 , EN 61810-1	
Discrecional/alineables sin separación	
5 mm/28 mm/15 mm	

4 kV AC (50 Hz, 1 min)	
-40 °C ... 85 °C	
Tiempo de trabajo 100 %	
1x 10 ⁷ operaciones	
IEC 60664 , EN 50178 , EN 61810-1	
Discrecional/alineables sin separación	
5 mm/28 mm/16 mm	

Datos de pedido

Tipo	Código	Emb.
REL-MR- 12DC/21	2961150	10
REL-MR- 24DC/21	2961105	10
REL-MR- 12DC/21AU	2961163	10
REL-MR- 24DC/21AU	2961121	10

Datos de pedido

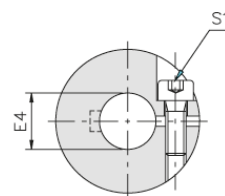
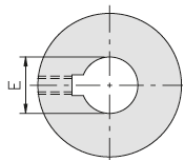
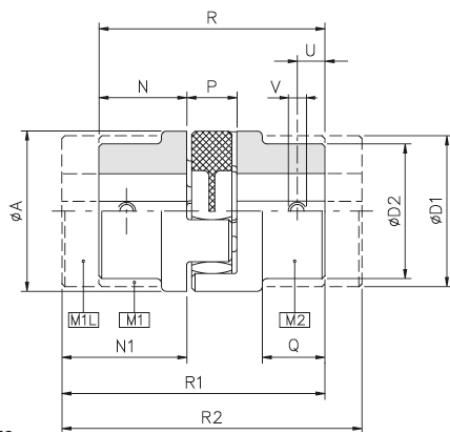
Tipo	Código	Emb.
REL-MR- 12DC/21/MS	2909641	10
REL-MR- 24DC/21/MS	2909642	10
REL-MR- 12DC/21AU/MS	2909644	10
REL-MR- 24DC/21AU/MS	2909645	10

Descripción	Tensión de entrada U_N
Relés miniatura de potencia enchufables con contactos de potencia	
①	12 V DC
②	24 V DC
Relés miniatura de potencia enchufables, con contactos dorados multicapa	
①	12 V DC
②	24 V DC



A.17 ACOPLAMIENTOS - COMINTEC

GAS/SG-ST - acoplamiento de estrella sin juego «de acero»: datos técnicos



En versión M1

DIMENSIONES



PARA LOS DATOS
TÉCNICOS, CONSULTAR
EL FOLLETO ESPECÍFICO

Tamaño	A	D1	D2	EH7 máx.		E4 H7 máx.		N	N1	P	Q	R	R1	R2	U	V
				M1/M1L	M2	mín.	máx.									
01 (14)	30	30	-	16	-	6	15	11	19	12	-	35	42,5	50	5	M4
00 (19)	40	40	32	25	20	8	20	25	37	16	16,5	66	78	90	10	M5
0 (24)	55	53	40	35	26	10	30	30	50	18	20	78	98	118	10	M5
1 (28)	65	63	48	40	32	14	35	35	60	20	24	90	115	140	15	M8

CARACTERÍSTICAS TÉCNICAS

Tamaño	Par [Nm]	Peso [kg]				Inercia [kgm ²]				Velocidad máx. [rpm]	Bloqueo con mordaza	
		M1	M1L	M2	Estrella	M1	M1L	M2	Estrella		Tomillo S1	Par de apriete S1 [Nm]
01 (14)	: página 27	0,06	0,1	-	0,005	0,00001	0,00001	-	0,0000005	25000	M4	5
00 (19)		0,2	0,3	0,2	0,009	0,00005	0,00007	0,00003	0,0000015	19000	M5	10,5
0 (24)		0,4	0,8	0,3	0,020	0,00020	0,00029	0,00014	0,0000080	13500	M6	17,5
1 (28)		0,7	1,3	0,5	0,030	0,00042	0,00066	0,00027	0,0000180	11800	M8	28

PAIRES TRANSMISIBLES CON BLOQUEO CON MORDAZA TIPO B

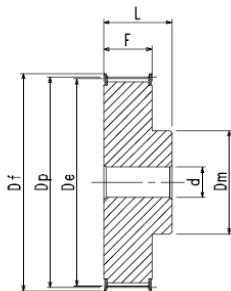
Pares transmisibles [Nm] en relación con el ø del orificio acabado [mm]																													
Tamaño	6	8	10	11	12	14	15	16	18	19	20	22	24	25	28	30	32	35	38	40	42	45	48	50	55	60	65	70	75
01 (14)	21	23	24	25	26	27	28	▲ 29																					
00 (19)		45	47	49	50	52	53	55	57	58	60																		
0 (24)			77	78	80	83	84	86	88	90	91	94	97	98	103	106													
1 (28)						158	161	164	169	171	174	179	184	187	194	199	204	212											

ELEMENTO ELÁSTICO SG: CARACTERÍSTICAS TÉCNICAS

Tamaño	Dureza	Par		Desalineaciones				Rigidez		
		nom [Nm]	máx. [Nm]	angular α [°]	axial X [mm]		radial K [mm]	torsional estática R _{stat} [Nm/Rad]	torsional dinámica R _{din} [Nm/rad]	radial R _r [N/mm]
					GAS/SG	-/DBSE				
00 (19)	92 Sh-A	10	20	1°	+ 1,2 - 0,5	+ 1,2 - 1	0,10	815	1900	1250
	98 Sh-A	17	34	0° 54'			0,06	980	2340	2000
	64 Sh-D	21	42	0° 48'			0,04	1450	4450	2950

A.18 POLEAS SÍNCRONAS - POGGI

Poleas dentadas GT® macizas



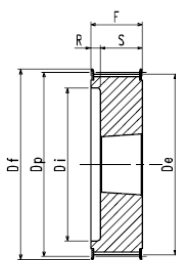
6F

GT® 5MR 15

Passo - Pitch - Teilung - Pas - Paso 5 mm

Materiale Material Werkstoff Matériel Material	Codice Item number Codierung Code Código	Descrizione Designation Bezeichnung Désignation Referencia	N° denti No. of teeth Anzahl der Zähne Nombre de dents Cantidad de dientes	Dp mm	De mm	Df mm	Dm mm	Di mm	F mm	L mm	d mm	Peso Weight Gewicht Poids Peso kg
Acciaio Steel Stahl Acier Acero	32E021015	21 5MR 15 - 6F	21	33,42	32,28	38,0	24,0	-	20,5	26,0	6	0,130
	32E022015	22 5MR 15 - 6F	22	35,01	33,87	38,0	25,5	-	20,5	26,0	6	0,140
	32E024015	24 5MR 15 - 6F	24	38,20	37,06	42,0	27,0	-	20,5	28,0	6	0,180

Poleas dentadas HTD® para casquillo cónico **POGGILOCK®**



5F

HTD® 8M 20

Passo - Pitch - Teilung - Pas - Paso 8 mm

Materiale Material Werkstoff Matériel Material	Codice Item number Codierung Code Código	Descrizione Designation Bezeichnung Désignation Referencia	Bussola Bush Buchse Moyeu Casquillo	Foro Bore Bohrung Alésage Agujero max mm	Dp mm	De mm	Df mm	Dm mm	Di mm	F mm	S mm	R mm	Peso Weight Gewicht Poids Peso kg
Acciaio Steel Stahl Acier Acero	33G022020P	PL 22 8M 20 - 5F	1008	25	56,02	54,65	60,0	-	41,0	28,0	22,0	6,00	0,24
	33G024020P	PL 24 8M 20 - 5F	1108	28	61,12	59,75	66,0	-	42,0	28,0	22,0	6,00	0,30
	33G026020P	PL 26 8M 20 - 5F	1108	28	66,21	64,84	71,0	-	46,0	28,0	22,0	6,00	0,36

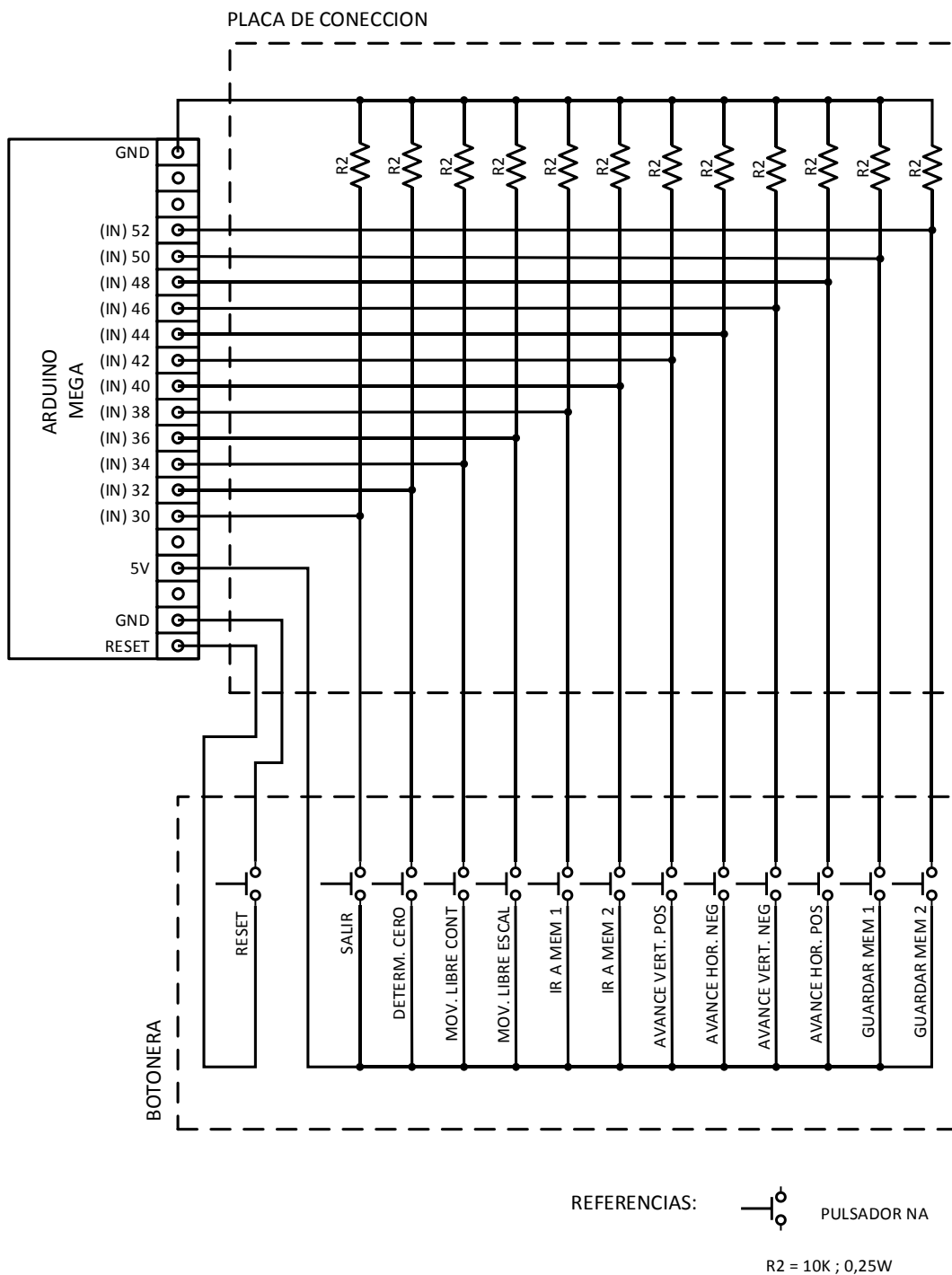


ANEXO B

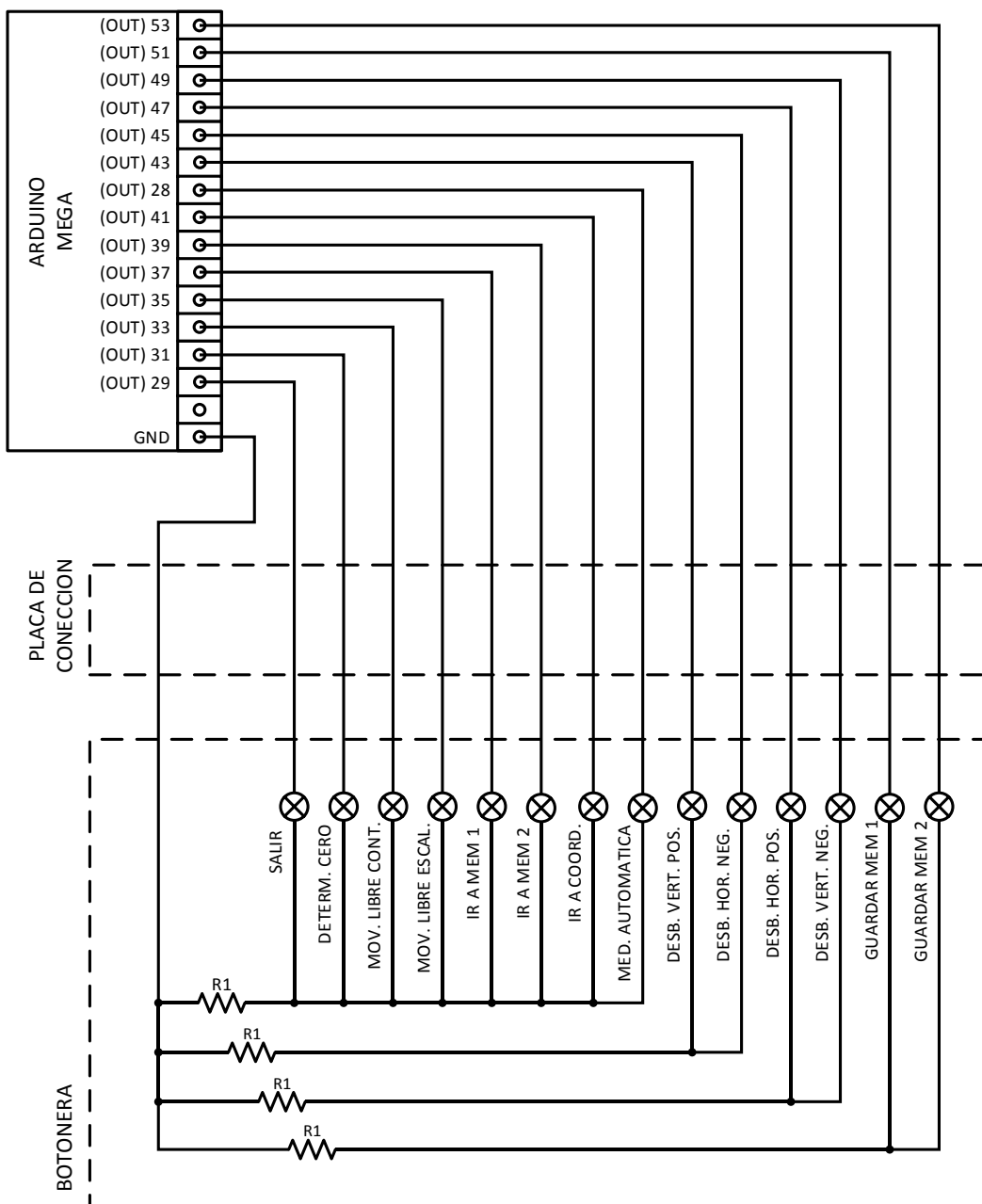
CIRCUITOS ELÉCTRICOS Y ELECTRÓNICOS



B.1 PULSADORES



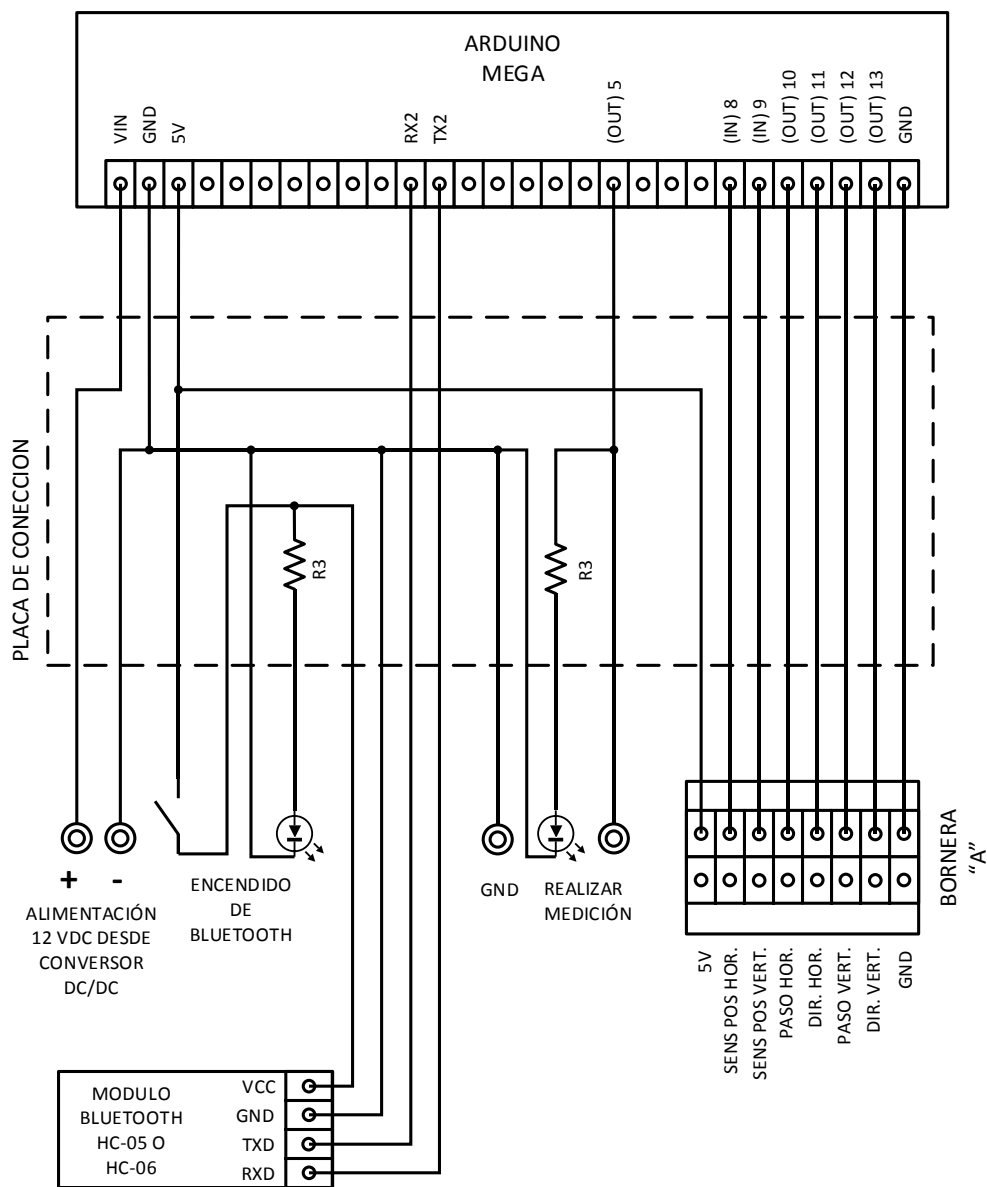
B.2 INDICADORES



REFERENCIAS: R1 = 0,47K ; 0,25W

⊗ DIODO LED 5mm DIFUSO

B.3 CONEXIÓN CON DISPOSITIVOS EXTERNOS



REFERENCIAS:



DIODO LED 5mm DIFUSO



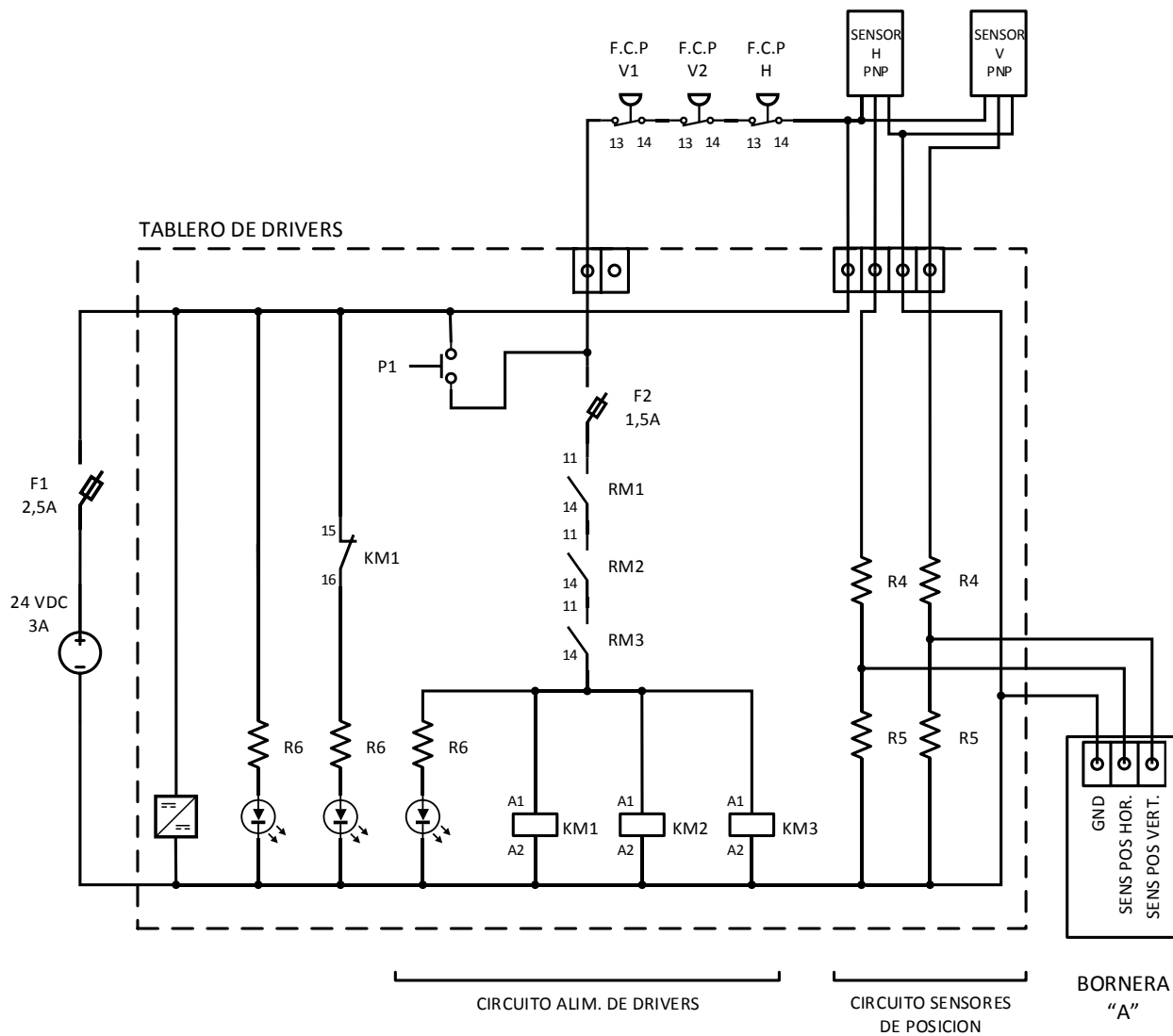
CONECTOR JACK BANANA

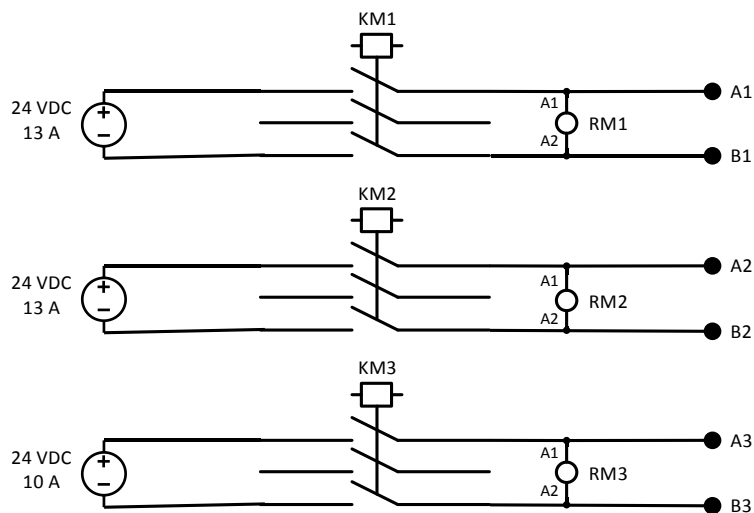
SENS POS: SENSOR DE POSICIÓN

R3 = 0,47K ; 0,25W



B.4 CIRCUITO DE COMANDO – ALIMENTACIÓN DE DRIVERS,
PROTECCIÓN CONTRA COLISIONES Y DETERMINACIÓN DEL PUNTO DE REFERENCIA





REFERENCIAS:



SENSOR DE POSICIÓN IFM OPU200
BARRERA FOTOELÉCTRICA TIPO HORQUILLA
PNP MODO LUZ
CABLE DE CONEXIÓN IFM EVC 142



FIN DE CARRERA DE PROTECCIÓN – NC
CON ACTUADOR TIPO EMBOLO DE ROLDANA
(V1 Y V2 ACTUADOR A 0°; H ACTUADOR A 90°)

KM1;
KM2;
KM3: CONTACTORES 20A AC1 BOB. 24VDC AUX: 1NC

RM1;
RM2;
RM3: RELÉ MINIATURA DE POTENCIA 6A $U_n=24VDC$
ó RELÉ ESTADO SOLIDO DE POTENCIA 3A $U_n=24VDC$

CABLEADO DE ALIMENTACIÓN DE DRIVER:
CABLE IRAM 247-3 PRYSMIAN 2,5MM²

CABLEADO DE CONEXIÓN DE SENSORES Y FINES DE CARRERA:
CABLE IRAM 247-3 PRYSMIAN 1,5MM²



PULSADOR NA



BORNERAS



DIODO LED 5mm DIFUSO



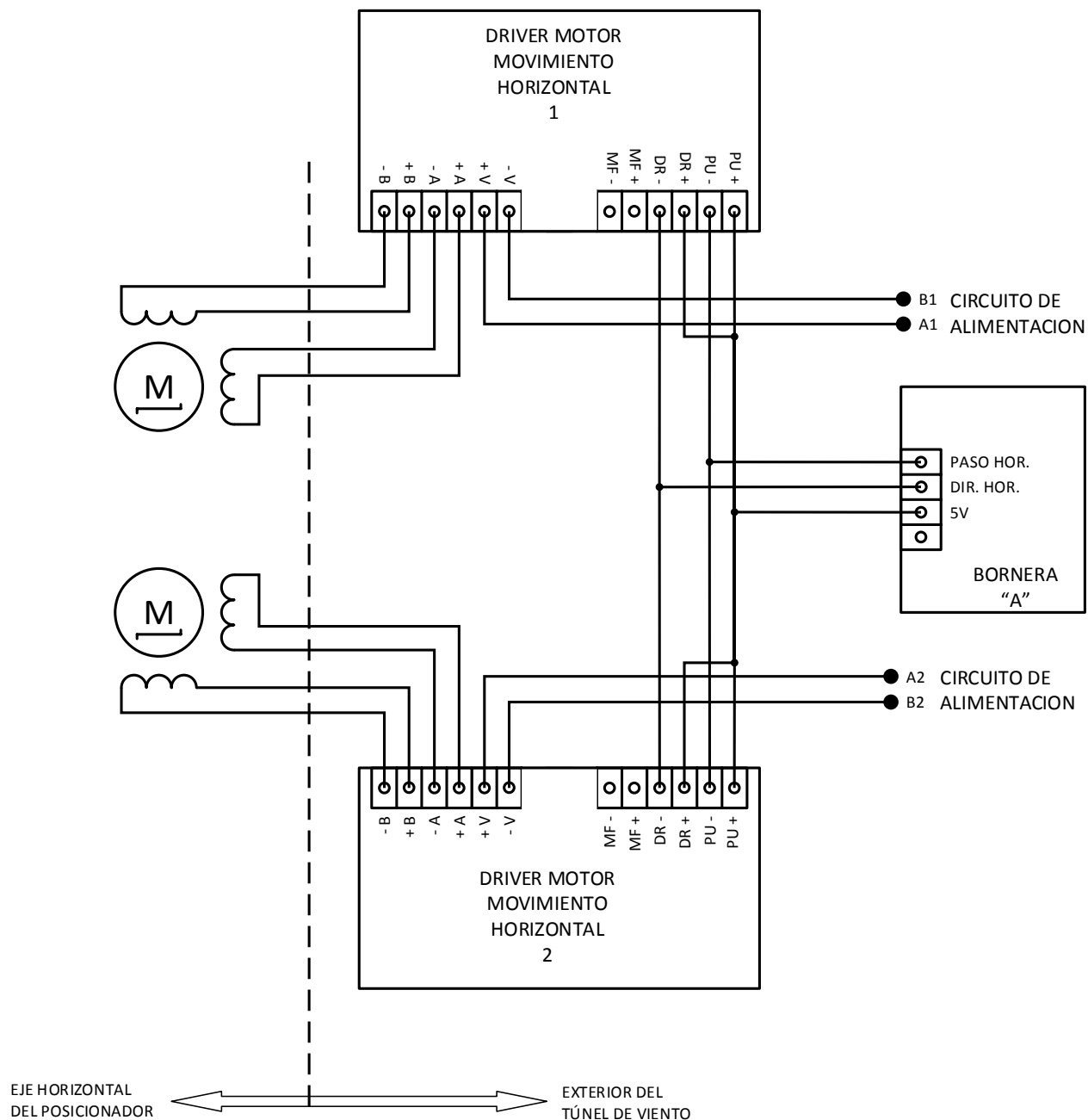
Convertor DC/DC 24V a 12V
(Alimentación de Arduino)

R4 = 4K7 ; 0,25W

R5 = 1K ; 0,25W

R6 = 2K2 ó 2K7 ; 0,5W

B.5 DRIVERS Y MOTORES HORIZONTALES



REFERENCIAS:

CABLEADO DE ALIMENTACIÓN A MOTORES:
CABLE BLINDADO MARLEW XF 0415 WNe/NBR1 4x1,5MM2 ó
CABLE BLINDADO MARLEW FX 0510 WNe/NBR1 5x1MM2

CABLEADO DE SEÑAL:
2 CABLES DE 1 PAR BLINDADO MARLEW AR 5200

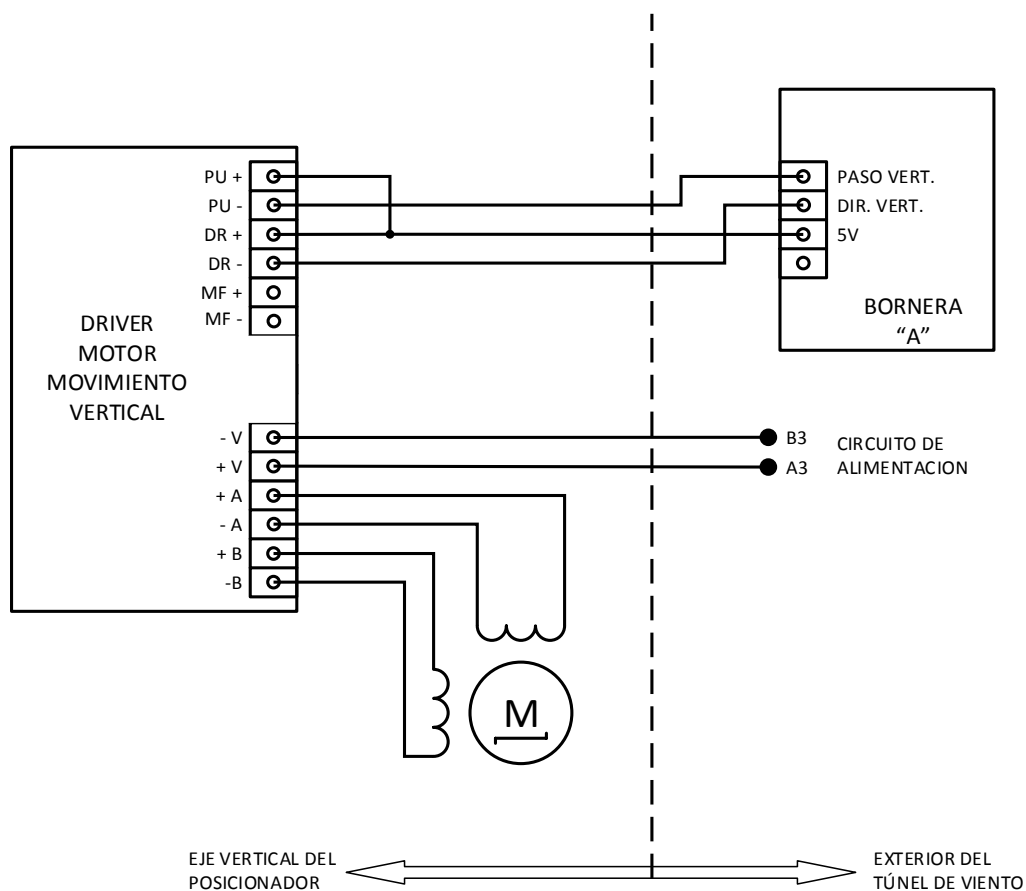
CABLEADO DE ALIMENTACIÓN DE DRIVER:
CABLE IRAM 247-3 PRYSMIAN 2,5MM2



MOTOR PASO A PASO
LEADSHINE D57CM31
3,1 NM

DRIVERS: LEADSHINE
DM556E

B.6 DRIVERS Y MOTORES VERTICALES



REFERENCIAS:

CABLEADO DE ALIMENTACIÓN A MOTORES:
CABLE BLINDADO MARLEW XF 0415 WNe/NBR1 4x1,5MM2 ó
CABLE BLINDADO MARLEW FX 0510 WNe/NBR1 5x1MM2

CABLEADO DE SEÑAL:
2 CABLES DE 1 PAR BLINDADO MARLEW AR 5200

CABLEADO DE ALIMENTACIÓN DE DRIVER:
CABLE IRAM 247-3 PRYSMIAN 2,5MM2



MOTOR PASO A PASO
LEADSHINE D57CM21
2,1 NM

DRIVERS: LEADSHINE
DM542E



ANEXO C

PROGRAMACIÓN DE ARDUINO



```
#include <EEPROM.h>                // LIBRERIA PARA EL USO DE LA MEMORIA EEPROM

/// ----- VARIABLES DE AJUSTE -----

long nvgMax;                        // MAXIMO DESPLAZAMIENTO EN PASOS EN EL MOVIMIENTO VERTICAL. DATO
                                    // INGRESADO DESDE LabView. (4.500 en prototipo) (APROX 50.000 en posicionador)
long nhgMax;                        // MAXIMO DESPLAZAMIENTO EN PASOS EN EL MOVIMIENTO HORIZONTAL. DATO
                                    // INGRESADO DESDE LabView. (5.500 en prototipo) (APROX 24.000 en posicionador)
long NumPasosXMetroH;               // CANTIDAD DE PASOS PARA UN DESPLAZAMIENTO HORIZONTAL DE UN
                                    // METRO. DATO INGRESADO DESDE LabView.(10.000 en prototipo) (APROX
                                    // 22.700 en posicionador)
long NumPasosXMetroV;               // CANTIDAD DE PASOS PARA UN DESPLAZAMIENTO VERTICAL DE UN
                                    // METRO. DATO INGRESADO DESDE LabView.(10.000 en prototipo) (APROX
                                    // 16.600 en posicionador)
float ConversorDistPasosVert;       // FACTOR PARA CONVERTIR Nº DE PASOS A DISTANCIAS EN mm EN EL
                                    // MOVIMIENTO VERTICAL.
float ConversorDistPasosHor;        // FACTOR PARA CONVERTIR Nº DE PASOS A DISTANCIAS EN mm EN EL
                                    // MOVIMIENTO HORIZONTAL.
long HMaxmm;                       // MAXIMO DESPLAZAMIENTO EN mm EN EL MOVIMIENTO HORIZONTAL.
long VMaxmm;                       // MAXIMO DESPLAZAMIENTO EN mm EN EL MOVIMIENTO VERTICAL.

char DatosVariablesAjuste [25];     // ALMACENA LAS VARIABLES DE AJUSTE ENVIADOS DESDE LABVIEW EN FORMA DE
                                    // CADENA DE CARACTERES . SON 24 VALORES PERO SE COLOCA DE EXTENSION 25
                                    // PARA EL CARACTER NULO AL FINAL QUE INDICA EL FIN DE LA CADENA
char datos3 [7];                   // EXTRAE VALORES DEL ARRAY "DatosVariablesAjuste[]" PARA CONVERTIRLOS EN
                                    // VALORES NUMERICOS Y ASIGNARLOS A CADA VARIABLE. 6 DIGITOS POR VARIABLE MÁS
                                    // EL CARACTER NULO

/// ----- VARIABLES FIJAS -----

/// ENTRADAS DIGITALES ///

int DeterminarCero = 32;            // ENTRADA DE PULSADOR - INICIA FUNCION 1 "DETERMINAR CERO" (pull down)
int MovContinuo = 34;               // ENTRADA DE PULSADOR - INICIA FUNCION 2 "MOVIMIENTO LIBRE CONTINUO" (pull down)
int MovEscalonado = 36;             // ENTRADA DE PULSADOR - INICIA FUNCION 3 "MOVIMIENTO LIBRE ESCALONADO" (pull down)
int IrAMemoria1 = 38;               // ENTRADA DE PULSADOR - INICIA FUNCION 4 "IR A MEMORIA 1" (pull down)
int IrAMemoria2 = 40;               // ENTRADA DE PULSADOR - INICIA FUNCION 5 "IR A MEMORIA 2" (pull down)
int Salir = 30;                     // ENTRADA DE PULSADOR - FINALIZA LA FUNCION EN EJECUCIÓN (pull down)

int EntAvanceVert = 42;             // ENTRADA DE PULSADOR - ORDENA AVANCE VERTICAL POSITIVO EN FUNCIONES
                                    // "MOVIMIENTO LIBRE" (pull down)
int EntRetrocesoVert = 46;          // ENTRADA DE PULSADOR - ORDENA AVANCE VERTICAL NEGATIVO EN FUNCIONES
                                    // "MOVIMIENTO LIBRE" (pull down)
```



```
int EntAvanceHor = 48;    // ENTRADA DE PULSADOR - ORDENA AVANCE HORIZONTAL POSITIVO EN FUNCIONES
                          // "MOVIMIENTO LIBRE" (pull down)

int EntRetrocesoHor = 44; // ENTRADA DE PULSADOR - ORDENA AVANCE HORIZONTAL NEGATIVO EN FUNCIONES
                          // "MOVIMIENTO LIBRE" (pull down)

int Memoria1 = 50;       // ENTRADA DE PULSADOR - GUARDA LA POSICION ACTUAL EN LA MEMORIA 1 (pull down)
int Memoria2 = 52;       // ENTRADA DE PULSADOR - GUARDA LA POSICION ACTUAL EN LA MEMORIA 2 (pull down)

int SensorPosV = 9;      // ENTRADA SENSOR DE POSICIÓN VERT. - al detectar (SOMBRA) envía 0V (LOW)(pull down modo luz)
int SensorPosH = 8;      // ENTRADA SENSOR DE POSICIÓN HOR. - al detectar (SOMBRA) envía 0V (LOW)(pull down modo luz)

/// SALIDAS DIGITALES ///

int SalPasoVert = 12;    // SALIDA A DRIVER - INDICA UN PASO VERTICAL
int SalDirPosVert = 13;  // SALIDA A DRIVER - INDICA DIRECCION DEL PASO VERTICAL - "HIGH" INDICA AVANCE POSITIVO
int SalPasoHor = 10;     // SALIDA A DRIVER - INDICA UN PASO HORIZONTAL
int SalDirPosHor = 11;   // SALIDA A DRIVER - INDICA DIRECCION DEL PASO HOR. - "HIGH" INDICA AVANCE POSITIVO

int LuzSalir = 29;       // SALIDA A LUZ - INDICA QUE NO SE ESTA EJECUTANDO NINGUNA FUNCION
int LuzDeterminarCero = 31; // SALIDA A LUZ - INDICA EJECUCIÓN LA FUNCION "DETERMINAR CERO"
int LuzMovContinuo = 33; // SALIDA A LUZ - INDICA EJECUCIÓN LA FUNCION "MOVIMIENTO LIBRE CONTINUO"
int LuzMovEscalonado = 35; // SALIDA A LUZ - INDICA EJECUCIÓN LA FUNCION "MOVIMIENTO LIBRE ESCALONADO"
int LuzIrAMemoria1 = 37; // SALIDA A LUZ - INDICA EJECUCIÓN LA FUNCION "IR A MEMORIA 1"
int LuzIrAMemoria2 = 39; // SALIDA A LUZ - INDICA EJECUCIÓN LA FUNCION "IR A MEMORIA 2"
int LuzIrACoordenada = 41; // SALIDA A LUZ - INDICA EJECUCIÓN LA FUNCION "IR A COORDENADA"
int LuzMedAutomatica = 28; // SALIDA A LUZ - INDICA EJECUCIÓN LA FUNCION "MEDICION AUTOMATICA";
int LuzGuardarEnMemoria1 = 51; // SALIDA A LUZ - INDICA GUARDADO DE POSICION EN MEMORIA 1
int LuzGuardarEnMemoria2 = 53; // SALIDA A LUZ - INDICA GUARDADO DE POSICION EN MEMORIA 2
int RealizarMedicion = 5; // SALIDA A LUZ Y RELE - INDICA MOMENTO PARA REALIZAR EL MUESTREO EN FUNCION
                          // "MEDICIÓN AUTOMÁTICA"

int LuzDVP = 43;         // SALIDA A LUZ - INDICA DESBORDE VERT. POSITIVO EN FUNCION "MOVIMIENTO LIBRE"
int LuzDVN = 49;         // SALIDA A LUZ - INDICA DESBORDE VERT. NEGATIVO EN FUNCION "MOVIMIENTO LIBRE"
int LuzDHP = 47;         // SALIDA A LUZ - INDICA DESBORDE HOR. POSITIVO EN FUNCION "MOVIMIENTO LIBRE"
int LuzDHN = 45;         // SALIDA A LUZ - INDICA DESBORDE HOR. NEGATIVO EN FUNCION "MOVIMIENTO LIBRE"

/// VARIABLES GLOBALES ///

int Limites = 0;         // BANDERA PARA LA ASIGNACIÓN DE LAS VARIABLES DE AJUSTE EN EL void setup ()
int PosicionEEPROM = 0;  // VARIABLE PARA LEER LA EEPROM
boolean CondSiempreVerdadera = 1; // CONDICION SIEMPRE VERDADERA UTILIZADA EN ALGUNOS BUCLE While()

int Funcion = 0;         // GUARDA LA FUNCION A EJECUTARCE
long nvg = 0;            // ALMACENA LA POSICIÓN VERTICAL EN Nº DE PASOS (nvg = 0 INDICA EL CERO VERTICAL
                          // DEL SISTEMA DE REFERENCIA EN USO)
long nhg = 0;            // ALMACENA LA POSICIÓN HORIZONTAL EN Nº DE PASOS (nvg = 0 INDICA EL CERO
```



```
// HORIZONTAL DEL SISTEMA DE REFERENCIA EN USO)
float nvghmm = 0; // ALMACENA LA POSICIÓN VERTICAL EN mm
float nhghmm = 0; // ALMACENA LA POSICIÓN HORIZONTAL EN mm

int DelayAcel[] = { // ARRAY DE PAUSA ENTRE PASOS PARA LA RAMPA DE ACELERACIÓN Y DESACELERACIÓN.
                    // Extension del Array = PasosAcel + 1. (DURACION DE CADA PASO = 2*DelayAcel[])
    1667, // POSICION 0 DEL ARRAY (SOLO PARA ULTIMO PASO DE LA RAMPA DE DESACELERACION) NO IMPORTA SU VALOR.
    1667, // POSICION 1 DEL ARRAY - DELAY ENTRE 1° Y 2° PASO
    1208, // POSICION 2 " 2° Y 3° PASO
    997, // POSICION 3 " 3° Y 4° PASO
    869, // POSICION 4 " 4° Y 5° PASO
    780, // POSICION 5 " 5° Y 6° PASO
    714, // POSICION 6 " 6° Y 7° PASO
    662, // POSICION 7 " 7° Y 8° PASO
    621, // POSICION 8 " 8° Y 9° PASO
    586, // POSICION 9 " 9° Y 10° PASO
    556, // POSICION 10 " 10° Y 11° PASO
    531, // POSICION 11 " 11° Y 12° PASO
    508, // POSICION 12 " 12° Y 13° PASO
    489, // POSICION 13 " 13° Y 14° PASO
    471, // POSICION 14 " 14° Y 15° PASO
    455, // POSICION 15 " 15° Y 16° PASO
    441, // POSICION 16 " 16° Y 17° PASO
    428, // POSICION 17 " 17° Y 18° PASO
    416, // POSICION 18 " 18° Y 19° PASO
    405, // POSICION 19 " 19° Y 20° PASO
    400}; // POSICION 20 " 20° Y 21° PASO (VELOCIDAD FINAL) // n° DE POSICION = VALOR DE VARIABLE
        // "PasosAcelMax"

int PasosAcelMax = 20; // CANTIDAD MAXIMA DE PASOS DE LA RAMPA DE ACELERACION Y DESACELERACION
int PasosAcel = 0; // CANTIDAD DE PASOS DE LA RAMPA DE ACELERACION Y DESACELERACION (su valor real se calcula
                  // antes de cada aceleracion)
```

```
/// VARIABLES PARA FUNCION IrACoordenada () ///
```

```
long nvFinal = 0; // COORDENADA VERTICAL DE DESTINO PARA EJECUTAR LA FUNCION IrACoordenada ()
long nhFinal = 0; // COORDENADA HORIZONTAL DE DESTINO PARA EJECUTAR LA FUNCION IrACoordenada ()
long V = 0; // V y H ALMACENAN LA CANTIDAD DE PASOS VERTICALES Y HORIZONTALES DESDE LA POSICION
long H = 0; // ACTUAL HASTA LA POSICION DE DESTINO EN LA FUNCION IrACoordenada ()
long nAuxFinal1 = 0; // VARIABLE AUXILIAR - N° de pasos del desplazamiento comun
long nAuxFinal2 = 0; // VARIABLE AUXILIAR - N° de pasos del desplazamiento independiente
long nAux = 0; // CONTADOR AUXILIAR
int MovIndep = 0; // VARIABLE AUXILIAR – Puerto de salida correspondiente a la dirección del desplaz.independiente
```

```
/// VARIABLES PARA FUNCION 1 "DETERMINAR CERO" ///
```



```
boolean V1 = 0;           // BANDERA QUE REGISTRA SI SE CONSIGUIÓ IDENTIFICAR EL FIN DE CARRERA VERTICAL
boolean H1 = 0;           // BANDERA QUE REGISTRA SI SE CONSIGUIÓ IDENTIFICAR EL FIN DE CARRERA HORIZONTAL
boolean Mem1Utilizada = 0; // BANDERA QUE REGISTRA SI SE ALMACENÓ UNA POSICION EN LA MEMORIA 1
boolean Mem2Utilizada = 0; // BANDERA QUE REGISTRA SI SE ALMACENÓ UNA POSICION EN LA MEMORIA 2
boolean CeroRealizado = 0; // BANDERA QUE REGISTRA SI SE EJECUTÓ LA FUNCION "DETERMINAR CERO". SEGUN
                           // ELLA SE ACTIVARÁN O NO LAS FUNCIONES "MEDICION AUTOMATICA" E "IR A
                           // COORDENADA", Y LAS FUNCIONES "MOVIMIENTO LIBRE" SE EJECUTARÁN CON O SIN
                           // PROTECCION CONTRA COLISIONES

int NumPasosBloqSensor = 5700; // NUMERO DE PASOS A AVANZAR ANTES DE CONSIDERAR QUE UN SENSORES DE
                               // POSICION SE ENCUENTRA BLOQUEADO (5700 pasos equivale a aprox 25 cm horizontales
                               // y 34 cm verticales)

/// VARIABLES PARA FUNCION 2 "MOVIMIENTO LIBRE CONTINUO" Y 3 "MOVIMIENTO LIBRE ESCALONADO"

int M;                      // VARIABLE INTERMEDIA PARA ALMACENAR LAS ORDENES RECIBIDAS DESDE LAS INTERFASES
int A = 10;                 // VARIABLE PARA ALMACENAR LAS ORDENES RECIBIDAS DESDE LAS INTERFASES
int NumPasosVert = 10;      // NUMERO DE PASOS QUE SE REALIZARÁN EN CADA MOVIMIENTO VERTICAL EN LA FUNCION
                           // "MOVIMIENTO LIBRE ESCALONADO"
int NumPasosHor = 10;      // NUMERO DE PASOS QUE SE REALIZARÁN EN CADA MOVIMIENTO HORIZONTAL EN LA FUNCION
                           // "MOVIMIENTO LIBRE ESCALONADO"

long nv1 = 0;              // MEMORIA 1 VALOR VERTICAL - ALMACENA EL VALOR DE "nvg" CUANDO SE ORDENA GUARDAR
                           // POSICION EN MEMORIA 1
long nh1 = 0;              // MEMORIA 1 VALOR HORIZONTAL - ALMACENA EL VALOR DE "nhg" CUANDO SE ORDENA
                           // GUARDAR POSICION EN MEMORIA 1
long nv2 = 0;              // MEMORIA 2 VALOR VERTICAL - ALMACENA EL VALOR DE "nvg" CUANDO SE ORDENA GUARDAR
                           // POSICION EN MEMORIA 2
long nh2 = 0;              // MEMORIA 2 VALOR HORIZONTAL - ALMACENA EL VALOR DE "nhg" CUANDO SE ORDENA
                           // GUARDAR POSICION EN MEMORIA 2

boolean dvpML = 0;         // BANDERAS QUE INDICAN DESBORDES VERTICALES Y HORIZONTALES, POSITIVOS Y NEGATIVOS
boolean dvnML = 0;         // EN EL MOVIMIENTO LIBRE
boolean dhpML = 0;
boolean dhnML = 0;

/// VARIABLES PARA FUNCION 6 "MEDICION AUTOMATICA" ///

long nvr = 0;              // ALMACENA EL VALOR INICIAL DE "nvg" PARA UTILIZARLO COMO CERO VERTICAL DEL "SISTEMA DE
                           // REFERENCIA RELATIVO"
long nhr = 0;              // ALMACENA EL VALOR INICIAL DE "nhg" PARA UTILIZARLO COMO CERO HORIZONTAL DEL
                           // "SISTEMA DE REFERENCIA RELATIVO"
char DatosMedAut [30];     // STRING QUE ALMACENA LOS DATOS DE DISTANCIAS, CANTIDAD DE MEDICIONES Y TIEMPOS
                           // ENVIADOS DESDE LABVIEW EN FORMA DE CADENA (son 29 caracteres mas el caracter nulo)
char datos1 [6];           // STRING QUE EXTRAER VALORES DE "DatosMedAut[]" PARA CONVERTIRLOS EN VALORES
                           // NUMERICOS Y ASIGNARLOS A CADA VARIABLE. (son 5 valores en realidad)
int posicion = 0;          // SUMADOR PARA EXTRAER CARACTERES DEL STRING "DatosMedAut[]"
```



```
char Modificar;           // VARIABLE PARA CONFIRMAR O MODIFICAR LOS DATOS INGRESADOS

int DistHor;              // DATO INGRESADO - DISTANCIA DESEADA ENTRE MEDICIONES HORIZONTALES EN [MM]
int DistVert;             // DATO INGRESADO - DISTANCIA DESEADA ENTRE MEDICIONES VERTICALES EN [MM]
int mhp;                  // DATO INGRESADO - NUMERO DE MEDICIONES HORIZONTALES POSITIVAS
int mhn;                  // DATO INGRESADO - NUMERO DE MEDICIONES HORIZONTALES NEGATIVAS
int mvp;                  // DATO INGRESADO - NUMERO DE MEDICIONES VERTICALES POSITIVAS
int mvn;                  // DATO INGRESADO - NUMERO DE MEDICIONES VERTICALES NEGATIVAS
unsigned long TiempoMuestreo; // DATO INGRESADO - TIEMPO DE MUESTREO
unsigned long TiempoEstabiliz; // DATO INGRESADO - TIEMPO DE ESTABILIZACION

long ph;                  // NUMERO DE PASOS ENTRE MEDICIONES HORIZONTALES
long pv;                  // NUMERO DE PASOS ENTRE MEDICIONES VERTICALES
unsigned long dhp1;       // dhp1, dhn1, dvp1 y dvn1 CALCULAN EL NUMERO TOTAL DE PASOS EN CADA DIRECCION
long dhn1;                // PARA DETERMINAR SI SE PRODUCEN DESBORDES
unsigned long dvp1;
long dvn1;
unsigned long DuracionMedicAutom; // DURACION TOTAL DEL PROCESO DE MEDICION AUTOMATICA
unsigned long DuracionMedicAutom1; // VARIABLE INTERMEDIA PARA CALCULAR LA DURACION TOTAL
unsigned long DuracionMedicAutom2; // "
unsigned long DuracionMedicAutom3; // "
unsigned long DuracionMedicAutom4; // "
unsigned long DuracionMedicAutom5; // "
unsigned long DuracionMedicAutom6; // "
unsigned long DuracionMedicAutom7; // "

float pvExtra;            // VARIABLE PARA CORREGIR REDONDEO DE DECIMALES EN pv
float phExtra;            // VARIABLE PARA CORREGIR REDONDEO DE DECIMALES EN ph

// SE UTILIZAN TAMBIEN LAS VARIABLES LOCALES BOOLEANAS "D", "E", "F", "G", "MEDAUTOM", "dhp", "dhn", "dvp" y "dvn" Y
// LAS VARIABLES LOCALES ENTERAS "X" E "Y"

/// VARIABLES PARA FUNCION 7 "IR A COORDENADA" ///

char DatosIrACoord [13]; // STRING QUE ALMACENA LOS DATOS DE LAS COORDENADAS DESEADAS EN PASOS O
                          // MM. (son 12 valores en realidad)
char datos2 [7];         // EXTRAE VALORES DEL ARRAY "DatosIrACoord[]" PARA CONVERTIRLOS EN VALORES
                          // NUMERICOS Y ASIGNARLOS A CADA VARIABLE. (6 digitos mas carácter nulo)
char Coordenadas;        // ALMACENA SI LAS COORDENADAS SE ENVIAN EN PASOS, EN MM O SE ORDENA QUE
                          // SE SALGA DE LA FUNCION
long nh3;                // ALMACENA EL VALOR DESEADO DE "nhg" DE LA COORDENADA DESEADA
long nv3;                // ALMACENA EL VALOR DESEADO DE "nvg" DE LA COORDENADA DESEADA
float nh3Extra;          // VARIABLE PARA CORREGIR REDONDEO DE DECIMALES EN nh3 CUANDO SE ENVIAN
                          // LAS COORDENADAS EN mm
float nv3Extra;          // VARIABLE PARA CORREGIR REDONDEO DE DECIMALES EN nv3 CUANDO SE ENVIAN
                          // LAS COORDENADAS EN mm
```



```
void setup() {
```

```
Serial.begin(19200);          // INICIA PUERTO SERIE 0 – USB 19200 BAUDIOS  
Serial2.begin(9600);         // INICIA PUERTO SERIE 2 – BLUETOOTH 9600 BAUDIOS
```

```
/// ENTRADAS DIGITALES///          // ESTABLECE LOS PINES DIGITALES DE ENTRADA
```

```
pinMode (Salir, INPUT);  
pinMode (DeterminarCero, INPUT);  
pinMode (MovContinuo, INPUT);  
pinMode (MovEscalonado, INPUT);  
pinMode (IrAMemoria1, INPUT);  
pinMode (IrAMemoria2, INPUT);  
pinMode (EntAvanceVert, INPUT);  
pinMode (EntRetrocesoVert, INPUT);  
pinMode (EntAvanceHor, INPUT);  
pinMode (EntRetrocesoHor, INPUT);  
pinMode (Memoria1, INPUT);  
pinMode (Memoria2, INPUT);  
pinMode (SensorPosV, INPUT);  
pinMode (SensorPosH, INPUT);
```

```
/// SALIDAS DIGITALES///          // ESTABLECE LOS PINES DIGITALES DE SALIDA
```

```
pinMode (LuzSalir, OUTPUT);  
digitalWrite (LuzSalir, HIGH);          // INDICA QUE EL PROGRAMA EMPEZO A EJECUTARSE  
pinMode (LuzDeterminarCero, OUTPUT);  
pinMode (LuzMovContinuo, OUTPUT);  
pinMode (LuzMovEscalonado, OUTPUT);  
pinMode (LuzIrAMemoria1, OUTPUT);  
pinMode (LuzIrAMemoria2, OUTPUT);  
pinMode (LuzIrACoordenada, OUTPUT);  
pinMode (LuzMedAutomatica, OUTPUT);  
pinMode (RealizarMedicion, OUTPUT);  
pinMode (LuzGuardarEnMemoria1, OUTPUT);  
pinMode (LuzGuardarEnMemoria2, OUTPUT);  
pinMode (SalPasoVert, OUTPUT);  
pinMode (SalDirPosVert, OUTPUT);  
pinMode (SalPasoHor, OUTPUT);  
pinMode (SalDirPosHor, OUTPUT);  
pinMode (LuzDVP, OUTPUT);  
pinMode (LuzDVN, OUTPUT);  
pinMode (LuzDHP, OUTPUT);  
pinMode (LuzDHN, OUTPUT);
```



/// ASIGNACION DE VALORES A LAS VARIABLES DE AJUSTE ///

```
while(Serial.available() < 24){          // SE ESPERA A RECIBIR LAS VARIABLES DE AJUSTE POR PUERTO SERIE 0
                                        // Ó A QUE SE OPRIMAN LOS PULSADORES SALIR Y MOV LIBRE CONT. JUNTOS
if (digitalRead(Salir) == HIGH && digitalRead(MovContinuo) == HIGH){
    Limites = 1;
    break;
}
delay(200);
}

switch(Limites){

    case 0:                            // SI SE RECIBIERON LOS VALORES DE LAS
        memset(DatosVariablesAjuste, 0, sizeof(DatosVariablesAjuste)); // VARIABLES (Limites == 0) ESTOS SE
        posicion = 0;                // ALMACENAN EN LAS VARIABLES DE
        while (Serial.available()>0){ // AJUSTE
            delay (5);
            DatosVariablesAjuste[posicion] = Serial.read();
            posicion++;
        }

        memset(datos3, 0, sizeof(datos3));
        posicion=0;
        for (int i=0; i<6; i++){
            datos3[posicion] = DatosVariablesAjuste[i];
            posicion++;
        }
        nhgMax=atol(datos3);

        memset(datos3, 0, sizeof(datos3));
        posicion=0;
        for (int i=6; i<12; i++){
            datos3[posicion] = DatosVariablesAjuste[i];
            posicion++;
        }
        nvgMax=atol(datos3);

        memset(datos3, 0, sizeof(datos3));
        posicion=0;
        for (int i=12; i<18; i++){
            datos3[posicion] = DatosVariablesAjuste[i];
            posicion++;
        }
        NumPasosXMetroH=atol(datos3);

        memset(datos3, 0, sizeof(datos3));
```



```
    posicion=0;
    for (int i=18; i<24; i++){
        datos3[posicion] = DatosVariablesAjuste[i];
        posicion++;
    }
    NumPasosXMetroV=atol(datos3);

    EEPROM.put (PosicionEEPROM,NumPasosXMetroH);
    PosicionEEPROM += sizeof(int);
    EEPROM.put (PosicionEEPROM,NumPasosXMetroV);
    PosicionEEPROM += sizeof(int);
    EEPROM.put (PosicionEEPROM,nhgMax);
    PosicionEEPROM += sizeof(long);
    EEPROM.put (PosicionEEPROM,nvgMax);
    break;

case 1:                                     // SI SE OPRIMIERON LOS PULSADORES
                                           // (Limites == 1) SE GUARDAN EN LAS
                                           // VARIABLES DE AJUSTE LOS VALORES
                                           // ALMACENADOS EN LA MEMORIA
                                           // EEPROM
    EEPROM.get (PosicionEEPROM,NumPasosXMetroH);
    PosicionEEPROM += sizeof(int);
    EEPROM.get (PosicionEEPROM,NumPasosXMetroV);
    PosicionEEPROM += sizeof(int);
    EEPROM.get (PosicionEEPROM,nhgMax);
    PosicionEEPROM += sizeof(long);
    EEPROM.get (PosicionEEPROM,nvgMax);
    break;
}

ConversorDistPasosHor=NumPasosXMetroH;      // CALCULA LAS CONSTANTES DE CONVERSIÓN
ConversorDistPasosHor=ConversorDistPasosHor/1000;
ConversorDistPasosVert=NumPasosXMetroV;
ConversorDistPasosVert=ConversorDistPasosVert/1000;
HMaxmm=nhgMax/ConversorDistPasosHor;        // CALCULA LA ZONA DE TRABAJO EN mm
VMaxmm=nvgMax/ConversorDistPasosVert;

Serial.flush();
Serial2.flush();
Serial.print('?');                           // DEVUELVE A LA APLICACION DE
Serial.print('A'); Serial.print (nhgMax); Serial.print('A'); // LabVIEW LOS VALORES INGREADOS
Serial.print('B'); Serial.print (nvgMax); Serial.print('B'); // DE LAS VARIABLES DE AJUSTE mm Y
Serial.print('C'); Serial.print (NumPasosXMetroH); Serial.print('C'); // LA ZONA DE TRABAJO EN mm
Serial.print('D'); Serial.print (NumPasosXMetroV); Serial.print('D');
Serial.print('G'); Serial.print (HMaxmm); Serial.print('G');
Serial.print('H'); Serial.print (VMaxmm); Serial.print('H');
delay (500);

digitalWrite (LuzSalir, LOW);    delay (150); // SE ENCIENDEN EN SECUENCIA LAS
```




```
digitalWrite (LuzDeterminarCero, HIGH); delay (150);           // LUCES DE LAS FUNCIONES INDICANDO
digitalWrite (LuzMovContinuo, HIGH);  delay (150);             // QUE LAS VARIABLES DE AJUSTE
digitalWrite (LuzMovEscalonado, HIGH); delay (150);           // FUERON CARGADAS A LA
digitalWrite (LuzIrAMemoria1, HIGH);  delay (150);             // PROGRAMACIÓN
digitalWrite (LuzIrAMemoria2, HIGH);  delay (150);
digitalWrite (LuzIrACoordenada, HIGH); delay (150);
digitalWrite (LuzMedAutomatica, HIGH); delay (300);
digitalWrite (LuzDeterminarCero, LOW);
digitalWrite (LuzMovContinuo, LOW);
digitalWrite (LuzMovEscalonado, LOW);
digitalWrite (LuzIrAMemoria1, LOW);
digitalWrite (LuzIrAMemoria2, LOW);
digitalWrite (LuzIrACoordenada, LOW);
digitalWrite (LuzMedAutomatica, LOW);
digitalWrite (LuzSalir, HIGH);

for (int i=0; i<65; i++){                                     // LEE EL PUERTO SERIE 2 PARA VACIAR SU BUFFER
    Serial2.read();                                           // DE ENTRADA
}
delay (100);

}                                                             // FIN DEL void setup ()
```

```
void loop() {
```

```
    ImprimirPosicion();

    while (CondSiempreVerdadera == 1){                       // BUCLE while() DE ESPERA DE FUNCION.
        if (digitalRead(DeterminarCero) == HIGH && CeroRealizado == 0){ // LA CONDICION ES SIEMPRE
            Funcion = 1;                                         // VERDADERA. SE SALE DEL MISMO
            break;                                               // MEDIANTE LOS break.
        }                                                       //
        if (digitalRead(MovContinuo) == HIGH){                 // CUANDO SE RECIBE LA ORDEN,
            Funcion = 2;                                         // MEDIANTE PUERTO SERIE O
            break;                                               // BOTONERA, DE EJECUTAR UNA
        }                                                       // FUNCION SE ASIGNA A LA VARIABLE
        if (digitalRead(MovEscalonado) == HIGH){               // Funcion EL VALOR CORRESPONDIENTE
            Funcion = 3;                                         // A LA FUNCION DESEADA
            break;
        }
        if (digitalRead(IrAMemoria1) == HIGH){
            Funcion = 4;
            break;
        }
        if (digitalRead(IrAMemoria2) == HIGH){
            Funcion = 5;
            break;
        }
    }
}
```



```
}
if (Serial.available()>0){
    Funcion = Serial.read()-48;
    break;
}
if (Serial2.available()>0){
    Funcion = Serial2.read()-48;
    break;
}
}

if (Funcion == 2 || Funcion == 3){
    if (Funcion == 2){
        Serial.print("@F2F");           // SE INDICA A LA INTERFACES QUE LA FUNCION
        Serial2.print('2');             //
    }
    else{
        Serial.print("!@F3F");
        Serial2.print('3');
    }
}

A = 7;                                // VALOR INICIAL PARA LA VARIABLE A
delay (200);

if (CeroRealizado == 1){
    if (nvg == nvGMax){                 // SE DETERMINA SI EL POSICIONADOR SE ENCUENTRA EN EL
        dvpML = 1;                     // LIMITE VERTICAL SUPERIOR
        digitalWrite (LuzDVP, HIGH);
    }
    if (nvg == 0){                     // SE DETERMINA SI EL POSICIONADOR SE ENCUENTRA EN EL
        dvnML = 1;                     // LIMITE VERTICAL INFERIOR
        digitalWrite (LuzDVN, HIGH);
    }
    if (nhg == nhGMax){                 // SE DETERMINA SI EL POSICIONADOR SE ENCUENTRA
        dhpML = 1;                     // EN EL LIMITE HORIZONTAL SUPERIOR
        digitalWrite (LuzDHP, HIGH);
    }
    if (nhg == 0){                     // SE DETERMINA SI EL POSICIONADOR SE ENCUENTRA
        dhnmL = 1;                     // EN EL LIMITE HORIZONTAL INFERIOR
        digitalWrite (LuzDHN, HIGH);
    }
    Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);           // SE INDICA A LA INTERFAZ SI EL
    Serial.print (dhpML);Serial.print (dhnmL);Serial.print ('L');           // POSICIONADOR SE ENCUENTRA EN
}                                     // ALGUNO DE LOS LIMITES
}
```



/// ----- FUNCION 1: DETERMINAR CERO -----

```
if (Funcion == 1){
    digitalWrite (LuzDeterminarCero, HIGH);
    digitalWrite (LuzSalir, LOW);
    Serial.print ("@F1F");
    Serial2.print ('1');

    V1 = 0;
    H1 = 0;
    PasosAcel = PasosAcelMax;

    while (CondSiempreVerdadera == 1){

        if (digitalRead(SensorPosH) == HIGH{

            digitalWrite (SalDirPosHor, LOW);

            for (int i=1; i<=PasosAcel; i++){

                digitalWrite (SalPasoHor, HIGH);
                delayMicroseconds (DelayAcel [i]);
                digitalWrite (SalPasoHor, LOW);
                delayMicroseconds (DelayAcel [i]);
                nhg --;
            }

            if (digitalRead(SensorPosH) == HIGH{

                while (digitalRead(SensorPosH) == HIGH{

                    digitalWrite (SalPasoHor, HIGH);
                    delayMicroseconds (DelayAcel [PasosAcel]);
                    digitalWrite (SalPasoHor, LOW);
                    delayMicroseconds (DelayAcel [PasosAcel]);
                    nhg --;
                }

                for (int i=PasosAcel-1; i>=0; i--){

                    digitalWrite (SalPasoHor, HIGH);
                    delayMicroseconds (DelayAcel [i]);
                    digitalWrite (SalPasoHor, LOW);
                    delayMicroseconds (DelayAcel [i]);
                    nhg --;
                }
                break;
            }
            else{

                for (int i=PasosAcel-1; i>=0; i--){
```



```
        digitalWrite (SalPasoHor, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [i]);
        nhg --;
    }
    delay (200);
}

digitalWrite (SalDirPosHor, HIGH);
nAux = 0;

for (int i=1; i<=PasosAcel; i++){                                // RAMPA DE ACELERACION POSITIVA
    digitalWrite (SalPasoHor, HIGH);
    delayMicroseconds (DelayAcel [i]);
    digitalWrite (SalPasoHor, LOW);
    delayMicroseconds (DelayAcel [i]);
    nhg ++;
}

while (digitalRead(SensorPosH) == LOW){                          //AVANCE CONTINUO POSITIVO HASTA
    digitalWrite (SalPasoHor, HIGH);                             // LIBERAR SENSOR
    delayMicroseconds (DelayAcel [PasosAcel]);                  // (== LOW para posicionador)
    digitalWrite (SalPasoHor, LOW);                             // (== HIGH para prototipo)
    delayMicroseconds (DelayAcel [PasosAcel]);
    nhg ++;
    nAux ++;
    if (nAux > NumPasosBloqSensor){                              // SENSOR DE POSICION BLOQUEADO
        H1 = 1;
        break;
    }
}

for (int i=1; i<20; i++){                                        // AVANCE CONTINUO POSITIVO EXTRA
    digitalWrite (SalPasoHor, HIGH);
    delayMicroseconds (DelayAcel [PasosAcel]);
    digitalWrite (SalPasoHor, LOW);
    delayMicroseconds (DelayAcel [PasosAcel]);
    nhg ++;
}

for (int i=PasosAcel-1; i>=0; i--){                              // RAMPA DE DESACELERACION
    digitalWrite (SalPasoHor, HIGH);
    delayMicroseconds (DelayAcel [i]);
    digitalWrite (SalPasoHor, LOW);
    delayMicroseconds (DelayAcel [i]);
    nhg ++;
}
```



```
delay (50);

if (H1 == 1){
    for (int i=1; i<=5; i++){
        digitalWrite (LuzDeterminarCero, LOW);
        delay (250);
        digitalWrite (LuzDeterminarCero, HIGH);
        delay (250);
    }
    break;
}

}

while (CondSiempreVerdadera == 1){

    if (digitalRead(SensorPosV) == HIGH){

        digitalWrite (SalDirPosVert, LOW);

        for (int i=1; i<=PasosAcel; i++){

            digitalWrite (SalPasoVert, HIGH);
            delayMicroseconds (DelayAcel [i]);
            digitalWrite (SalPasoVert, LOW);
            delayMicroseconds (DelayAcel [i]);
            nvgl --;
        }

        if (digitalRead(SensorPosV) == HIGH){

            while (digitalRead(SensorPosV) == HIGH){

                digitalWrite (SalPasoVert, HIGH);
                delayMicroseconds (DelayAcel [PasosAcel]);
                digitalWrite (SalPasoVert, LOW);
                delayMicroseconds (DelayAcel [PasosAcel]);
                nvgl --;
            }

            for (int i=PasosAcel-1; i>=0; i--){

                digitalWrite (SalPasoVert, HIGH);
                delayMicroseconds (DelayAcel [i]);
                digitalWrite (SalPasoVert, LOW);
                delayMicroseconds (DelayAcel [i]);
                nhgl --;
            }
            break;
        }

        else{ // 2DO
```



```
        for (int i=PasosAcel-1; i>=0; i--){
            digitalWrite (SalPasoVert, HIGH);
            delayMicroseconds (DelayAcel [i]);
            digitalWrite (SalPasoVert, LOW);
            delayMicroseconds (DelayAcel [i]);
            nvg --;
        }
        delay (200);
    }

    digitalWrite (SalDirPosVert, HIGH);
    nAux = 0;

    for (int i=1; i<=PasosAcel; i++){
        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [i]);
        nvg ++;
    }

    while (digitalRead(SensorPosV) == LOW){
        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [PasosAcel]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [PasosAcel]);
        nvg ++;
        nAux ++;
        if (nAux > NumPasosBloqSensor){
            V1 = 1;
            break;
        }
    }

    for (int i=1; i<20; i++){
        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [PasosAcel]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [PasosAcel]);
        nvg ++;
    }

    for (int i=PasosAcel-1; i>=0; i--){
        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [i]);
    }
}
```

// RAMPA DE DESACELERACION NEGATIVA

// RAMPA DE ACELERACION POSITIVA

// AVANCE CONTINUO POSITIVO HASTA LIBERAR
// SENSOR // (== LOW para posicionador)
// (== HIGH para prototipo)

// SENSOR DE POSICION BLOQUEADO

// AVANCE CONTINUO POSITIVO EXTRA

// RAMPA DE DESACELERACION



```
        nvg ++;
    }

    delay (50);

    if (V1 == 1){
        for (int i=1; i<=5; i++){
            digitalWrite (LuzDeterminarCero, LOW);
            delay (600);
            digitalWrite (LuzDeterminarCero, HIGH);
            delay (100);
            digitalWrite (LuzDeterminarCero, LOW);
            delay (100);
            digitalWrite (LuzDeterminarCero, HIGH);
            delay (100);
        }
        break;
    }
}

if (H1 == 1 || V1 == 1){
    Serial.println ("XFalla en:");
    if (H1 == 1){
        Serial.println (" - Sensor Horizontal");
    }
    if (V1 == 1){
        Serial.println (" - Sensor Vertical");
    }
    Serial.println (' ');
    Serial.println ("PUNTO DE REFERENCIA NO DETERMINADOX");
    delay (150);
}

if (H1 == 0 && V1 == 0){
    if (Mem1Utilizada == 1){
        nv1 = nv1 - nvg;
        nh1 = nh1 - nhg;
        if (nv1 < 0 || nh1 < 0 || nv1 > nvgMax || nh1 > nhgMax){
            nh1 = nv1 = 0;
        }
    }
    else{
        nv1 = nh1 = 0;
    }
    if (Mem2Utilizada == 1){
        nv2 = nv2 - nvg;
        nh2 = nh2 - nhg;
    }
}
```



```
        if (nv2 < 0 || nh2 < 0 || nv2 > nvMax || nh2 > nhMax){
            nh2 = nv2 = 0;
        }
    }
    else{
        nv2 = nh2 = 0;
    }
    nvMax = 0; // SE ESTABLECE EL PUNTO ACTUAL COMO CERO GLOBAL
    nhMax = 0; // PONIENDO A CERO LOS CONTADORES DE PASOS
    CeroRealizado = 1; // BANDERA QUE INDICA QUE LA FUNCION YA SE EJECUTÓ
}
for (int i=0; i<65; i++){ // SE LEEN LOS PUERTOS SERIE PARA VACIAR SUS BUFFERS
    Serial.read(); // DE ENTRADA
    Serial2.read();
}

digitalWrite (LuzDeterminarCero, LOW); // SI INDICA EN LA BOTONERA QUE LA FUNCION FINALIZÓ
}
```

/// ----- FUNCION 2: MOVIMIENTO LIBRE CONTINUO (CON PCCP) -----

```
while (Funcion == 2 && CeroRealizado == 1){ // SE INGRESA SOLO SI FUE DETERMINADO EL
                                            // CERO GLOBAL

    digitalWrite (LuzMovContinuo, HIGH);
    digitalWrite (LuzSalir, LOW);

    if (Serial.available()>0){ // RECIBE DATOS DESDE LA APLICACION DE
        M = Serial.read()-48; // LABVIEW. SI M == 'P'-48 ES POR QUE
        if (M == 'P'-48){ // LA APLICACION SOLICITO LA POSICION
            }
        else{
            A = M;
        }
    }
    if (Serial2.available()>0){ // RECIBE DATOS DESDE LA APP
        A = Serial2.read()-48;
    }

    if ((A == 8 || digitalRead(EntAvanceVert) == HIGH) && nvMax < nvMax){

        dvnML = 0;
        digitalWrite (LuzDVN, LOW);
        Serial.print ("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
        Serial.print ('L');
        If (nvMax - nvMax >= 2*PasosAcelMax){ // NUMERO DE PASOS DE LAS RAMPAS
            PasosAcel = PasosAcelMax;
        }
    }
}
```




```
}  
else{  
    PasosAcel = ((nvgMax - nvg)-1)/2;  
}  
  
digitalWrite (SalDirPosVert, HIGH);           // RAMPA DE ACELERACION  
for (int i=1; i<=PasosAcel; i++){  
    digitalWrite (SalPasoVert, HIGH);  
    delayMicroseconds (DelayAcel [i]);  
    digitalWrite (SalPasoVert, LOW);  
    delayMicroseconds (DelayAcel [i]);  
    nvg ++;  
}  
  
while (A == 8 || digitalRead(EntAvanceVert) == HIGH){    //AVANCE CONTINUO  
  
    if (nvg + PasosAcel >= nvgMax){  
        break;  
    }  
    digitalWrite (SalPasoVert, HIGH);  
    delayMicroseconds (DelayAcel [PasosAcel]);  
    digitalWrite (SalPasoVert, LOW);  
    delayMicroseconds (DelayAcel [PasosAcel]);  
    nvg ++;  
    if (Serial.available()>0){  
        M = Serial.read()-48;  
        if (M == 'P'-48){  
            ImprimirPosicion();  
        }  
        else{  
            A = M;  
        }  
    }  
    if (Serial2.available()>0){  
        A = Serial2.read()-48;  
    }  
}  
  
for (int i=PasosAcel-1; i>=0; i--){           // RAMPA DE DESACELERACION  
    digitalWrite (SalPasoVert, HIGH);  
    delayMicroseconds (DelayAcel [i]);  
    digitalWrite (SalPasoVert, LOW);  
    delayMicroseconds (DelayAcel [i]);  
    nvg ++;  
}  
ImprimirPosicion();  
for (int i=0; i<65; i++){
```



```
        Serial.read();
        Serial2.read();
    }

    if (nvg >= nvgMax){                                     // SE INDICA A LA INTERFACES QUE HAY UN
        dvpML = 1;                                         // DESBORDE VERTICAL POSITIVO
        Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
        Serial.print ('L');
        digitalWrite (LuzDVP, HIGH);
    }
}

if ((A == 5 || digitalRead(EntRetrocesoVert) == HIGH) && nvg > 0){

    dvpML = 0;
    digitalWrite (LuzDVP, LOW);
    Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');

    if (nvg >= 2*PasosAcelMax){                           // NUMERO DE PASOS DE LAS RAMPAS
        PasosAcel = PasosAcelMax;
    }
    else{
        PasosAcel = (nvg - 1)/2;
    }

    digitalWrite (SalDirPosVert, LOW);                    // RAMPA DE ACELERACION
    for (int i=1; i<=PasosAcel; i++){
        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [i]);
        nvg --;
    }

    while (A == 5 || digitalRead(EntRetrocesoVert) == HIGH){ //AVANCE CONTINUO

        if (nvg <= PasosAcel){
            break;
        }
        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [PasosAcel]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [PasosAcel]);
        nvg--;

        if (Serial.available()>0){
```



```
        M = Serial.read()-48;
        if (M == 'P'-48){
            ImprimirPosicion();
        }
        else{
            A = M;
        }
    }

    if (Serial2.available()>0){
        A = Serial2.read()-48;
    }
}

for (int i=PasosAcel-1; i>=0; i--){
    digitalWrite (SalPasoVert, HIGH);
    delayMicroseconds (DelayAcel [i]);
    digitalWrite (SalPasoVert, LOW);
    delayMicroseconds (DelayAcel [i]);
    nvg --;
}
ImprimirPosicion();
for (int i=0; i<65; i++){
    Serial.read();
    Serial2.read();
}

if (nvg <= 0){
    dvnML = 1;
    Serial.print ("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');
    digitalWrite (LuzDVN, HIGH);
}
}

if ((A == 6 || digitalRead(EntAvanceHor) == HIGH) && nhg < nhgMax){

    dhnML = 0;
    digitalWrite (LuzDHN, LOW);
    Serial.print ("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');

    if (nhgMax - nhg >= 2*PasosAcelMax){
        PasosAcel = PasosAcelMax;
    }
    else{
        PasosAcel = ((nhgMax - nhg)-1)/2;
    }
}
```



```
}

digitalWrite (SalDirPosHor, HIGH);                // RAMPA DE ACELERACION
for (int i=1; i<=PasosAcel; i++){
    digitalWrite (SalPasoHor, HIGH);
    delayMicroseconds (DelayAcel [i]);
    digitalWrite (SalPasoHor, LOW);
    delayMicroseconds (DelayAcel [i]);
    nhg ++;
}

while (A == 6 || digitalRead(EntAvanceHor) == HIGH){    //AVANCE CONTINUO

    if (nhg + PasosAcel >= nhgMax){
        break;
    }
    digitalWrite (SalPasoHor, HIGH);
    delayMicroseconds (DelayAcel [PasosAcel]);
    digitalWrite (SalPasoHor, LOW);
    delayMicroseconds (DelayAcel [PasosAcel]);
    nhg ++;

    if (Serial.available()>0){
        M = Serial.read()-48;
        if (M == 'P'-48){
            ImprimirPosicion();
        }
        else{
            A = M;
        }
    }

    if (Serial2.available()>0){
        A = Serial2.read()-48;
    }
}

for (int i=PasosAcel-1; i>=0; i--){                // RAMPA DE DESACELERACION
    digitalWrite (SalPasoHor, HIGH);
    delayMicroseconds (DelayAcel [i]);
    digitalWrite (SalPasoHor, LOW);
    delayMicroseconds (DelayAcel [i]);
    nhg ++;
}
ImprimirPosicion();
for (int i=0; i<65; i++){
    Serial.read();
}
```



```
        Serial2.read();
    }

    if (nhg >= nhgMax){                                     // SE INDICA A LA INTERFACES QUE HAY UN
        dhpML = 1;                                         // DESBORDE HORIZONTAL POSITIVO
        Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
        Serial.print ('L');
        digitalWrite (LuzDHP, HIGH);
    }
}

if ((A == 4 || digitalRead(EntRetrocesoHor) == HIGH) && nhg > 0){

    dhpML = 0;
    digitalWrite (LuzDHP, LOW);
    Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');

    if (nhg >= 2*PasosAcelMax){    // NUMERO DE PASOS DE LAS RAMPAS
        PasosAcel = PasosAcelMax;
    }
    else{
        PasosAcel = (nhg - 1)/2;
    }

    digitalWrite (SalDirPosHor, LOW);                    // RAMPA DE ACELERACION
    for (int i=1; i<=PasosAcel; i++){
        digitalWrite (SalPasoHor, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [i]);
        nhg --;
    }

    while (A == 4 || digitalRead(EntRetrocesoHor) == HIGH){    //AVANCE CONTINUO

        if (nhg <= PasosAcel){
            break;
        }
        digitalWrite (SalPasoHor, HIGH);
        delayMicroseconds (DelayAcel [PasosAcel]);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [PasosAcel]);
        nhg --;

        if (Serial.available()>0){
            M = Serial.read()-48;
```



```
        if (M == 'P'-48){
            ImprimirPosicion();
        }
        else{
            A = M;
        }
    }

    if (Serial2.available()>0){
        A = Serial2.read()-48;
    }
}

for (int i=PasosAcel-1; i>=0; i--){          // RAMPA DE DESACELERACION
    digitalWrite (SalPasoHor, HIGH);
    delayMicroseconds (DelayAcel [i]);
    digitalWrite (SalPasoHor, LOW);
    delayMicroseconds (DelayAcel [i]);
    nhg --;
}
ImprimirPosicion();
for (int i=0; i<65; i++){
    Serial.read();
    Serial2.read();
}

if (nhg <= 0){                                // SE INDICA A LA INTERFACES QUE HAY UN
    dhnML = 1;                                // DESBORDE HORIZONTAL NEGATIVO
    Serial.print ("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');
    digitalWrite (LuzDHN, HIGH);
}
}

if (digitalRead(Memoria1) == HIGH || A == 1){
    nv1 = nvg;                                // ALMACENA LA POSICION EN LAS MEMORIAS 1
    nh1 = nhg;
    Mem1Utilizada = 1;
    digitalWrite(LuzGuardarEnMemoria1,HIGH);
    Serial.print ("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ("10L");                      // SE INDICA A LA INTERFACES QUE LA POSICION
    Serial2.print ('A');                        // SE GUARDÓ EN LA MEM 1
    delay (500);
    digitalWrite(LuzGuardarEnMemoria1,LOW);
    A = 7;
    for (int i=0; i<65; i++){
        Serial.read();
    }
}
```



```
        Serial2.read();
    }
    while (digitalRead(Memoria1) == HIGH){           // ESPER A QUE SE SUELTE EL PULSADOR SI SE
    }                                                  // ESTA TRABAJANDO CON LA BOTONERA
}

if (digitalRead(Memoria2) == HIGH || A == 2){
    nv2 = nvg;                                       // ALMACENA LA POSICION EN LAS MEMORIAS 2
    nh2 = nhg;
    Mem2Utilizada = 1;
    digitalWrite(LuzGuardarEnMemoria2,HIGH);
    Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ("01L");                           // SE INDICA A LA INTERFACES QUE LA POSICION
    Serial2.print ('B');                             // SE GUARDÓ EN LA MEM 2
    delay (500);
    digitalWrite(LuzGuardarEnMemoria2,LOW);
    A = 7;
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
    while (digitalRead(Memoria2) == HIGH){           // ESPER A QUE SE SUELTE EL PULSADOR SI SE ESTA
    }                                                  // TRABAJANDO CON LA BOTONERA
}

if (digitalRead(Salir) == HIGH || A == 0){
    Funcion = 0;                                     // CONDICION DE SALIDA
    dvpML = dvnML = dhpML = dhnML = 0;
    Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');
    digitalWrite (LuzDVP, LOW);
    digitalWrite (LuzDVN, LOW);
    digitalWrite (LuzDHP, LOW);
    digitalWrite (LuzDHN, LOW);
}
}
digitalWrite (LuzMovContinuo, LOW);
```

/// ----- FUNCION 20: MOVIMIENTO LIBRE CONTINUO (SIN PCCP) -----

```
while (Funcion == 2 && CeroRealizado == 0){           // SE INGRESA SOLO SI NO FUE DETERMINADO EL
                                                        // CERO GLOBAL

    digitalWrite (LuzMovContinuo, HIGH);
    digitalWrite (LuzSalir, LOW);
    PasosAcel = PasosAcelMax;
```



```
if (Serial.available()>0){                                     // PUERTO SERIE ASIGNA A LA VARIABLE "A" EL
    M = Serial.read()-48;                                     // VALOR CORRESPONDIENTE AL DESPLAZAMIENTO
    if (M == 'P'-48){                                        // DESEADO
    }
    else{
        A = M;
    }
}
if (Serial2.available()>0){
    A = Serial2.read()-48;
}

if (A == 8 || digitalRead(EntAvanceVert) == HIGH){           // DESPLAZAMIENTO VERTICAL POSITIVO

    digitalWrite (SalDirPosVert, HIGH);

    for (int i=1; i<=PasosAcel; i++){                       // RAMPA DE ACELERACION
        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [i]);
        nvg ++;
    }

    while (A == 8 || digitalRead(EntAvanceVert) == HIGH){    // AVANCE CONTINUO

        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [PasosAcel]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [PasosAcel]);
        nvg ++;

        if (Serial.available()>0){
            M = Serial.read()-48;
            if (M == 'P'-48){
                ImprimirPosicion();
            }
            else{
                A = M;
            }
        }

        if (Serial2.available()>0){
            A = Serial2.read()-48;
        }
    }

    for (int i=PasosAcel-1; i>=0; i--){                       // RAMPA DE DESACELERACION
        digitalWrite (SalPasoVert, HIGH);
```




```
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [i]);
        nvg ++;
    }
    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
}

if (A == 5 || digitalRead(EntRetrocesoVert) == HIGH){           // DESPLAZAMIENTO VERTICAL NEGATIVO

    digitalWrite (SalDirPosVert, LOW);

    for (int i=1; i<=PasosAcel; i++){                           // RAMPA DE ACELERACION
        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [i]);
        nvg --;
    }

    while (A == 5 || digitalRead(EntRetrocesoVert) == HIGH){   // AVANCE CONTINUO

        digitalWrite (SalPasoVert, HIGH);
        delayMicroseconds (DelayAcel [PasosAcel]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [PasosAcel]);
        nvg --;

        if (Serial.available()>0){
            M = Serial.read()-48;
            if (M == 'P'-48){
                ImprimirPosicion();
            }
            else{
                A = M;
            }
        }

        if (Serial2.available()>0){
            A = Serial2.read()-48;
        }
    }

    for (int i=PasosAcel-1; i>=0; i--){                          // RAMPA DE DESACELERACION
        digitalWrite (SalPasoVert, HIGH);
```



```
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoVert, LOW);
        delayMicroseconds (DelayAcel [i]);
        nhg --;
    }
    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
}

if (A == 6 || digitalRead(EntAvanceHor) == HIGH){           // DESPLAZAMIENTO HORIZONTAL POSITIVO

    digitalWrite (SalDirPosHor, HIGH);

    for (int i=1; i<=PasosAcel; i++){                       // RAMPA DE ACELERACION
        digitalWrite (SalPasoHor, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [i]);
        nhg ++;
    }

    while (A == 6 || digitalRead(EntAvanceHor) == HIGH){    // AVANCE CONTINUO

        digitalWrite (SalPasoHor, HIGH);
        delayMicroseconds (DelayAcel [PasosAcel]);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [PasosAcel]);
        nhg ++;

        if (Serial.available()>0){
            M = Serial.read()-48;
            if (M == 'P'-48){
                ImprimirPosicion();
            }
            else{
                A = M;
            }
        }

        if (Serial2.available()>0){
            A = Serial2.read()-48;
        }
    }

    for (int i=PasosAcel-1; i>=0; i--){                      // RAMPA DE DESACELERACION
        digitalWrite (SalPasoHor, HIGH);
```



```
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [i]);
        nhg ++;
    }
    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
}

if (A == 4 || digitalRead(EntRetrocesoHor) == HIGH){           // DESPLAZAMIENTO HORIZONTAL NEGATIVO

    digitalWrite (SalDirPosHor, LOW);

    for (int i=1; i<=PasosAcel; i++){                          // RAMPA DE ACELERACION
        digitalWrite (SalPasoHor, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [i]);
        nhg --;
    }

    while (A == 4 || digitalRead(EntRetrocesoHor) == HIGH){    // AVANCE CONTINUO

        digitalWrite (SalPasoHor, HIGH);
        delayMicroseconds (DelayAcel [PasosAcel]);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [PasosAcel]);
        nhg --;

        if (Serial.available()>0){
            M = Serial.read()-48;
            if (M == 'P'-48){
                ImprimirPosicion();
            }
            else{
                A = M;
            }
        }

        if (Serial2.available()>0){
            A = Serial2.read()-48;
        }
    }

    for (int i=PasosAcel-1; i>=0; i--){                          // RAMPA DE DESACELERACION
        digitalWrite (SalPasoHor, HIGH);
```



```
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [i]);
        nhg --;
    }
    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
}

if (A == 1 || digitalRead(Memoria1) == HIGH){                                // GUARDAR POSICION EN MEMORIA 1

    nv1 = nvg;
    nh1 = nhg;
    Mem1Utilizada = 1;
    digitalWrite(LuzGuardarEnMemoria1,HIGH);
    Serial.print ("@L000010L");
    Serial2.print ('A');
    delay (500);
    digitalWrite(LuzGuardarEnMemoria1,LOW);
    A = 7;
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
    while (digitalRead(Memoria1) == HIGH){                                // ESPER A QUE SE SUELTE EL PULSADOR SI SE ESTA
    }                                                                    // TRABAJANDO CON LA BOTONERA
}

if (A == 2 || digitalRead(Memoria2) == HIGH){                                // GUARDAR POSICION EN MEMORIA 2

    nv2 = nvg;
    nh2 = nhg;
    Mem2Utilizada = 1;
    digitalWrite(LuzGuardarEnMemoria2,HIGH);
    Serial.print ("@L000001L");
    Serial2.print ('B');
    delay (500);
    digitalWrite(LuzGuardarEnMemoria2,LOW);
    A = 7;
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
    while (digitalRead(Memoria2) == HIGH){                                // ESPER A QUE SE SUELTE EL PULSADOR SI SE ESTA
```



```
    }
}

if (A == 0 || digitalRead(Salir) == HIGH){
    Funcion = 0;
}
}

digitalWrite (LuzMovContinuo, LOW);
```

/// ----- FUNCION 3: MOVIMIENTO LIBRE ESCALONADO (CON PCCP) -----

```
while (Funcion == 3 && CeroRealizado == 1){
    digitalWrite (LuzMovEscalonado, HIGH);
    digitalWrite (LuzSalir, LOW);

    A = 7;
    if (Serial.available()>0){
        A = Serial.read()-48;
    }
    if (Serial2.available()>0){
        A = Serial2.read()-48;
    }

    if ((A == 8 || digitalRead(EntAvanceVert) == HIGH) && nvg < nvgMax){

        dvnML = 0;
        digitalWrite (LuzDVN, LOW);
        Serial.print ("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
        Serial.print ('L');

        if (nvgMax - nvg >= NumPasosVert ){
            nvFinal = nvg + NumPasosVert;
            nhFinal = nhg;
            IrACoordenada ();
        }
        else{
            nvFinal = nvgMax;
            nhFinal = nhg;
            IrACoordenada ();
        }
        ImprimirPosicion();
        for (int i=0; i<65; i++){
            Serial.read();
            Serial2.read();
        }
    }
}
```



```
if (nvg >= nvgMax){                                     // SE INDICA A LA INTERFACES QUE HAY UN
    dvpML = 1;                                           // DESBORDE VERTICAL POSITIVO
    Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');
    digitalWrite (LuzDVP, HIGH);
}
while (digitalRead(EntAvanceVert) == HIGH){
}
}

if ((A == 5 || digitalRead(EntRetrocesoVert) == HIGH) && nvg > 0){

    dvpML = 0;
    digitalWrite (LuzDVP, LOW);
    Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');

    if (nvg >= NumPasosVert){                             // SE DETERMINA LAS COORDENADAS DEL PUNTO
        nvFinal = nvg - NumPasosVert;                     // FINAL DEL DESPLAZAMIENTO
        nhFinal = nhg;
        IrACoordenada ();
    }
    else{
        nvFinal = 0;
        nhFinal = nhg;
        IrACoordenada ();
    }

    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }

    if (nvg <= 0){                                         // SE INDICA A LA INTERFACES QUE HAY UN
        dvnML = 1;                                         // DESBORDE VERTICAL NEGATIVO
        Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
        Serial.print ('L');
        digitalWrite (LuzDVN, HIGH);
    }
    while (digitalRead(EntRetrocesoVert) == HIGH){
    }
}

if ((A == 6 || digitalRead(EntAvanceHor) == HIGH) && nhg < nhgMax){

    dhnML = 0;
```



```
digitalWrite (LuzDHN, LOW);
Serial.print ("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
Serial.print ('L');

if (nhgMax - nhg >= NumPasosHor){                                     // SE DETERMINA LAS COORDENADAS DEL PUNTO
    nvFinal = nvG;                                                  // FINAL DEL DESPLAZAMIENTO
    nhFinal = nhg + NumPasosHor;
    IrACoordenada ();
}
else{
    nvFinal = nvG;
    nhFinal = nhgMax;
    IrACoordenada ();
}

ImprimirPosicion();
for (int i=0; i<65; i++){
    Serial.read();
    Serial2.read();
}

if (nhg >= nhgMax){                                                  // SE INDICA A LA INTERFACES QUE HAY UN
    dhpML = 1;                                                       // DESBORDE HORIZONTAL POSITIVO
    Serial.print ("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');
    digitalWrite (LuzDHP, HIGH);
}
while (digitalRead(EntAvanceHor) == HIGH){
}
}

if ((A == 4 || digitalRead(EntRetrocesoHor) == HIGH) && nhg > 0){

    dhpML = 0;
    digitalWrite (LuzDHP, LOW);
    Serial.print ("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ('L');

    if (nhg >= NumPasosHor){                                          // SE DETERMINA LAS COORDENADAS DEL PUNTO
        nvFinal = nvG;                                              // FINAL DEL DESPLAZAMIENTO
        nhFinal = nhg - NumPasosHor;
        IrACoordenada ();
    }
    else{
        nvFinal = nvG;
        nhFinal = 0;
        IrACoordenada ();
    }
}
```



```
    }

    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }

    if (nhg <= 0){                                     // SE INDICA A LA INTERFACES QUE HAY UN
        dhnML = 1;                                     // DESBORDE HORIZONTAL POSITIVO
        Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
        Serial.print ('L');
        digitalWrite (LuzDHN, HIGH);
    }
    while (digitalRead(EntRetrocesoHor) == HIGH){
    }
}

if (digitalRead(Memoria1) == HIGH || A == 1){          // ALMACENA LA POSICION EN LAS MEMORIAS 1
    nv1 = nvg;
    nh1 = nhg;
    Mem1Utilizada = 1;
    digitalWrite(LuzGuardarEnMemoria1,HIGH);
    Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ("10L");                             // SE INDICA A LA INTERFACES QUE LA
    Serial2.print ('A');                               // POSICION SE GUARDÓ EN LA MEM 1
    delay (500);
    digitalWrite(LuzGuardarEnMemoria1,LOW);
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
}

if (digitalRead(Memoria2) == HIGH || A == 2){          // ALMACENA LA POSICION EN LAS MEMORIAS 2
    nv2 = nvg;
    nh2 = nhg;
    Mem2Utilizada = 1;
    digitalWrite(LuzGuardarEnMemoria2,HIGH);
    Serial.print("@L");Serial.print (dvpML);Serial.print (dvnML);Serial.print (dhpML);Serial.print (dhnML);
    Serial.print ("01L");                             // SE INDICA A LA INTERFACES QUE LA
    Serial2.print ('B');                               // POSICION SE GUARDÓ EN LA MEM 2
    delay (500);
    digitalWrite(LuzGuardarEnMemoria2,LOW);
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
}
```




```
    }  
}  
  
if (digitalRead(Salir) == HIGH || A == 0){  
    Funcion = 0;  
    dvpML = dvnML = dhpML = dhnmML = 0;  
    Serial.print("@L");Serial.print(dvpML);Serial.print(dvnML);Serial.print(dhpML);Serial.print(dhnmML);  
    Serial.print('L');  
    digitalWrite(LuzDVP, LOW);  
    digitalWrite(LuzDVN, LOW);  
    digitalWrite(LuzDHP, LOW);  
    digitalWrite(LuzDHN, LOW);  
}  
}  
digitalWrite(LuzMovEscalonado, LOW);
```

```
/// ----- FUNCION 30: MOVIMIENTO LIBRE ESCALONADO (SIN PCCP) -----
```

```
while (Funcion == 3 && CeroRealizado == 0){  
  
    digitalWrite(LuzMovEscalonado, HIGH);  
    digitalWrite(LuzSalir, LOW);  
  
    A = 7;  
    if (Serial.available() > 0){  
        A = Serial.read() - 48;  
    }  
    if (Serial2.available() > 0){  
        A = Serial2.read() - 48;  
    }  
  
    if (A == 8 || digitalRead(EntAvanceVert) == HIGH){  
  
        nvFinal = nvG + NumPasosVert;  
        nhFinal = nhG;  
        IrACoordenada();  
        ImprimirPosicion();  
        for (int i = 0; i < 65; i++){  
            Serial.read();  
            Serial2.read();  
        }  
        while (digitalRead(EntAvanceVert) == HIGH){  
        }  
    }  
}
```



```
if (A == 5 || digitalRead(EntRetrocesoVert) == HIGH){

    nvFinal = nvg - NumPasosVert;                // SE DETERMINA LAS COORDENADAS DEL
    nhFinal = nhg;                                // PUNTO FINAL DEL DESPLAZAMIENTO
    IrACoordenada ();
    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
    while (digitalRead(EntRetrocesoVert) == HIGH){
    }
}

if (A == 6 || digitalRead(EntAvanceHor) == HIGH){

    nvFinal = nvg;                                // SE DETERMINA LAS COORDENADAS DEL
    nhFinal = nhg + NumPasosHor;                  // PUNTO FINAL DEL DESPLAZAMIENTO
    IrACoordenada ();
    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
    while (digitalRead(EntAvanceHor) == HIGH){
    }
}

if (A == 4 || digitalRead(EntRetrocesoHor) == HIGH){

    nvFinal = nvg;                                // SE DETERMINA LAS COORDENADAS DEL
    nhFinal = nhg - NumPasosHor;                  // PUNTO FINAL DEL DESPLAZAMIENTO
    IrACoordenada ();
    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
    while (digitalRead(EntRetrocesoHor) == HIGH){
    }
}

if (A == 1 || digitalRead(Memoria1) == HIGH){
    nv1 = nvg;
    nh1 = nhg;
    Mem1Utilizada = 1;
    digitalWrite(LuzGuardarEnMemoria1,HIGH);
}
```



```
Serial.print("@L000010L");
Serial2.print('A');
delay(500);
digitalWrite(LuzGuardarEnMemoria1,LOW);
for (int i=0; i<65; i++){
    Serial.read();
    Serial2.read();
}
while (digitalRead(Memoria1) == HIGH){
}
}

if (A == 2 || digitalRead(Memoria2) == HIGH){
    nv2 = nv;
    nh2 = nh;
    Mem2Utilizada = 1;
    digitalWrite(LuzGuardarEnMemoria2,HIGH);
    Serial.print("@L000001L");
    Serial2.print('B');
    delay(500);
    digitalWrite(LuzGuardarEnMemoria2,LOW);
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
    while (digitalRead(Memoria2) == HIGH){
    }
}

if (A == 0 || digitalRead(Salir) == HIGH){
    Funcion = 0;
}
}

digitalWrite (LuzMovEscalonado, LOW);
```

```
/// ----- FUNCION 4: IR A MEMORIA 1 -----
```

```
if (Funcion == 4){

    digitalWrite (LuzIrAMemoria1, HIGH);
    digitalWrite (LuzSalir, LOW);
    Serial.print("@F4F");
    Serial2.print('4');
    nvFinal = nv1;
    nhFinal = nh1;
    IrACoordenada ();
}
```

// SE INDICA A LA INTERFACES QUE LA FUNCION
// ESTA EN EJECUCION
// SE DETERMINA LAS COORDENADAS DEL PUNTO
// FINAL DEL DESPLAZAMIENTO



```
ImprimirPosicion();
for (int i=0; i<65; i++){
    Serial.read();
    Serial2.read();
}
Funcion = 0; // CONDICION DE SALIDA
digitalWrite (LuzIrAMemoria1, LOW);
}
```

/// ----- FUNCION 5: IR A MEMORIA 2 -----

```
if (Funcion == 5){
    digitalWrite (LuzIrAMemoria2, HIGH);
    digitalWrite (LuzSalir, LOW);
    Serial.print ("@F5F");
    Serial2.print ('4');
    nvFinal = nv2; // SE DETERMINA LAS COORDENADAS DEL PUNTO
    nhFinal = nh2; // FINAL DEL DESPLAZAMIENTO
    IrACoordenada ();
    ImprimirPosicion();
    for (int i=0; i<65; i++){
        Serial.read();
        Serial2.read();
    }
    Funcion = 0; // CONDICION DE SALIDA
    digitalWrite (LuzIrAMemoria2, LOW);
}
```

/// ----- FUNCION 6: MEDICION AUTOMATICA -----

```
if (Funcion == 6){

    digitalWrite (LuzMedAutomatica, HIGH);
    digitalWrite (LuzSalir, LOW);
    Serial.print ("@F6F"); // SE INDICA A LA INTERFACES QUE LA FUNCION
    Serial2.print ('4'); // ESTA EN EJECUCION

    nvr = nv2; // GUARDA LA POSICION DEL CERO RELATIVO
    nhr = nh2;

    int X = 0;
    int Y = 0;
    boolean D = 0; // SE DECLARAN BANDERAS COMO VARIABLES
    boolean E; // LOCALES
    boolean F;
```



```
boolean G;  
boolean MEDAUTOM = 0;  
boolean dhp;  
boolean dhn;  
boolean dvp;  
boolean dvn;
```

```
delay (250);
```

```
/// ----- [6.1] INGRESO DE DATOS Y DETECCION DE DESBORDES -----
```

```
/// INGRESO DE DATOS ///
```

```
while(D == 0){
```

```
    Serial.print("@M1M");
```

```
// SE INDICA A LA INTERFACES QUE SE ESTA  
// ESPERANDO LOS DATOS PARA EL PROCESO  
// DE MEDICION
```

```
    while(Serial.available() < 29){  
        delay(200);  
    }
```

```
// SE PERMANECE EN EL BUCLE HASTA QUE  
// LLEGUEN LOS DATOS DESDE LA APLICACION  
// DE LABVIEW
```

```
    memset(DatosMedAut, 0, sizeof(DatosMedAut));  
    posicion = 0;  
    while (Serial.available()>0){  
        delay (5);  
        DatosMedAut[posicion] = Serial.read();  
        posicion++;  
    }
```

```
// SE ALMACENAN LOS 29 CARACTERES EN EL  
// STRING DatosMedAut[]
```

```
    memset(datos1, 0, sizeof(datos1));  
    posicion=0;  
    for (int i=0; i<3; i++){  
        datos1[posicion] = DatosMedAut[i];  
        posicion++;  
    }  
    DistHor=atol(datos1);
```

```
// MEDIANTE EL STRING datos1[]SE DESCOMPONE  
// EL STRING DatosMedAut[] PARA SEPARAR LOS  
// DATOS CONTENIDOS EN EL Y ALMACENARLOS  
// EN SUS VARIABLES CORRESPONDIENTES
```

```
    memset(datos1, 0, sizeof(datos1));  
    posicion=0;  
    for (int i=3; i<7; i++){  
        datos1[posicion] = DatosMedAut[i];  
        posicion++;  
    }  
    mhp=atol(datos1);
```



```
memset(datos1, 0, sizeof(datos1));  
posicion=0;  
for (int i=7; i<11; i++){  
    datos1[posicion] = DatosMedAut[i];  
    posicion++;  
}  
mhn=atol(datos1);
```

```
memset(datos1, 0, sizeof(datos1));  
posicion=0;  
for (int i=11; i<14; i++){  
    datos1[posicion] = DatosMedAut[i];  
    posicion++;  
}  
DistVert=atol(datos1);
```

```
memset(datos1, 0, sizeof(datos1));  
posicion=0;  
for (int i=14; i<18; i++){  
    datos1[posicion] = DatosMedAut[i];  
    posicion++;  
}  
mvp=atol(datos1);
```

```
memset(datos1, 0, sizeof(datos1));  
posicion=0;  
for (int i=18; i<22; i++){  
    datos1[posicion] = DatosMedAut[i];  
    posicion++;  
}  
mvn=atol(datos1);
```

```
memset(datos1, 0, sizeof(datos1));  
posicion=0;  
for (int i=22; i<27; i++){  
    datos1[posicion] = DatosMedAut[i];  
    posicion++;  
}  
TiempoMuestreo=atol(datos1);
```

```
memset(datos1, 0, sizeof(datos1));  
posicion=0;  
for (int i=27; i<29; i++){  
    datos1[posicion] = DatosMedAut[i];  
    posicion++;  
}  
TiempoEstabiliz=atol(datos1);
```



TiempoEstabiliz = TiempoEstabiliz*1000;

/// N° DE PASOS ENTRE MEDICIONES, DISTANCIA REAL Y DURACION DEL PROCESO DE MEDICIÓN ///

```
ph = DistHor*ConversorDistPasosHor;           // CANTIDAD DE PASOS COMO NUMERO ENTERO
phExtra = DistHor*ConversorDistPasosHor;       // CANTIDAD DE PASOS COMO NUMERO DECIMAL
phExtra = (phExtra - ph)*100;
if (phExtra > 49){                             // SI EL DECIMAL ES MAYOR A 0,5 SE AUMENTA
    ph++;                                       // UNA UNIDAD AL NUMERO ENTERO
}
pv = DistVert*ConversorDistPasosVert;
pvExtra = DistVert*ConversorDistPasosVert;
pvExtra = (pvExtra - pv)*100;
if (pvExtra > 49){
    pv++;
}

unsigned long mhpUL = mhp;    // NUMERO DE MEDICIONES HORIZONTALES POSITIVAS (Max valor mhp = 2000)
unsigned long mhnUL = mhn;    // NUMERO DE MEDICIONES HORIZONTALES NEGATIVAS (Max valor mhn = 2000)
unsigned long mvpUL = mvp;    // NUMERO DE MEDICIONES VERTICALES POSITIVAS (Max valor mvp = 2000)
unsigned long mvnUL = mvn;    // NUMERO DE MEDICIONES VERTICALES NEGATIVAS (Max valor mvn = 2000)
unsigned long phUL = ph;      // NUMERO DE PASOS ENTE MEDICIONES HORIZONTALES
                                // Max valor ph = Max DistHor/0,044(mm/paso) = 5683
unsigned long pvUL = pv;      // NUMERO DE PASOS ENTE MEDICIONES VERTICALES
                                // Max valor pv = Max DistVer/0,044(mm/paso) = 5683
unsigned long DelayUL = DelayAcel [PasosAcelMax]/10;    // TENDRIA QUE SER /1000 para pasar a mSeg.
                                                         // Los 100 faltante se agregan en los calculos

DuracionMedicAutom1 = phUL*(mhnUL+mhpUL) ;           // Max Valor: 400.000.000
DuracionMedicAutom1 = DuracionMedicAutom1/1000 ;     // Max Valor: 400.000
DuracionMedicAutom1 = DuracionMedicAutom1*2*DelayUL ; // Max Valor: 32.000.000
DuracionMedicAutom1 = DuracionMedicAutom1/60 ;       // Max Valor: 533.333
DuracionMedicAutom1 = DuracionMedicAutom1*(2*(mvnUL+mvpUL)+1) ; // Max Valor: 4.267.200.000
DuracionMedicAutom1 = DuracionMedicAutom1/100 ;      // Max Valor: 42.672.000

DuracionMedicAutom2 = (phUL*mhnUL+pvUL*(mvnUL+2*mvpUL)) ; // Max Valor: 800.000.000
DuracionMedicAutom2 = DuracionMedicAutom2/100 ;       // Max Valor: 8.000.000
DuracionMedicAutom2 = DuracionMedicAutom2*2*DelayUL ; // Max Valor: 640.000.000
DuracionMedicAutom2 = DuracionMedicAutom2/1000/60 ;   // Max Valor: 10.666

DuracionMedicAutom3 = (mhnUL+mhpUL+1)*(mvnUL+mvpUL+1) ; // Max Valor: 16.008.001 medic.
DuracionMedicAutom4 = (TiempoMuestreo+1000+TiempoEstabiliz)/1000 ; // Max Valor: 121 seg
DuracionMedicAutom5 = DuracionMedicAutom3 * DuracionMedicAutom4 ; // Max Valor: 1.936.968.121 seg
DuracionMedicAutom5 = DuracionMedicAutom5 / 60 ;      // Max Valor: 32.282.802 min

DuracionMedicAutom = DuracionMedicAutom1 + DuracionMedicAutom2 + DuracionMedicAutom5 + 1 ;
                                                         // Max Valor: 74.965.469
```



/// DETECCION DE DESBORDES ///

```
dhp1 = nhr+mhpUL*phUL;
if (dhp1 > nhgMax){
    dhp = 1;                                     // DESBORDE HORIZONTAL POSITIVO
}
else{
    dhp = 0;
}

dhn1 = nhr-mhnUL*phUL;
if (dhn1 < 0){
    dhn = 1;                                     // DESBORDE HORIZONTAL NEGATIVO
}
else{
    dhn = 0;
}

dvp1 = nvr+mvpUL*pvUL;
if (dvp1 > nvGMax){
    dvp = 1;                                     // DESBORDE VERTICAL POSITIVO
}
else{
    dvp = 0;
}

dvn1 = nvr-mvnUL*pvUL;
if (dvn1 < 0){
    dvn = 1;                                     // DESBORDE VERTICAL NEGATIVO
}
else{
    dvn = 0;
}
```

/// CONFIRMACION DEL USUARIO ///

```
Serial.print ('/');                               // SI IMPRIMEN LOS VALORES INGRESADOS, LOS
Serial.print ('A');                               // CALCULADOS Y LOS DESBORDES PARA SU
Serial.print (dhp); Serial.print (dhn); Serial.print (dvp); Serial.print (dvn);           // VISUALIZACION EN PANTALLA
Serial.print ('A');
Serial.print ('B'); Serial.print (DistHor);      Serial.print ('B');
Serial.print ('C'); Serial.print (mhp);          Serial.print ('C');
Serial.print ('D'); Serial.print (mhn);          Serial.print ('D');
Serial.print ('E'); Serial.print (DistVert);      Serial.print ('E');
Serial.print ('G'); Serial.print (mvp);          Serial.print ('G');
Serial.print ('H'); Serial.print (mvn);          Serial.print ('H');
```




```
Serial.print ('I'); Serial.print (TiempoMuestreo);
Serial.print ('J'); Serial.print (ph/ConversorDistPasosHor);
Serial.print ('K'); Serial.print (ph);
Serial.print ('N'); Serial.print (pv/ConversorDistPasosVert);
Serial.print ('O'); Serial.print (pv);
Serial.print ('Q'); Serial.print (DuracionMedicAutom);
Serial.print ('R'); Serial.print (TiempoEstabiliz/1000);
delay(100);

Serial.print ('I');
Serial.print ('J');
Serial.print ('K');
Serial.print ('N');
Serial.print ('O');
Serial.print ('Q');
Serial.print ('R');

if (dhp == 0 && dhn == 0 && dvp == 0 && dvn == 0){
    Serial.print ("@M2M");
}
else {
    Serial.print ("@M3M");
}

// INDICA A LA INTERFAZ QUE DESBLOQUEE LOS
// BOTONES SALIR, INICIAR MEDICIONES Y
// MODIFICAR VALORES

// INDICA A LA INTERFAZ QUE DESBLOQUEE SOLO
// LOS BOTONES SALIR Y MODIFICAR VALORES

while(Serial.available() == 0){
    delay(50);
}

// ESPERA LA ORDEN DEL USUARIO

Modificar = Serial.read();

// ALMACENA LA ORDEN

if (Modificar == 'I' || Modificar == 'O'){
    D = 1;
    Serial.print ("@M4M");
}

// Modificar == 'I' INICIAR MEDICIONES
// Modificar == 'M' MODIFICAR VALORES
// Modificar == 'O' SALIR

}

// SI Modificar = 'M' -> D = 0 EL BUCLE DE INGRESO DE VALORES DE REPITE
// SI Modificar = 'I' Ó 'O' -> D = 1 SE SALE DEL BUCLE DE INGRESO DE VALORES

if (Modificar == 'I' ){
    MEDAUTOM = 0;
}
else{
    MEDAUTOM = 1;
}

// SI Modificar == 'O' -> MEDAUTOM = 1 CON LO
// CUAL NO SE EJECUTARÁ LA ESTRUCTURA
// IF () DONDE DE REALIZA EL PROCESO DE
// MEDICION

if (MEDAUTOM == 0){

    // ESTRUCTURA DONDE SE EJECUTA EL PROCESO
    // DE MEDICIÓN

    /// ----- RETROCESO -----

    nvFinal = nvr - mvn*pv;
    nhFinal = nhr - mhn*ph;
    IrACoordenada ();
    delay (50);
    ImprimirPosicion();

    // SE DETERMINA LAS COORDENADAS DEL PUNTO
    // FINAL DEL DESPLAZAMIENTO
```



/// ----- [6.2] PROCESO DE MEDICIÓN -----

```
int NumMedicion = 0; // CONTADOR QUE ALMACENARÁ EL NUMERO DE
// MEDICIONES REALIZADAS

for (Y = -1*mvn; Y <= mvp; Y++){
    for (X = -1*mhn; X <= mhp; X++){
        NumMedicion ++;
        delay (1000); // PRIMEROS 1000 mSeg DEL TIEMPO DE ESTABILIZACION
        Serial.print("¿Z");
        Serial.print ("("); Serial.print (X); Serial.print (" ; "); Serial.print (Y); Serial.print (" )");
        // ENVIA A LA INTERFAZ LA COORDENADA DEL PUNTO ACTUAL DE MEDICION
        Serial.print (NumMedicion*100/((mhn+mhp+1)*(mvn+mvp+1))); Serial.print('%');
        // ENVIA A LA INTERFAZ EL PORCENTAJE YA REALIZADO DEL PROCESO
        Serial.print('Z');
        delay (TiempoEstabiliz-1000); // COMPLETA EL TIEMPO DE ESTABILIZACION
        digitalWrite (RealizarMedicion, HIGH); // ACTIVA LA SEÑAL DE REFERENCIA
        Serial.print ("@M5M"); // INDICA A LA INTERFAZ QUE INICIO EL MUESTREO
        delay (TiempoMuestreo); // TIEMPO DE MUESTREO
        digitalWrite (RealizarMedicion, LOW); // DESACTIVA LA SEÑAL DE REFERENCIA
        Serial.print ("@M6M"); // INDICA A LA INTERFAZ QUE FINALIZO EL MUESTREO
        // DEL PUNTO ACTUAL

        delay (1000);

        if (digitalRead(Salir) == HIGH || (Serial.read()-48)==0){
            Serial.println ("¿ZCANCELANDOZ");
            delay (100);
            X = mhp+1; // CONDICIONES DE SALIDA PARA FINALIZAR
            Y = mvp+1; // ANTICIPADAMENTE LA FUNCION
        }

        if (X < mhp){ // AVANCE HORIZONTAL HASTA EL SIGUIENTE PUNTO
            // DE MEDICIÓN DE LA FILA
            nvFinal = nv; // SE DETERMINA LAS COORDENADAS DEL PUNTO FINAL
            nhFinal = nhg + ph; // DEL DESPLAZAMIENTO
            IrACoordenada ();
            if (digitalRead(Salir) == HIGH || (Serial.read()-48)==0){
                Serial.println ("¿ZCANCELANDOZ");
                delay (100);
                X = mhp+1; // CONDICIONES DE SALIDA PARA FINALIZAR
                Y = mvp+1; // ANTICIPADAMENTE LA FUNCION
            }
            delay (50);
            ImprimirPosicion();
        }
    }
}
```



```
        if (Y < mvp){                                     // RETROCESO HORIZONTAL HASTA EL PRIMER PUNTO
                                                            // DE MEDICION DE LA FILA
            nvFinal = nvg + pv;                             // SE DETERMINA LAS COORDENADAS DEL PUNTO
            nhFinal = nhr - mhn*ph;                         // FINAL DEL DESPLAZAMIENTO
            IrACoordenada ();
            if (digitalRead(Salir) == HIGH || (Serial.read()-48)==0){
                Serial.println ("¿ZANCELANDO?");
                delay (100);
                X = mhp+1;                                   // CONDICIONES DE SALIDA PARA FINALIZAR
                Y = mvp+1;                                   // ANTICIPADAMENTE LA FUNCION
            }
            delay (50);
            ImprimirPosicion();
        }
    }

/// ----- REGRESO AL PUNTO INICIAL -----

    nvFinal = nvr;                                         // SE DETERMINA LAS COORDENADAS DEL PUNTO
    nhFinal = nhr;                                         // FINAL DEL DESPLAZAMIENTO
    IrACoordenada ();
    delay (50);
    ImprimirPosicion();

    Serial.print ("@M7M");                                // SE INDICA A LA INTERFAZ QUE EL PROCESO DE
                                                            // MEDICION A FINALIZADO

}                                                         // FINAL DEL BUCLE DE MEDICION AUTOMATICA

delay (150);
Serial.print ("@M8M");                                    // SE INDICA A LA INTERFAZ QUE LA FUNCIÓN A FINALIZADO
for (int i=0; i<65; i++){                                // LEE LOS PUERTOS SERIE PARA VACIAR SUS BUFFERS DE
    Serial.read();                                        // ENTRADA PARA VACIARLOS
    Serial2.read();
}
Funcion =0;
digitalWrite (LuzMedAutomatica, LOW);
}

/// ----- FUNCION 7: IR A COORDENADA -----

while (Funcion == 7){

    digitalWrite (LuzIrACoordenada, HIGH);
    digitalWrite (LuzSalir, LOW);
    Serial.print ("@F7F");                                // SE INDICA A LA INTERFACES QUE LA FUNCION
    Serial2.print ('4');                                   // ESTA EN EJECUCION
```



```
Serial.print("@I1I");          // SE INDICA A LA INTERFAZ QUE LA FUNCION ESTA EN ESPERA DE ORDENES
delay(100);
for (int i=0; i<50; i++){
    Serial.read ();
}

/// INGRESO DE DATOS ///

while(Serial.available() == 0){          // ESPERA DE ORDENES
    delay(50);
}

// Coordenadas == 'A' SE ENVIARAN LAS COORDENADAS EN PASOS
Coordenadas = Serial.read();           // Coordenadas == 'B' SE ENVIARAN LAS COORDENADAS EN mm
// Coordenadas == '0' SALIR

if (Coordenadas == 'A' || Coordenadas == 'B'){

    Serial.print("@I2I");              // SE INDICA A LA INTERFAZ QUE EL POSICIONADOR SE ESTA POR
                                        // DIRIGIR A LA POSICION SOLICITADA
    while(Serial.available() < 12){    // ESPERA LAS COORDENADAS
        delay(200);
    }

    memset(DatosIrACoord, 0, sizeof(DatosIrACoord));
    posicion = 0;
    while (Serial.available())>0){
        delay (5);
        DatosIrACoord[posicion] = (Serial.read());          // SE ALMACENAN LOS 12 CARACTERES EN EL
        posicion++;                                          // STRING DatosIrACoord[]
    }

    memset(datos2, 0, sizeof(datos2));
    posicion=0;
    for (int i=0; i<6; i++){
        datos2[posicion] = DatosIrACoord[i];                // MEDIANTE EL STRING datos2[] SE DESCOMPONE
        posicion++;                                          // EL STRING DatosIrACoord[] PARA SEPARAR LOS
                                                            // DATOS CONTENIDOS EN EL Y ALMACENARLOS EN
                                                            // SUS VARIABLES CORRESPONDIENTES
    }
    if (Coordenadas == 'A'){
        nh3 = atol(datos2);                                  // SE ALMACENA LA COORDENADA DIRECTAMENTE
                                                            // EN LA VARIABLE
    }
    else{
        nh3Extra = ConversorDistPasosHor * atol(datos2)/10; // CANTIDAD DE PASOS COMO NUMERO DECIMAL
        nh3 = nh3Extra;                                       // CANTIDAD DE PASOS COMO NUMERO ENTERO
        nh3Extra = (nh3Extra - nh3)*100;
        if (nh3Extra > 49){
            nh3++;                                            // SI EL DECIMAL ES MAYOR A 0,5 SE AUMENTA
                                                            // UNA UNIDAD AL NUMERO ENTERO
        }
    }
}
```



```
}

memset(datos2, 0, sizeof(datos2));
posicion=0;
for (int i=6; i<12; i++){
    datos2[posicion] = DatosIrACoord[i];
    posicion++;
}
if (Coordenadas == 'A'){
    nv3 = atol(datos2);
}
else{
    nv3Extra = ConversorDistPasosVert * atol(datos2)/10;
    nv3 = nv3Extra;
    nv3Extra = (nv3Extra - nv3)*100;
    if (nv3Extra > 49){
        nv3++;
    }
}

// VERIFICA SI LAS COORDENADAS INGRESADAS
// ESTAN DENTRO DE LA ZONA DE TRABAJO
if (nv3 <= nvgMax && nh3 <= nhgMax){
    nvFinal = nv3;
    nhFinal = nh3;
    IrACoordenada ();
    delay (50);
    ImprimirPosicion();
}
// ASIGNA LAS COORDENADAS
// DEL PUNTO FINAL DEL DESPLAZAMIENTO

else {
    // SI LAS COORDENADAS SE ENCUENTRAN FUERA
    // DE LA ZONA DE TRABAJO SE IMPRIME EN
    if (Coordenadas == 'A'){
        Serial.print("+X");
        Serial.print ("Coordenada fuera de rango. Ingrese valores Horizontales entre (0 - ");
        Serial.print (nhgMax); Serial.print (") pasos y valores Verticales entre (0 - ");
        Serial.print (nvgMax); Serial.print (") pasos");
        Serial.print('X');
        // LA PANTALLA DE LA APLICACIÓN DE LabVIEW
        // LA ADVERTENCIA Y LOS RANGOS POSIBLES
    }
    else {
        // DE CADA VARIABLE SEGÚN SEAN PASOS O mm
        Serial.print("+X");
        Serial.print ("Coordenada fuera de rango. Ingrese valores Horizontales entre (0 - ");
        Serial.print (nhgMax/ConversorDistPasosHor, 0); Serial.print (") mm y valores Verticales entre (0 - ");
        Serial.print (nvgMax/ConversorDistPasosVert, 0); Serial.print (") mm");
        Serial.print('X');
    }
}

for (int i=0; i<65; i++){
    Serial.read();
    Serial2.read();
}

// LEE LOS PUERTOS SERIE PARA VACIAR SUS
// BUFFERS DE ENTRADA
```



```
        delay (50);
    }

    else{
        Funcion = 0;
    }
}
digitalWrite (LuzIrACoordenada, LOW);
```

```
/// ----- FUNCION 0: NINGUNA FUNCIÓN EN EJECUCIÓN -----
```

```
if (CeroRealizado == 1){
    Serial.print ("@F0F");
    Serial2.print ('0');
    digitalWrite (LuzSalir, HIGH);
}
else {
    Serial.print ("@F8F");
    Serial2.print ('8');
    digitalWrite (LuzSalir, HIGH);
}
delay (300);

} // FIN DEL void loop ()
```

```
void ImprimirPosicion(){
```

```
    Serial.print ('#');
    Serial.print ('A'); Serial.print (nvg); Serial.print ('A');
    Serial.print ('B'); Serial.print (nhg); Serial.print ('B');
} // FIN DEL void ImprimirPosicion ()
```

```
void IrACoordenada (){
```

```
    V = nvg - nvFinal;
    H = nhg - nhFinal;

    if (V < 0){
        digitalWrite (SalDirPosVert, HIGH);
    }
    else {
        digitalWrite (SalDirPosVert, LOW);
    }
    if (H < 0){
```



```
        digitalWrite (SalDirPosHor, HIGH);                                // COMUN
    }
    else {
        digitalWrite (SalDirPosHor, LOW);
    }

    if (abs(V) < abs(H)){
        nAuxFinal1 = abs(V);                                              // SE DETERMINA LA AMPLITUD DEL
        nAuxFinal2 = abs(H) - abs(V);                                    // DESPLAZAMIENTO COMUN nAuxFinal1 Y
        MovIndep = SalPasoHor;                                           // AMPLITUD DEL DESPLAZAMIENTO
    }                                                                    // INDEPENDIENTE nAuxFinal2
    else {
        nAuxFinal1 = abs(H);
        nAuxFinal2 = abs(V) - abs(H);
        MovIndep = SalPasoVert;
    }

// DESPLAZAMIENTO COMUN

    if (nAuxFinal1 >= 2*PasosAcelMax){                                    // SE DETERMINA CANTIDAD DE PASOS DE LA
        PasosAcel = PasosAcelMax;                                       // ACELERACION
    }
    else {
        PasosAcel = (nAuxFinal1 - 1)/2 ;
    }
    nAux = 0;

    for (int i=1; i<=PasosAcel; i++){                                    // RAMPA DE ACELERACION DEL
        digitalWrite (SalPasoVert, HIGH);                                // DESPLAZAMIENTO COMUN
        digitalWrite (SalPasoHor, HIGH);
        delayMicroseconds (DelayAcel [i]);
        digitalWrite (SalPasoVert, LOW);
        digitalWrite (SalPasoHor, LOW);
        delayMicroseconds (DelayAcel [i]);
        nAux ++;
        If (V < 0){nvg ++;}
        else {nvg --;}
        if (H < 0){nhg ++;}
        else {nhg --;}
    }

    while (CondSiempreVerdadera == 1){                                    // AVANCE CONTINUO DEL
                                                                    // DESPLAZAMIENTO COMUN

        if (nAux + PasosAcel >= nAuxFinal1){
            break;
        }
        digitalWrite (SalPasoVert, HIGH);
```



```
digitalWrite (SalPasoHor, HIGH);
delayMicroseconds (DelayAcel [PasosAcel]);
digitalWrite (SalPasoVert, LOW);
digitalWrite (SalPasoHor, LOW);
delayMicroseconds (DelayAcel [PasosAcel]);
nAux ++;
if (V < 0){nvg ++;}
else {nvg --;}
if (H < 0){nhg ++;}
else {nhg --;}

if (Funcion != 6){
    if (Serial.read() == 'P'){
        ImprimirPosicion();
    }
}
}

for (int i=PasosAcel-1; i>=0; i--){
    digitalWrite (SalPasoVert, HIGH);
    digitalWrite (SalPasoHor, HIGH);
    delayMicroseconds (DelayAcel [i]);
    digitalWrite (SalPasoVert, LOW);
    digitalWrite (SalPasoHor, LOW);
    delayMicroseconds (DelayAcel [i]);
    nAux ++;
    if (V < 0){nvg ++;}
    else {nvg --;}
    if (H < 0){nhg ++;}
    else {nhg --;}
}
delay (200);

// DESPLAZAMIENTO INDEPENDIENTE

if (nAuxFinal2 >= 2*PasosAcelMax){
    PasosAcel = PasosAcelMax;
}
else {
    PasosAcel = (nAuxFinal2 - 1)/2 ;
}
nAux = 0;

for (int i=1; i<=PasosAcel; i++){
    digitalWrite (MovIndep, HIGH);
    delayMicroseconds (DelayAcel [i]);
    digitalWrite (MovIndep, LOW);
```

// RAMPA DE DESACELERACION DEL
// DESPLAZAMIENTO COMUN

// SE DETERMINA CANTIDAD DE PASOS DE LA
// ACELERACION

// RAMPA DE ACELERACION DEL
// DESPLAZAMIENTO INDEPENDIENTE



```
    delayMicroseconds (DelayAcel [i]);
    nAux ++;
    if (MovIndep == SalPasoVert){
        if (V < 0){nvg ++;}
        else {nvg --;}
    }
    else {
        if (H < 0){nhg ++;}
        else {nhg --;}
    }
}

while (CondSiempreVerdadera == 1){
    if (nAux + PasosAcel >= nAuxFinal2){
        break;
    }
    digitalWrite (MovIndep, HIGH);
    delayMicroseconds (DelayAcel [PasosAcel]);
    digitalWrite (MovIndep, LOW);
    delayMicroseconds (DelayAcel [PasosAcel]);
    nAux ++;
    if (MovIndep == SalPasoVert){
        if (V < 0){nvg ++;}
        else {nvg --;}
    }
    else {
        if (H < 0){nhg ++;}
        else {nhg --;}
    }
    if (Funcion != 6){
        if (Serial.read() == 'P'){
            ImprimirPosicion();
        }
    }
}

for (int i=PasosAcel-1; i>=0; i--){
    digitalWrite (MovIndep, HIGH);
    delayMicroseconds (DelayAcel [i]);
    digitalWrite (MovIndep, LOW);
    delayMicroseconds (DelayAcel [i]);
    nAux ++;
    if (MovIndep == SalPasoVert){
        if (V < 0){nvg ++;}
        else {nvg --;}
    }
}
```

// AVANCE CONTINUO DEL
// DESPLAZAMIENTO INDEPENDIENTE

// RAMPA DE DESACELERACION DEL
// DESPLAZAMIENTO INDEPENDIENTE



```
    else {  
        if (H < 0){nhg ++;}  
        else {nhg --;}  
    }  
}  
  
if (Funcion != 6){  
    for (int i=0; i<65; i++){  
        Serial.read();  
        Serial2.read();  
    }  
}  
}  
  
// FIN DEL void IrACoordenada ()
```



ANEXO D

PLANOS CONSTRUCTIVOS



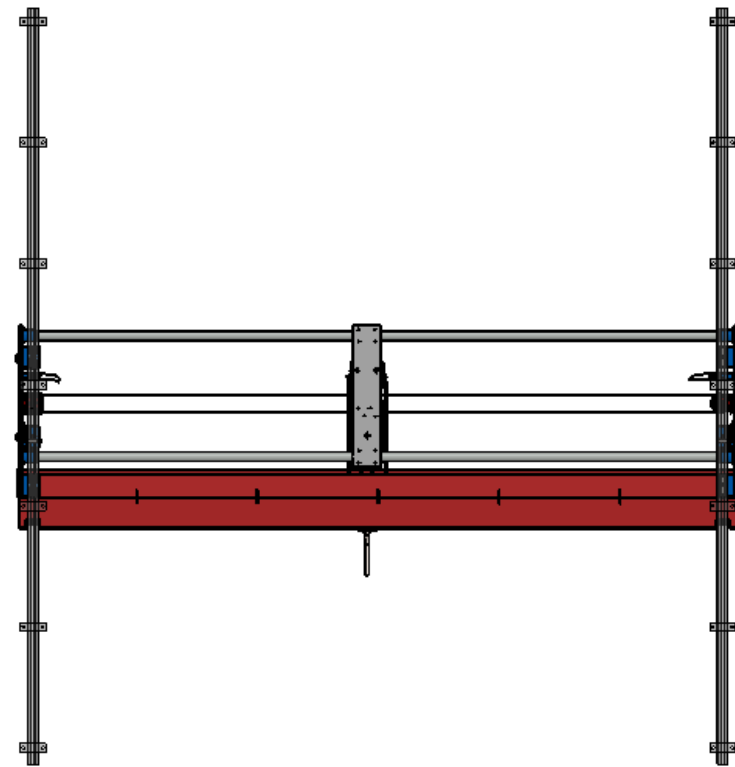
D.1 ÍNDICE DE PLANOS

NOMBRE	TIPO	REF.	N° PLANO
POSICIONADOR COMPLETO	ENSAMBLAJE	POS.1	1
POSICIONADOR COMPLETO – PARTES	ENSAMBLAJE	POS.2	2
EJE VERTICAL	ENSAMBLAJE	EV.1	3
EJE VERTICAL – CUERPO	ENSAMBLAJE	EV.2	4
EJE VERTICAL - EXTREMO SUPERIOR	ENSAMBLAJE	EV.3	5
EJE VERTICAL - EXTREMO INFERIOR	ENSAMBLAJE	EV.4	6
EJE VERTICAL – MOTOR	ENSAMBLAJE	EV.5	7
EJE VERTICAL - POLEA DE RETORNO	ENSAMBLAJE	EV.6	8
EJE VERTICAL - SENSOR DE POS. Y FINES DE CARRERA	ENSAMBLAJE	EV.7	9
EJE VERTICAL - CHAPA DE PROTECCIÓN	ENSAMBLAJE	EV.8	10
EJE HORIZONTAL SUPERIOR	ENSAMBLAJE	EH.1 (S)	11
EJE HOR. SUPERIOR – CUERPO	ENSAMBLAJE	EH.2 (S)	12
EJE HOR. SUPERIOR - EXTREMO IZQUIERDO	ENSAMBLAJE	EH.3 (S)	13
EJE HOR. SUPERIOR - EXTREMO DERECHO	ENSAMBLAJE	EH.4 (S)	14
EJE HOR. SUPERIOR - SOPORTE CENTRAL	ENSAMBLAJE	EH.5 (S)	15
EJE HORIZONTAL INFERIOR	ENSAMBLAJE	EH.6 (I)	16
EJE HOR. INFERIOR – CUERPO	ENSAMBLAJE	EH.7 (I)	17
EJE HOR. INFERIOR - EXTREMO IZQUIERDO	ENSAMBLAJE	EH.8 (I)	18
EJE HOR. INFERIOR - EXTREMO DERECHO	ENSAMBLAJE	EH.9 (I)	19
EJE HOR. INFERIOR - SOPORTE CENTRAL	ENSAMBLAJE	EH.10 (I)	20
EJE HORIZONTAL – MOTOR	ENSAMBLAJE	EH.11	21
EJE HORIZONTAL - POLEA DE RETORNO	ENSAMBLAJE	EH.12	22
EJE HORIZONTAL - SENSOR DE POS. Y FIN DE CARRERA	ENSAMBLAJE	EH.13	23
EJE HORIZONTAL - CARRO INFERIOR	ENSAMBLAJE	EH.14	24
PORTAINSTRUMENTO	ENSAMBLAJE	PI	25
PORTE MÓVIL – EJE VERT. Y SOPORTES CENTRALES HOR.	ENSAMBLAJE	PM	26
EJE VERTICAL GUÍA	PIEZA	P1	27
SOPORTE VERTICAL SUPERIOR	PIEZA	P2	28
SOPORTE VERTICAL INFERIOR	PIEZA	P3	29
SOPORTE DE EJE VERTICAL GUÍA	PIEZA	P4	30
SOPORTE DE CHAPA DE PROTECCIÓN VERTICAL 1	PIEZA	P5	31
SOPORTE DE CHAPA DE PROTECCIÓN VERTICAL 2	PIEZA	P6	32
SOPORTE DE CADENA PORTACABLES VERTICAL	PIEZA	P7	33
PLACA DE APOYO DEL EJE VERTICAL	PIEZA	P8	34
SOPORTE MOTOR VERTICAL	PIEZA	P9	35
SOPORTE POLEA DE RETORNO VERTICAL	PIEZA	P10	36
EJE POLEA DE RETORNO VERTICAL	PIEZA	P11	37
CASQUILLO POLEA DE RETORNO VERTICAL	PIEZA	P12	37

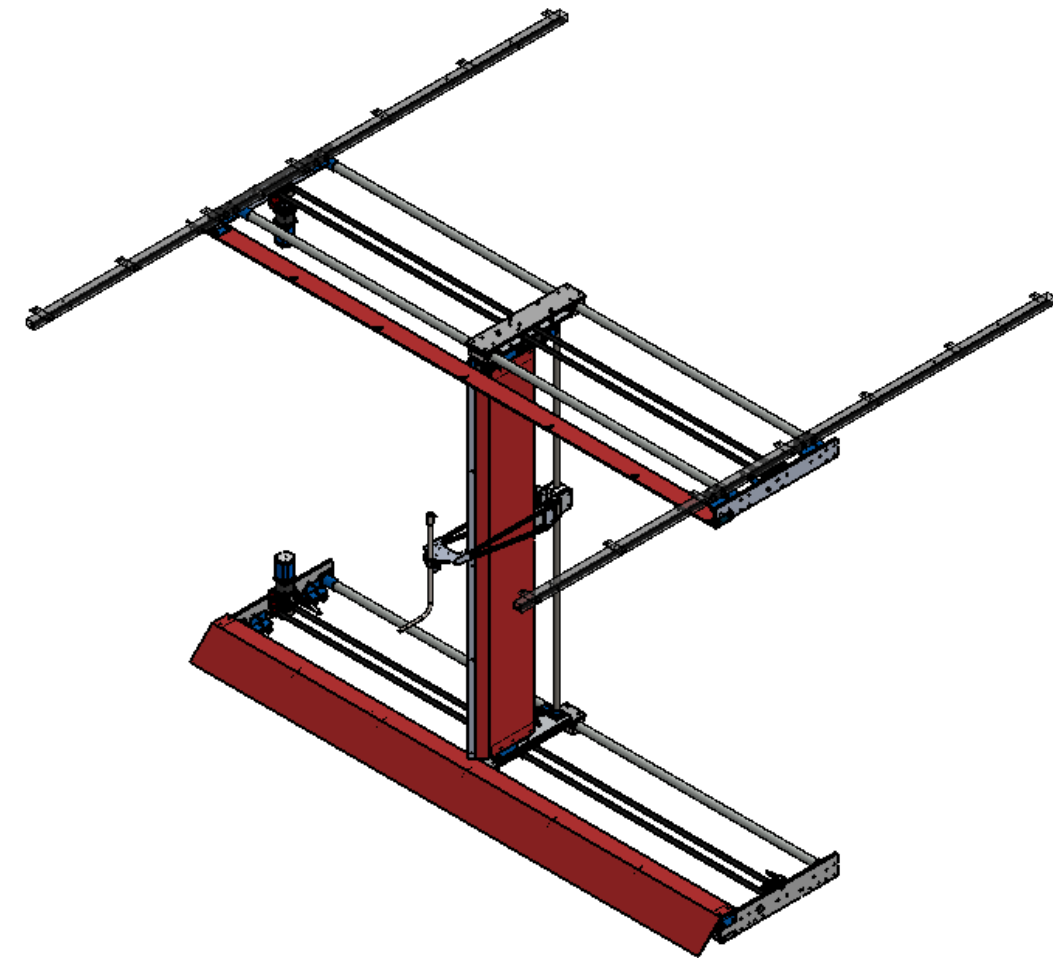
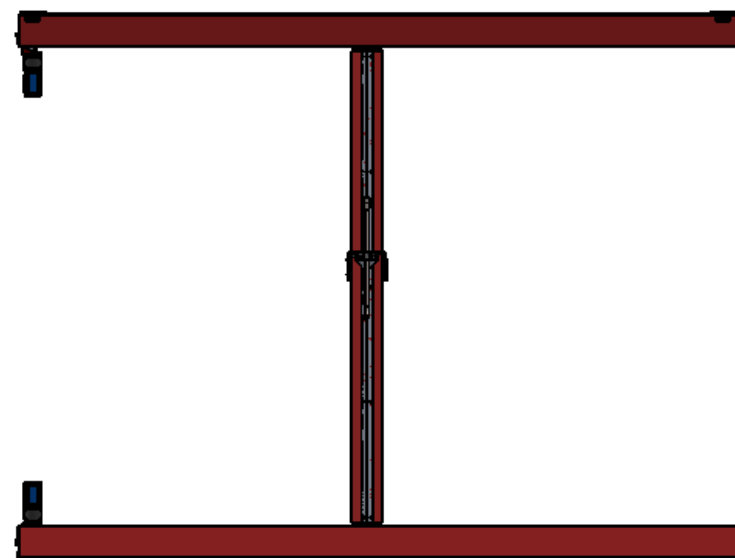


SOPORTE DE SENSOR DE POSICIÓN Y FIN DE CARRERA	PIEZA	P13	38
SOPORTE DE FIN DE CARRERA	PIEZA	P14	38
CHAPA DE PROTECCIÓN VERTICAL – MITAD	PIEZA	P15	39
PERFIL FRONTAL	PIEZA	P16	40
CHAPA DE PROTECCIÓN HORIZONTAL SUPERIOR	PIEZA	P17	41
RIEL PLANO	PIEZA	P18	42
EJE HORIZONTAL GUÍA	PIEZA	P19	43
SUPORTE HORIZONTAL EJE SUPERIOR	PIEZA	P20	44
SOPORTE DE EJE HORIZONTAL GUÍA	PIEZA	P21	45
SOPORTE CARRO SUPERIOR	PIEZA	P22	46
SOPORTE DE CHAPA DE PROTECCIÓN HOR. SUPERIOR	PIEZA	P23	47
TRABA AXIAL EJE HOR. SUPERIOR	PIEZA	P24	48
SOPORTE CENTRAL EJE HOR. SUPERIOR	PIEZA	P25	49
BUJES PARA EJES HORIZONTALES	PIEZA	P26	50
SUJECCIÓN DE CORREA HORIZONTAL	PIEZA	P27	51
CHAPA DE PROTECCIÓN HORIZONTAL INFERIOR	PIEZA	P28	52
SUPORTE HORIZONTAL EJE INFERIOR	PIEZA	P29	53
SOPORTE DE CHAPA DE PROTECCIÓN HOR. INFERIOR	PIEZA	P30	54
SOPORTE DE CADENA PORTACABLES HOR.	PIEZA	P31	55
ACCIONAMIENTO DE SENSOR Y FIN CARRERA	PIEZA	P32	56
ACCIONAMIENTO DE FIN DE CARRERA	PIEZA	P33	57
SOPORTE CENTRAL EJE HOR. INFERIOR	PIEZA	P34	58
SUJECCIÓN CADENA PORTACABLES HORIZONTAL	PIEZA	P35	59
SOPORTE MOTOR HORIZONTAL	PIEZA	P36	60
SOPORTE POLEA HORIZONTAL	PIEZA	P37	61
EJE POLEA MOTORA HORIZONTAL	PIEZA	P38	62
EJE POLEA DE RETORNO HORIZONTAL	PIEZA	P39	62
SOPORTE POLEA RETORNO HORIZONTAL	PIEZA	P40	63
SOPORTE CARRO INFERIOR	PIEZA	P41	64
TRABA AXIAL EJE HOR. INFERIOR	PIEZA	P42	65
PORTAINSTRUMENTO - BRAZO DERECHO	PIEZA	P43	66
PORTAINSTRUMENTO - BRAZO IZQUIERDO	PIEZA	P44	67
PORTAINSTRUMENTO - PLACA SOPORTE DERECHA	PIEZA	P45	68
PORTAINSTRUMENTO - PLACA SOPORTE IZQUIERDA	PIEZA	P46	69
BUJES PARA EJES VERTICALES	PIEZA	P47	70
SUJECCIÓN DE CORREA VERTICAL	PIEZA	P48	71
SUJECCIÓN CADENA PORTACABLES VERTICAL	PIEZA	P49	72
PORTAINSTRUMENTO - UNIÓN ENTRE BRAZOS	PIEZA	P50	73

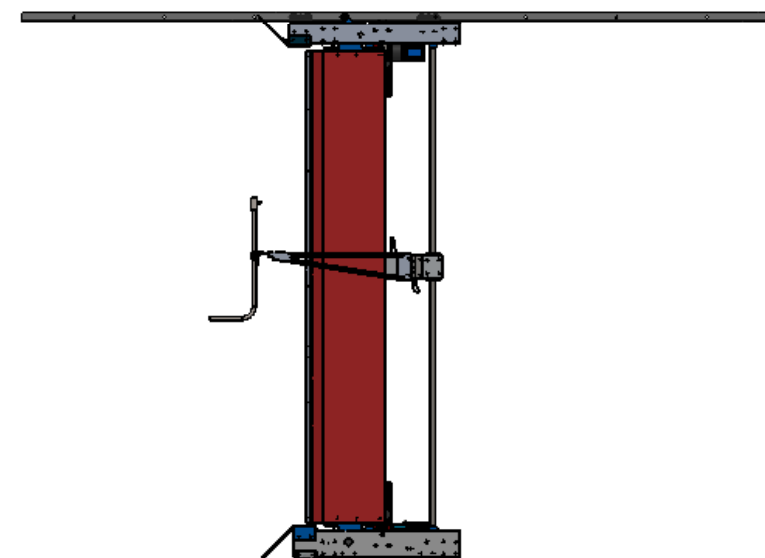
VISTA SUPERIOR



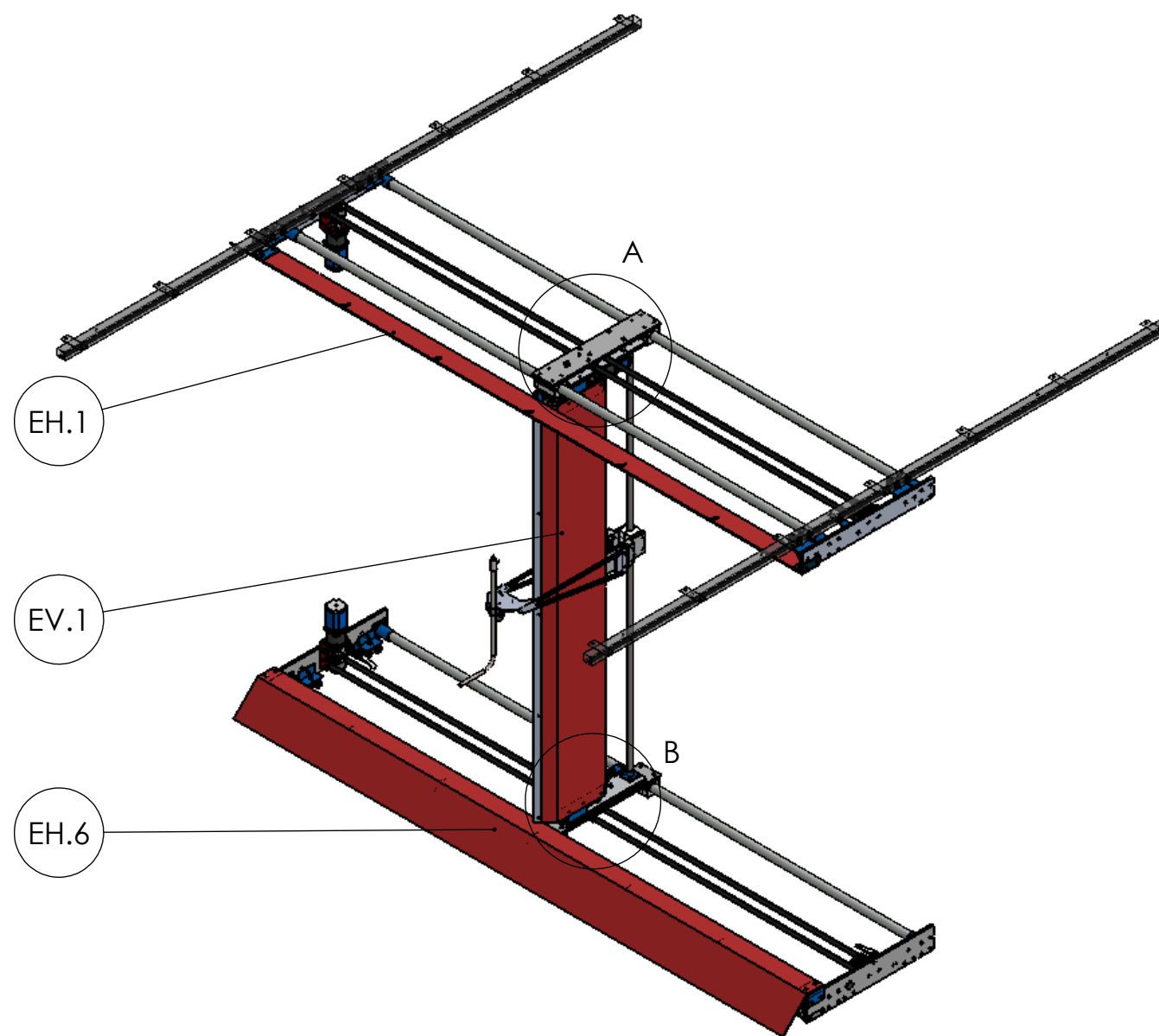
VISTA FRONTAL



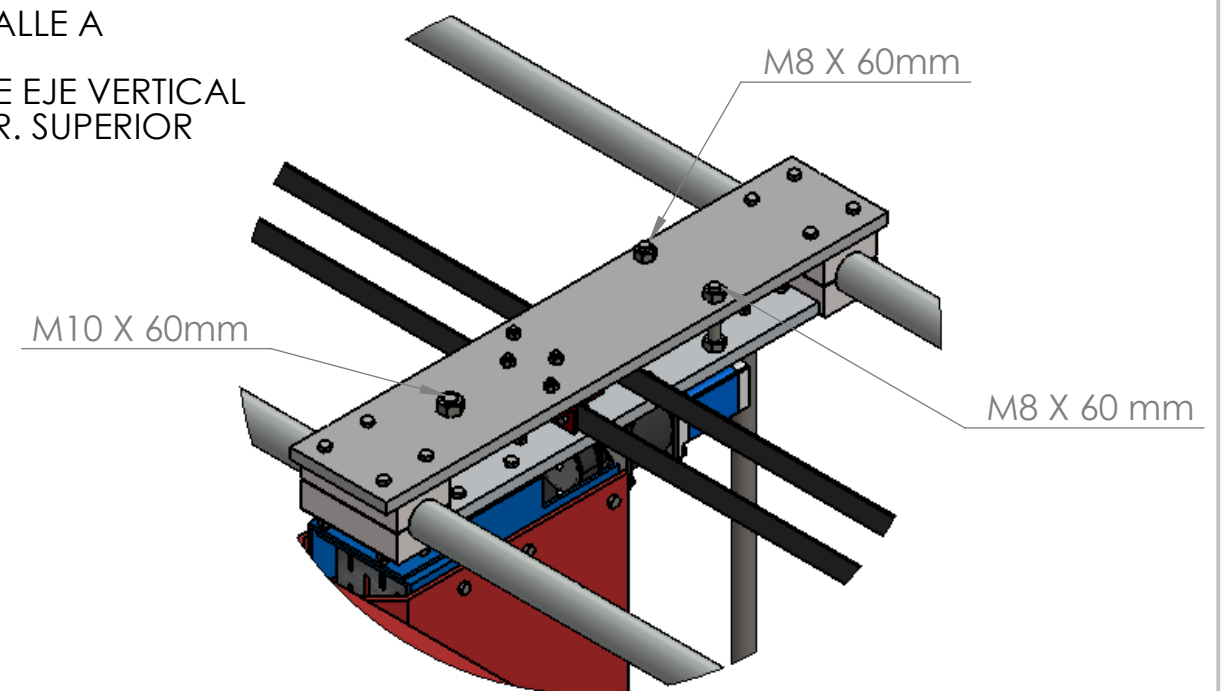
VISTA LATERAL



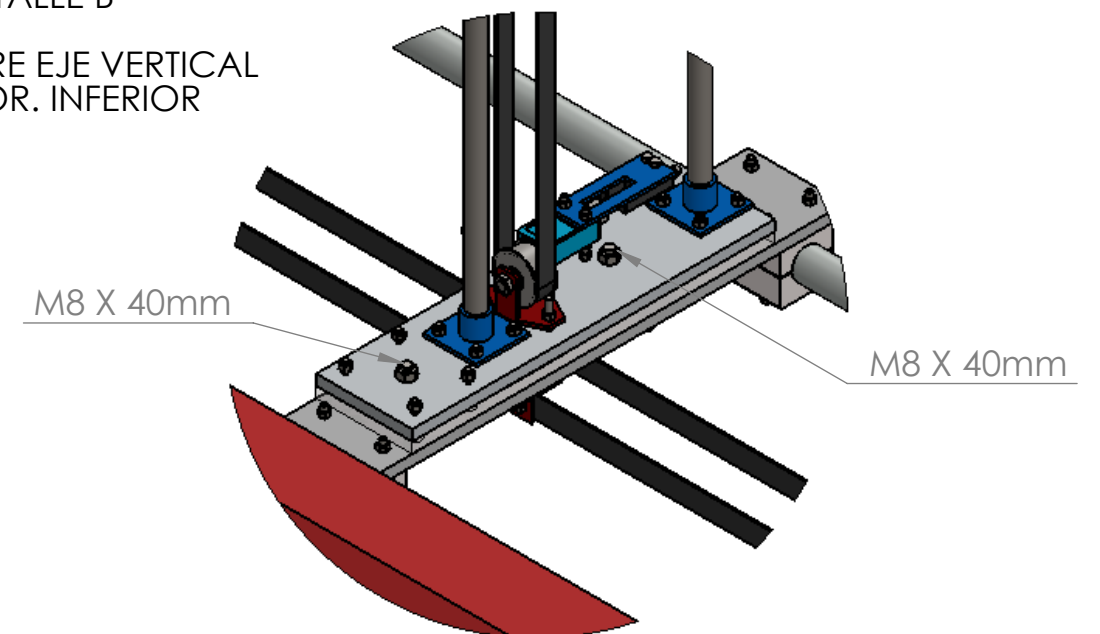
 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA	NOMBRE: POSICIONADOR COMPLETO	
DISEÑO: OBREGON MARCELO JOAQUIN	REFERENCIA: POS.1	
AÑO: 2022	ESCALA: S/E	HOJA: 1/73
MATERIAL: -		



DETALLE A
UNION ENTRE EJE VERTICAL
Y EJE HOR. SUPERIOR

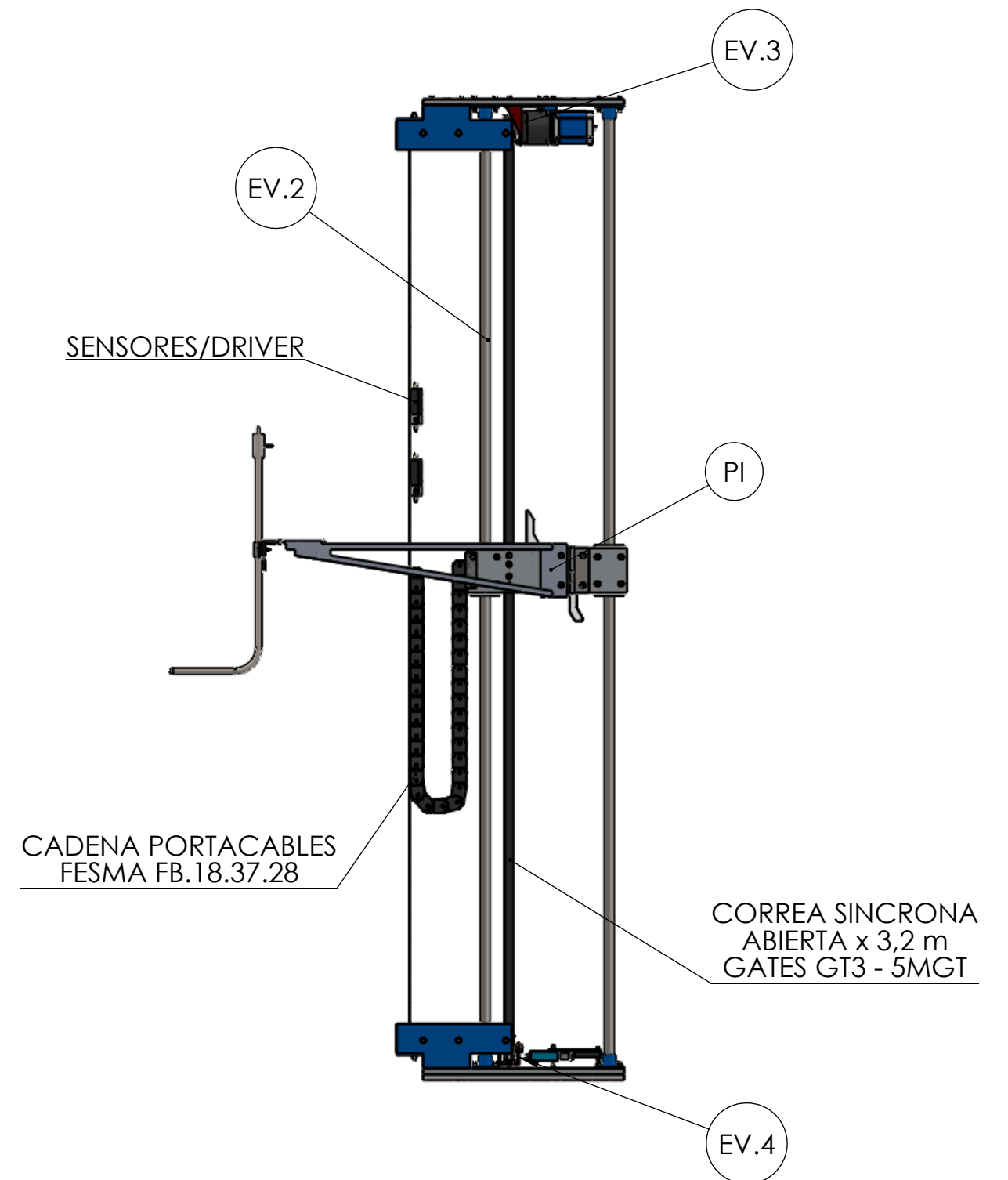
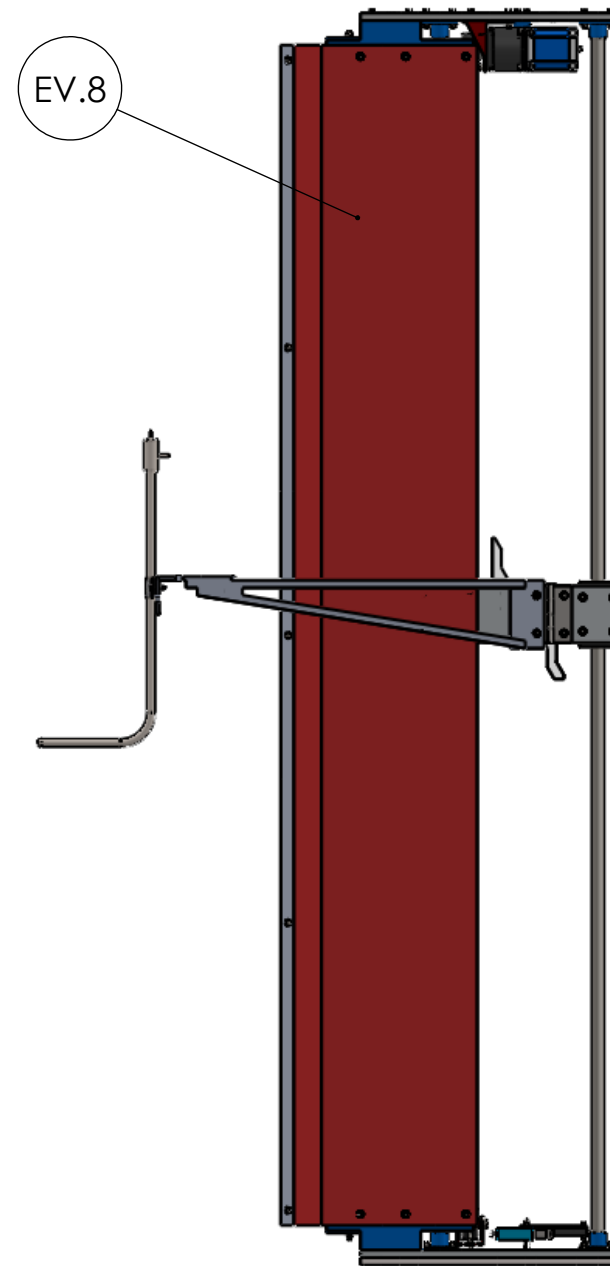
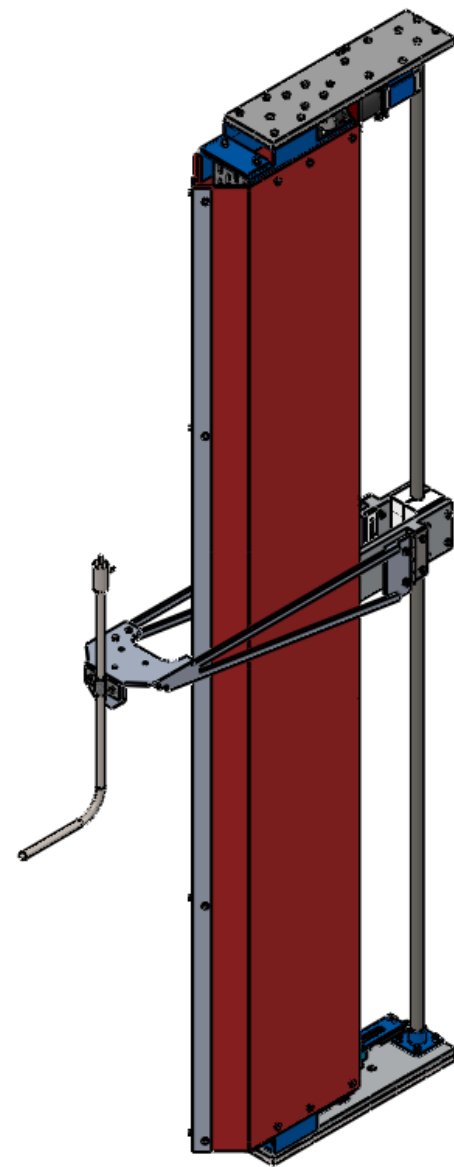


DETALLE B
UNION ENTRE EJE VERTICAL
Y EJE HOR. INFERIOR



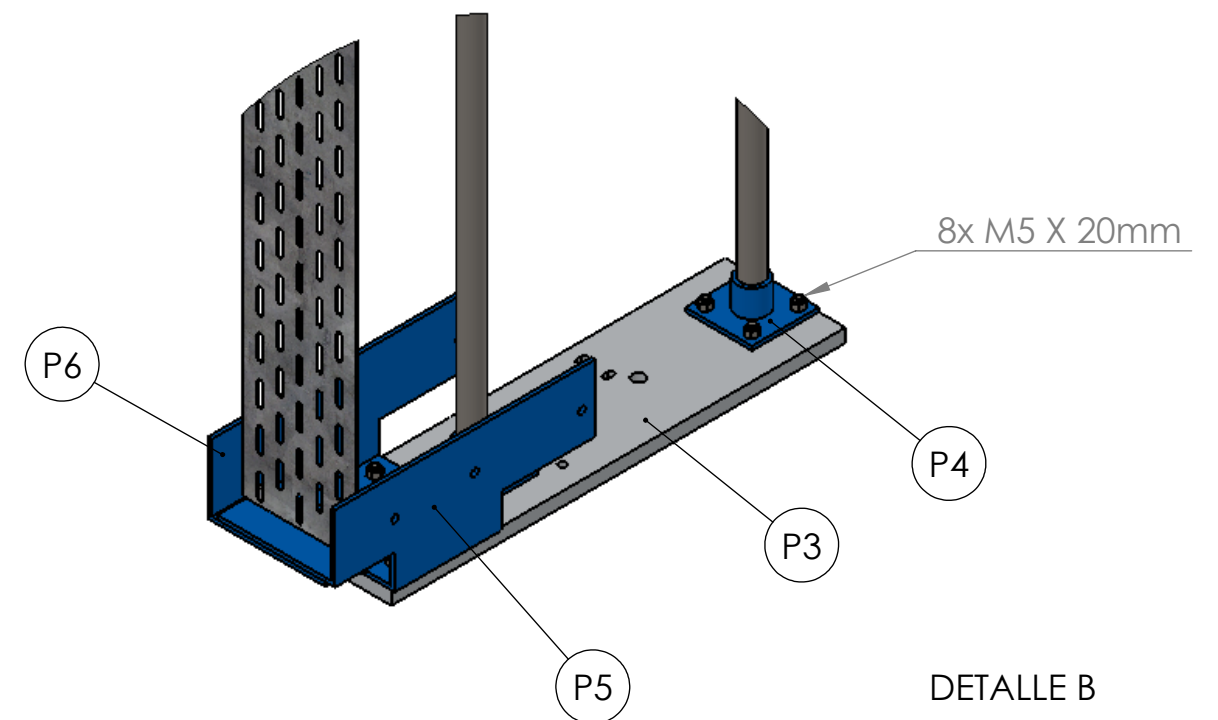
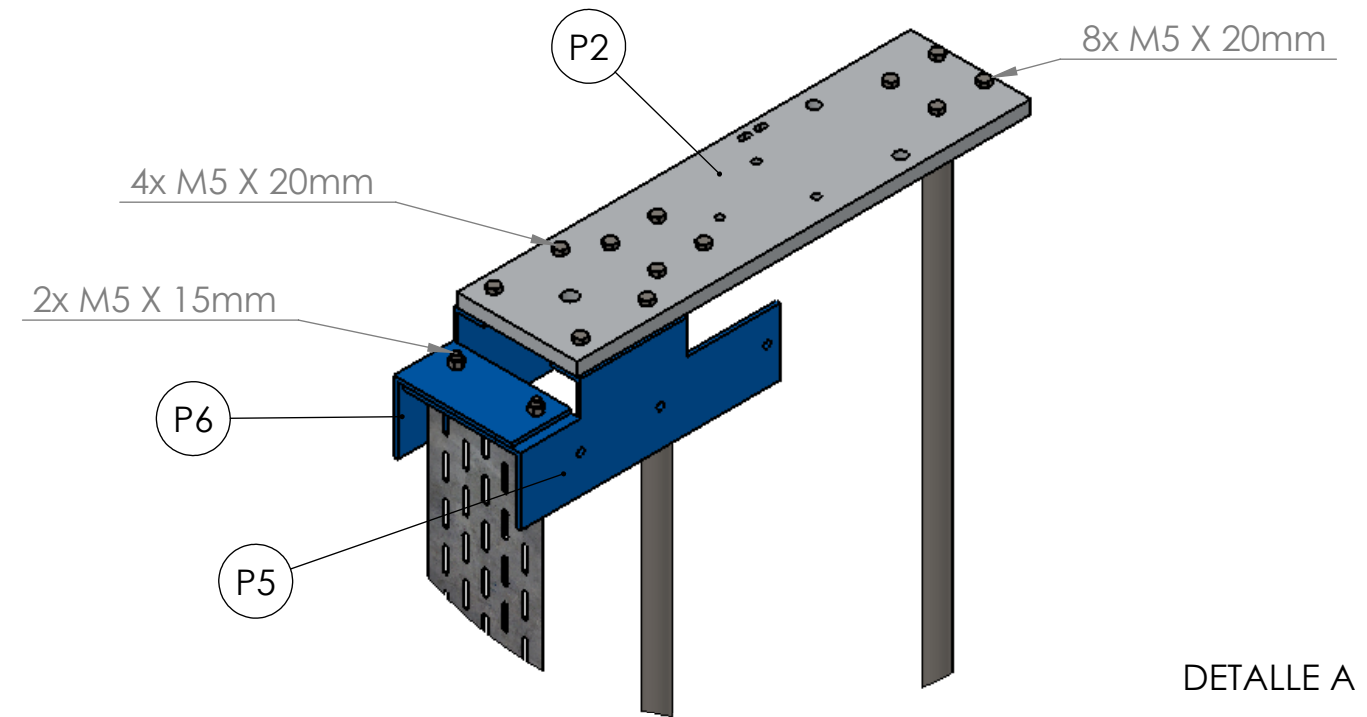
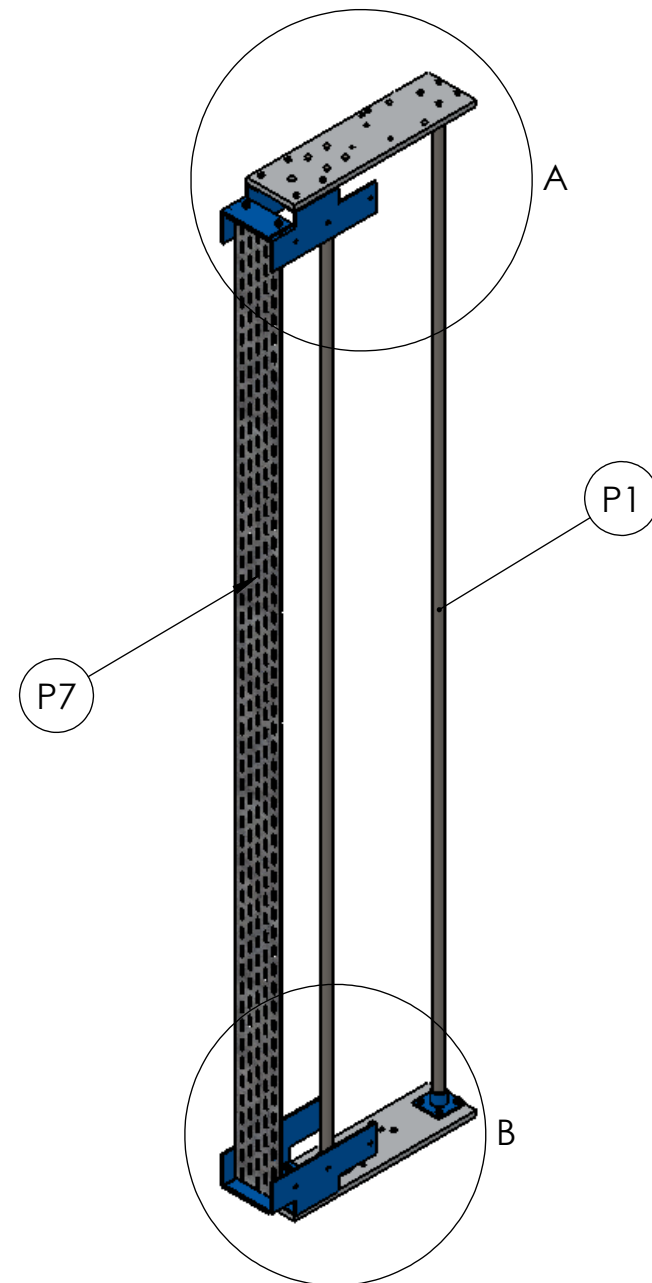
REF	TIPO	NOMBRE
EV.1	ENSAMBLAJE	EJE VERTICAL
EH.1	ENSAMBLAJE	EJE HORIZONTAL SUPERIOR
EH.6	ENSAMBLAJE	EJE HORIZONTAL INFERIOR

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: POSICIONADOR COMPLETO - PARTES	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: POS.2	A3
AÑO: 2022	ESCALA: S/E	HOJA: 2/73	MATERIAL: -	



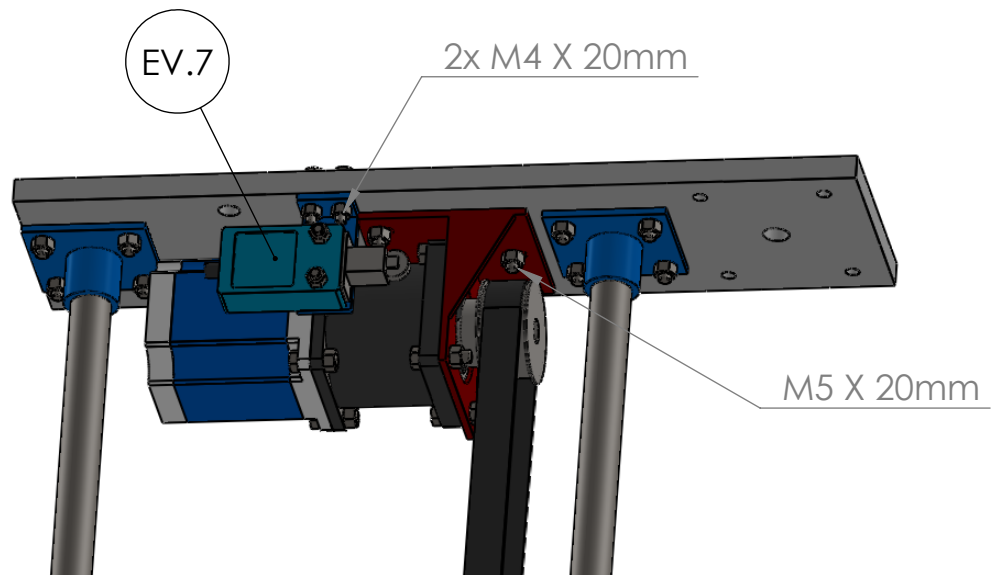
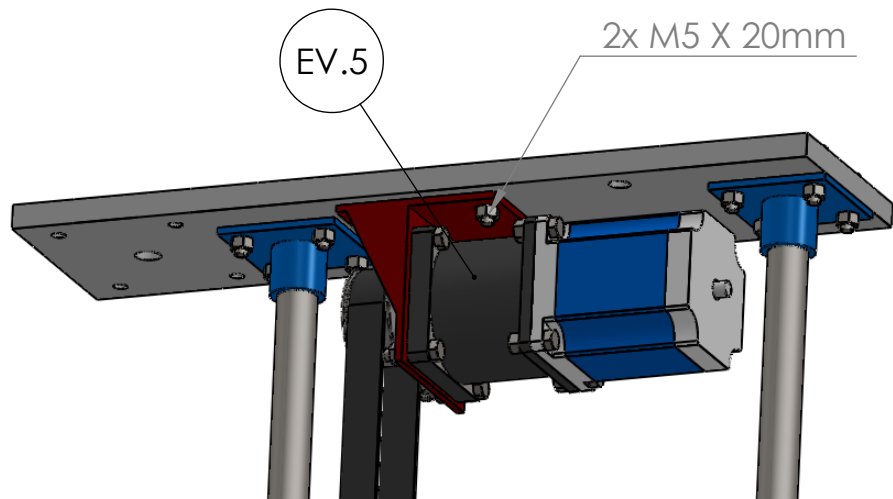
REF	TIPO	NOMBRE
EV.2	ENSAMBLAJE	EJE VERTICAL - CUERPO
EV.3	ENSAMBLAJE	EJE VERTICAL - EXTREMO SUPERIOR
EV.4	ENSAMBLAJE	EJE VERTICAL - EXTREMO INFERIOR
EV.8	ENSAMBLAJE	EJE VERTICAL - CHAPA DE PROTECCIÓN
PI	ENSAMBLAJE	PORTAINSTRUMENTOS

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE VERTICAL	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: EV.1	A3
AÑO: 2022	ESCALA: S/E	HOJA: 3/73	MATERIAL: -	



REF	TIPO	NOMBRE
P1	PIEZA	EJE VERTICAL GUIA
P2	PIEZA	SOPORTE VERTICAL SUPERIOR
P3	PIEZA	SOPORTE VERTICAL INFERIOR
P4	PIEZA	SOPORTE DE EJE VERTICAL GUIA
P5	PIEZA	SOPORTE DE CHAPA DE PROTECCIÓN VERTICAL 1
P6	PIEZA	SOPORTE DE CHAPA DE PROTECCIÓN VERTICAL 2
P7	PIEZA	GUIA DE CADENA PORTACABLES VERTICAL

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE VERTICAL - CUERPO	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: EV.2	A3
AÑO: 2022	ESCALA: S/E	HOJA: 4/73	MATERIAL: -	



REF	TIPO	NOMBRE
EV.5	ENSAMBLAJE	EJE VERTICAL - MOTOR
EV.7	ENSAMBLAJE	EJE VERTICAL - SENSOR DE POS. Y FINES DE CARRERA



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
5/73

NOMBRE:

EJE VERTICAL -
EXTREMO SUPERIOR

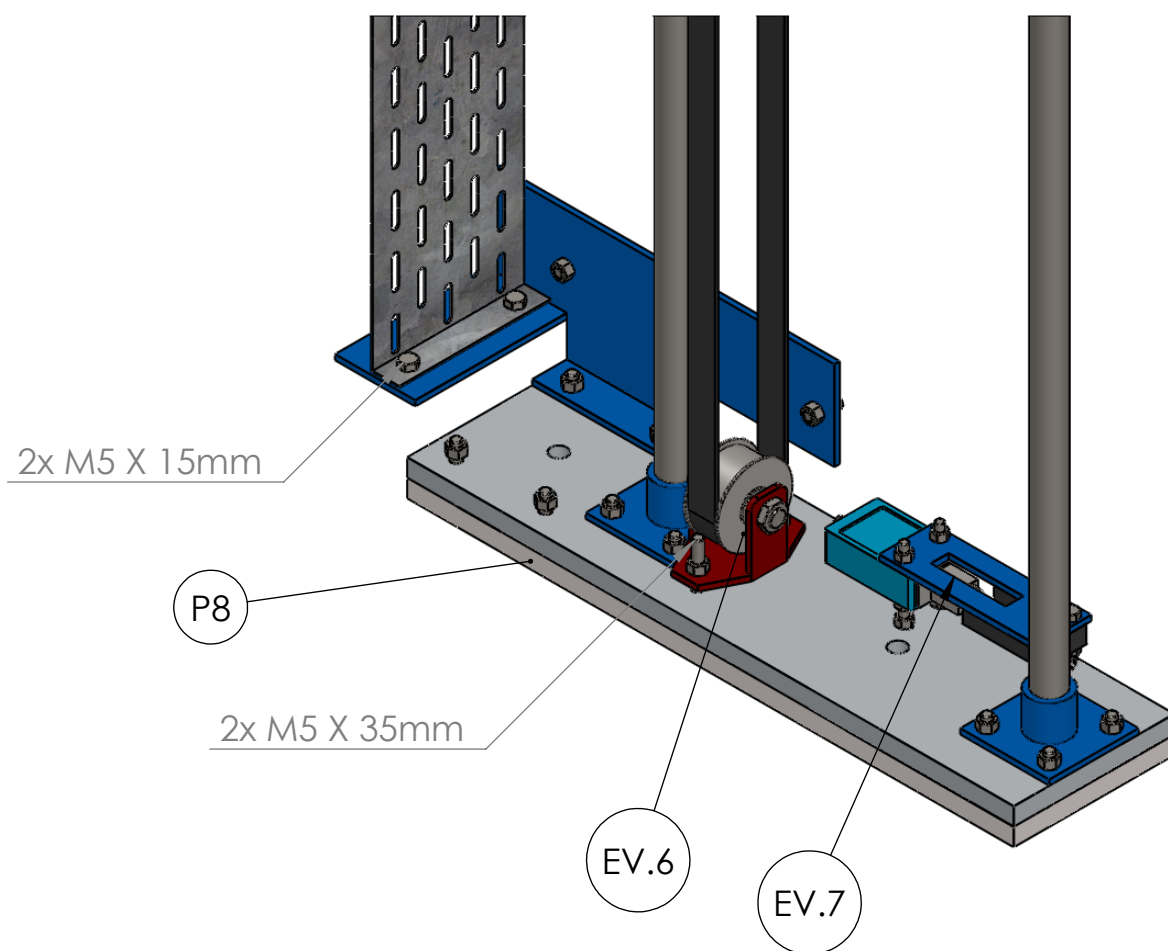
REFERENCIA:

EV.3

MATERIAL:

-

A4



REF	TIPO	NOMBRE
EV.6	ENSAMBLAJE	EJE VERTICAL - POLEA DE RETORNO
EV.7	ENSAMBLAJE	EJE VERTICAL - SENSOR DE POS. Y FINES DE CARRERA
P8	PIEZA	PLACA DE APOYO DEL EJE VERTICAL



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

HOJA: 6/73

NOMBRE:

EJE VERTICAL -
EXTREMO INFERIOR

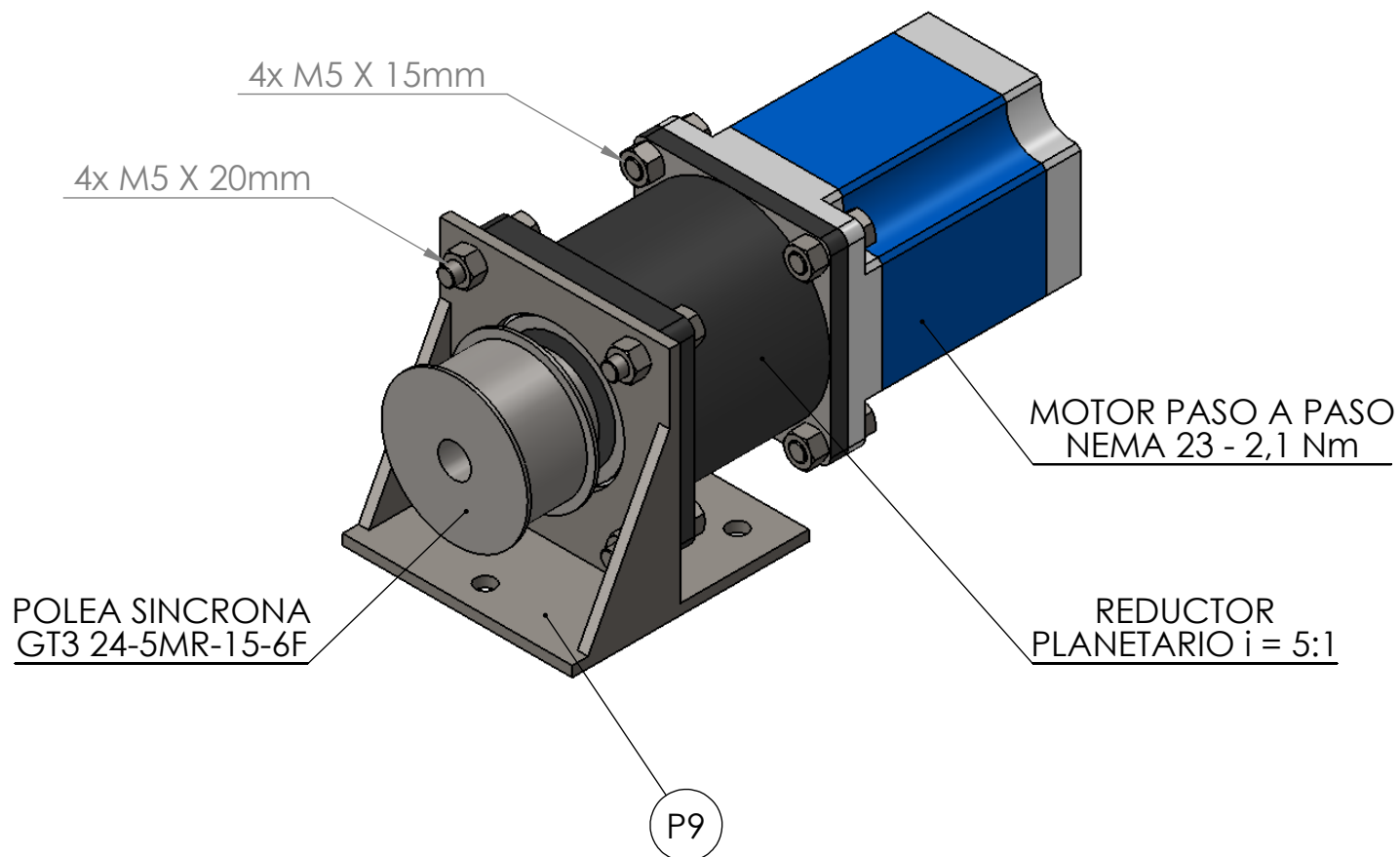
REFERENCIA:

EV.4

MATERIAL:

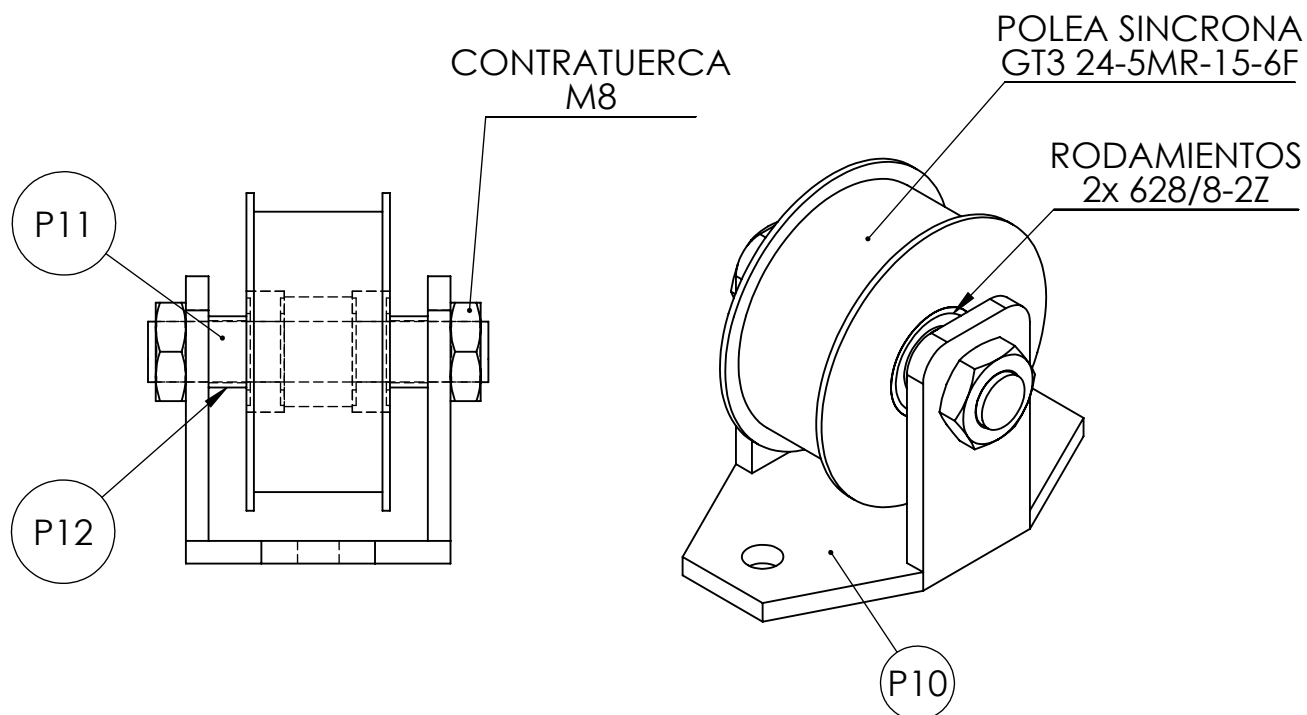
-

A4

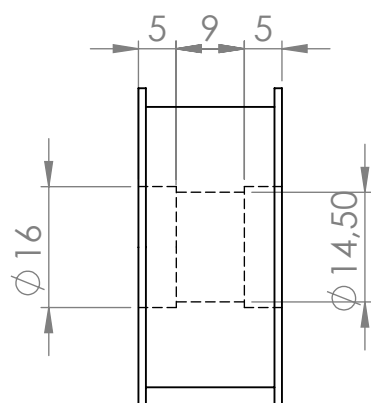


REF	TIPO	NOMBRE
P9	PIEZA	SOPORTE MOTOR VERTICAL

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE VERTICAL - MOTOR	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: EV.5	A4
AÑO: 2022	ESCALA: S/E	HOJA: 7/73	MATERIAL: -	

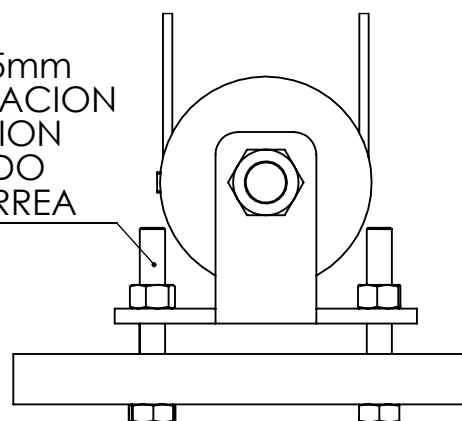


TALADRO DE LA POLEA



MONTAJE EN SOPORTE VERTICAL INFERIOR

2x M5 X 35mm
PARA REGULACION
DE POSICION
Y TENSADO
DE LA CORREA



TODAS LAS DIMENSIONES
EN mm

REF	TIPO	NOMBRE
P10	PIEZA	SOPORTE POLEA DE RETORNO VERTICAL
P11	PIEZA	EJE POLEA DE RETORNO VERTICAL
P12	PIEZA	CASQUILLO POLEA DE RETORNO VERTICAL



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:

OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
8/73

NOMBRE:

EJE VERTICAL -
POLEA DE RETORNO

REFERENCIA:

EV.6

MATERIAL:

-

A4

FIN DE CARRERA
ACUADOR: EMBOLO
DE ROLDANA
ABB LS21M14D11-P01

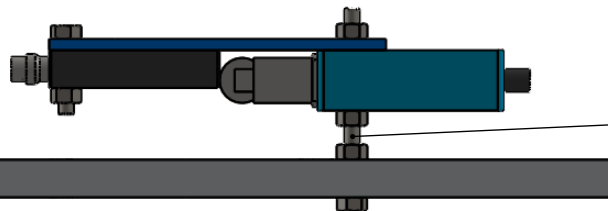
SENSOR DE POSICION
BARRERA FOTOELECTRICA
IFM OPU200

2x M4 X 20mm

P13

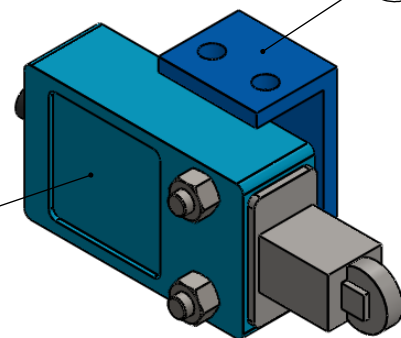
MONTAJE EN SOPORTE
VERTICAL INFERIOR

2x M4 X 50mm
REGULACION DE
POSICION MEDIANTE
CONTRATUERCAS



FIN DE CARRERA
ACUADOR: EMBOLO
DE ROLDANA
ABB LS21M12D11-P01

P14



REF	TIPO	NOMBRE
P13	PIEZA	SOPORTE DE SENSOR DE POSICION Y FIN DE CARRERA
P14	PIEZA	SOPORTE DE FIN DE CARRERA



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:

OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
9/73

NOMBRE:

EJE VERTICAL - SENSOR DE
POS. Y FINES DE CARRERA

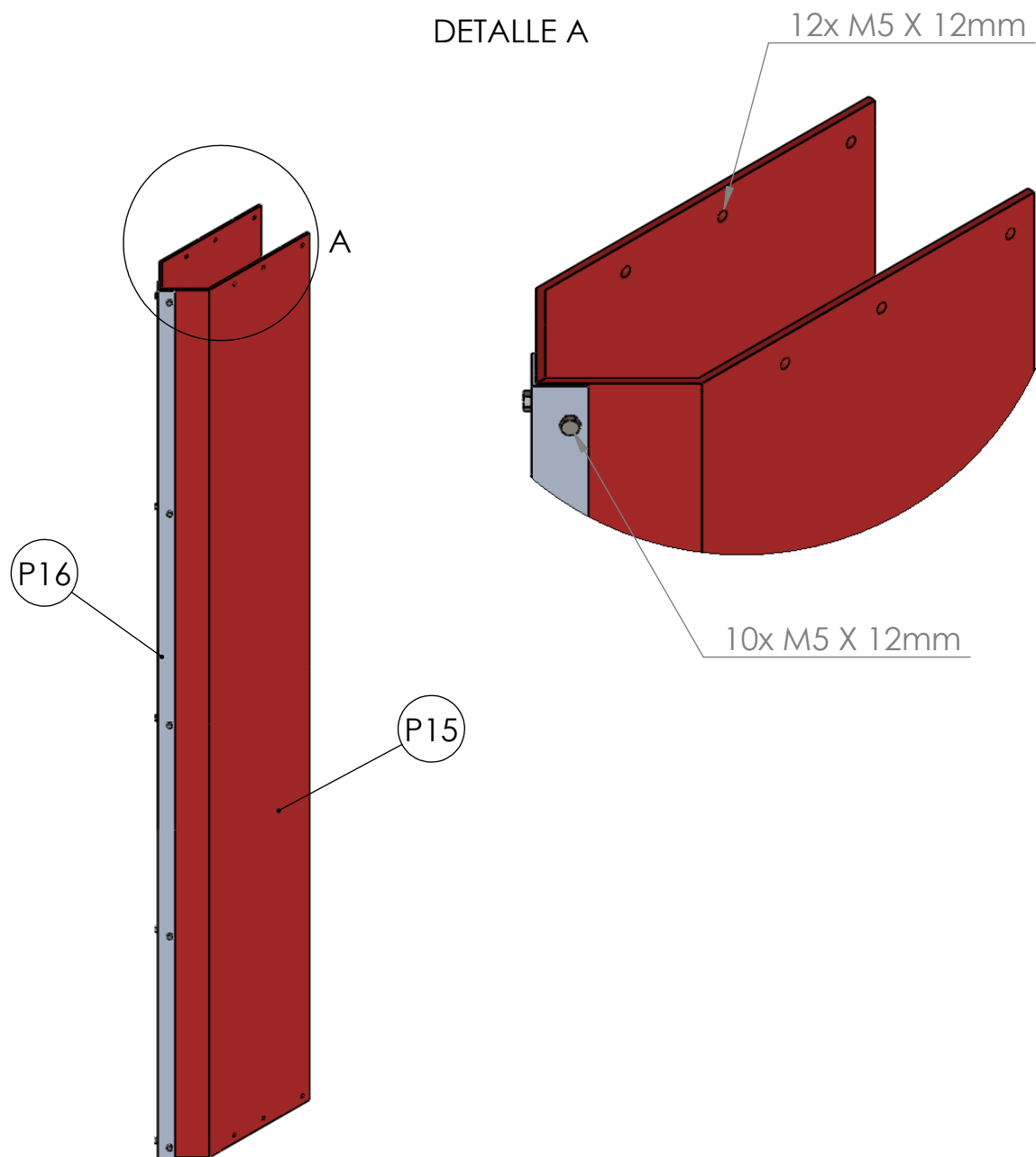
REFERENCIA:

EV.7

MATERIAL:

-

A4



REF	TIPO	NOMBRE
P15	PIEZA	CHAPA DE PROTECCION VERTICAL - MITAD
P16	PIEZA	PERFIL FRONTAL



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
10/73

NOMBRE:

EJE VERTICAL -
CHAPA DE PROTECCIÓN

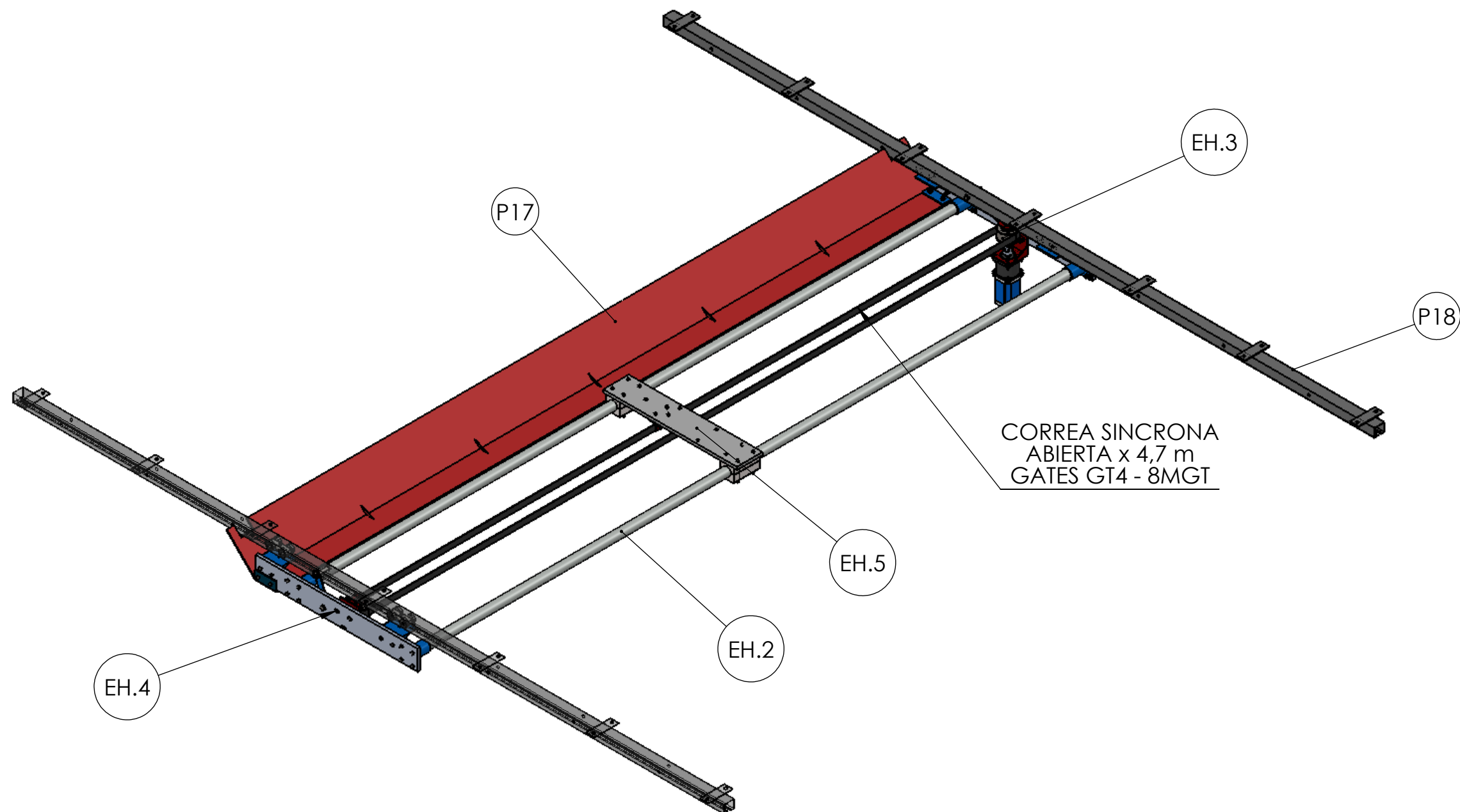
REFERENCIA:

EV.8

MATERIAL:

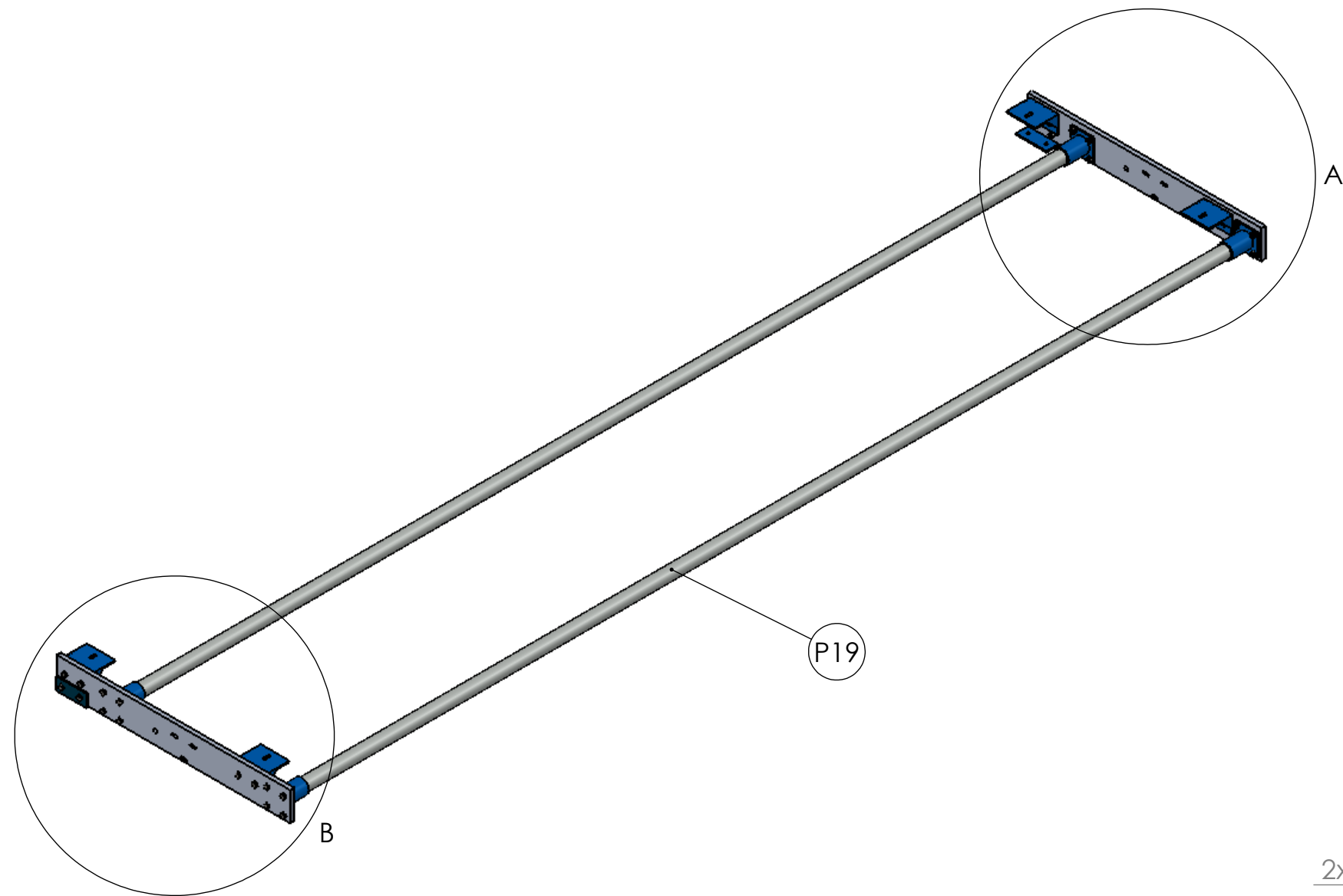
-

A4

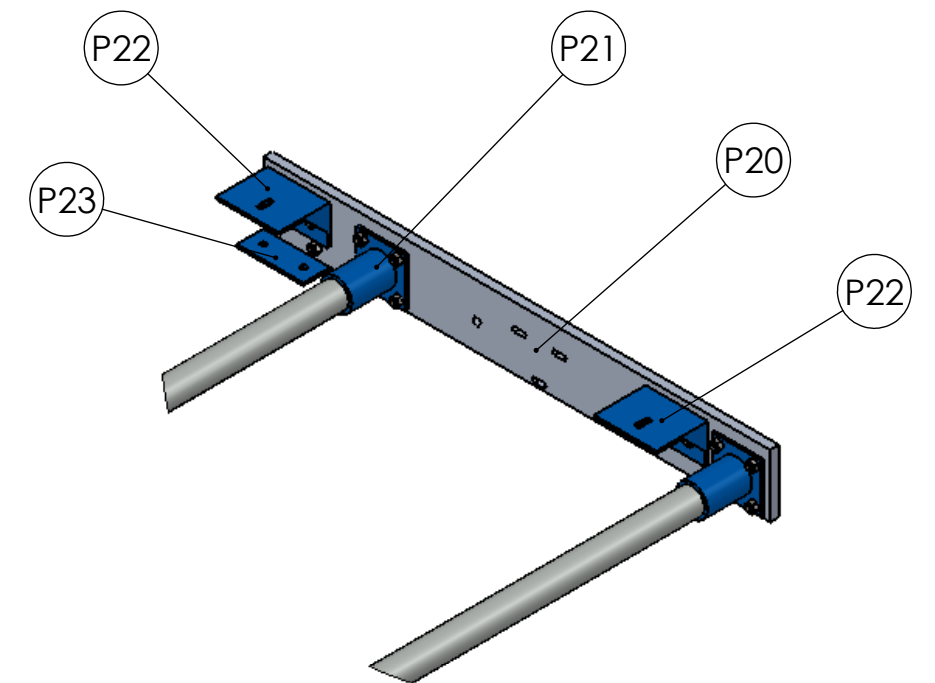


REF	TIPO	NOMBRE
EH.2	ENSAMBLAJE	EJE HOR. SUPERIOR - CUERPO
EH.3	ENSAMBLAJE	EJE HOR. SUPERIOR - EXTREMO IZQUIERDO
EH.4	ENSAMBLAJE	EJE HOR. SUPERIOR - EXTREMO DERECHO
EH.5	ENSAMBLAJE	EJE HOR. SUPERIOR - SOPORTE CENTRAL
P17	PIEZA	CHAPA DE PROTECCIÓN HORIZONTAL SUPERIOR
P18	PIEZA	RIEL PLANO

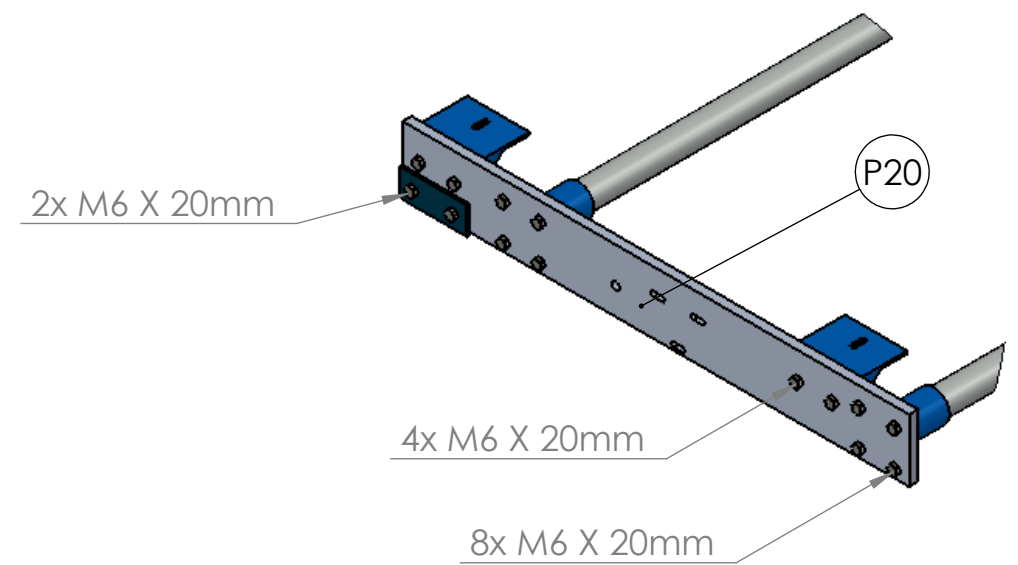
 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE HORIZONTAL SUPERIOR	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: EH.1	A3
AÑO: 2022	ESCALA: S/E	HOJA: 11/73	MATERIAL: -	



DETALLE A

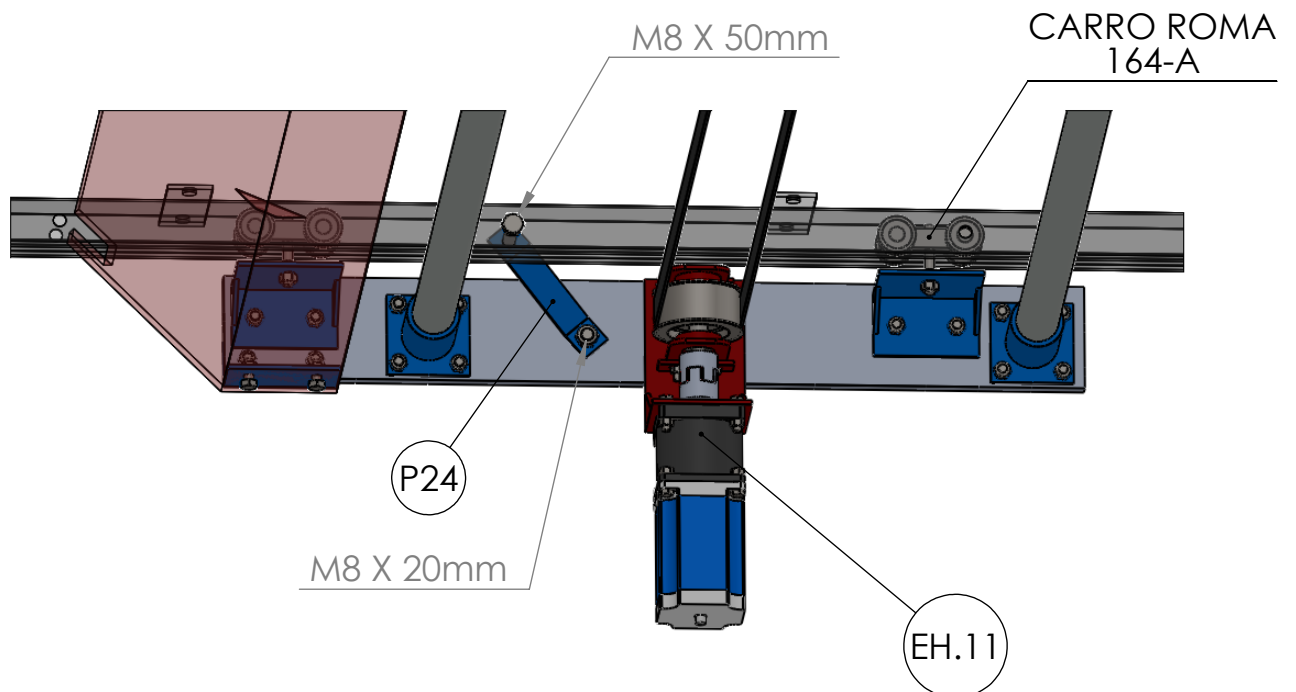
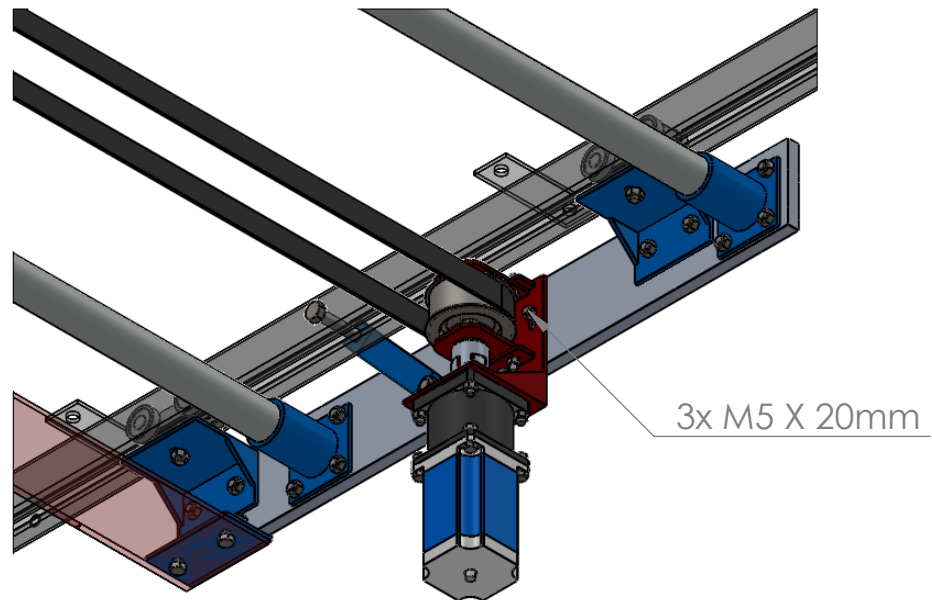


DETALLE B



REF	TIPO	NOMBRE
P19	PIEZA	EJE HORIZONTAL GUIA
P20	PIEZA	SOPORTE HORIZONTAL EJE SUPERIOR
P21	PIEZA	SOPORTE DE EJE HORIZONTAL GUIA
P22	PIEZA	SOPORTE CARRO SUPERIOR
P23	PIEZA	SOPORTE DE CHAPA DE PROTECCIÓN HOR. SUPERIOR

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE HOR. SUPERIOR - CUERPO	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: EH.2	A3
AÑO: 2022	ESCALA: S/E	HOJA: 12/73	MATERIAL: -	



REF	TIPO	NOMBRE
EH.11	ENSAMBLAJE	EJE HORIZONTAL - MOTOR
P24	PIEZA	TRABA AXIAL EJE HOR. SUPERIOR



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

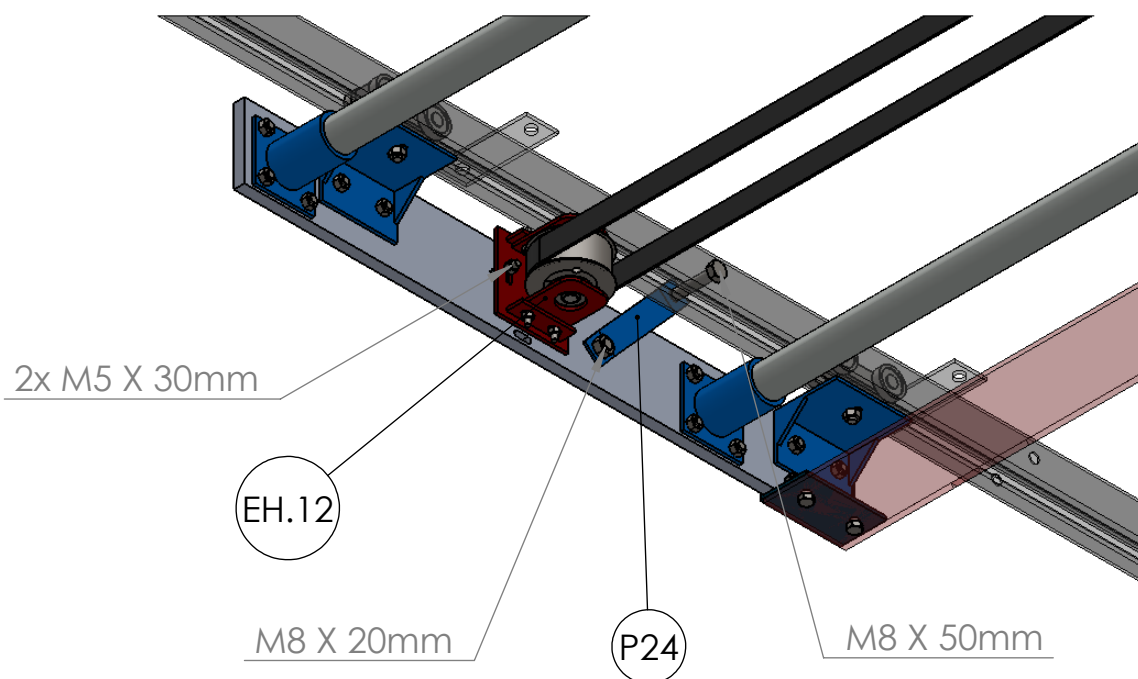
HOJA:
13/73

NOMBRE:
EJE HOR. SUPERIOR -
EXTREMO IZQUIERDO

REFERENCIA:
EH.3

MATERIAL:
-

A4



REF	TIPO	NOMBRE
EH.12	ENSAMBLAJE	EJE HORIZONTAL - POLEA DE RETORNO
P24	PIEZA	TRABA AXIAL EJE HOR. SUPERIOR



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

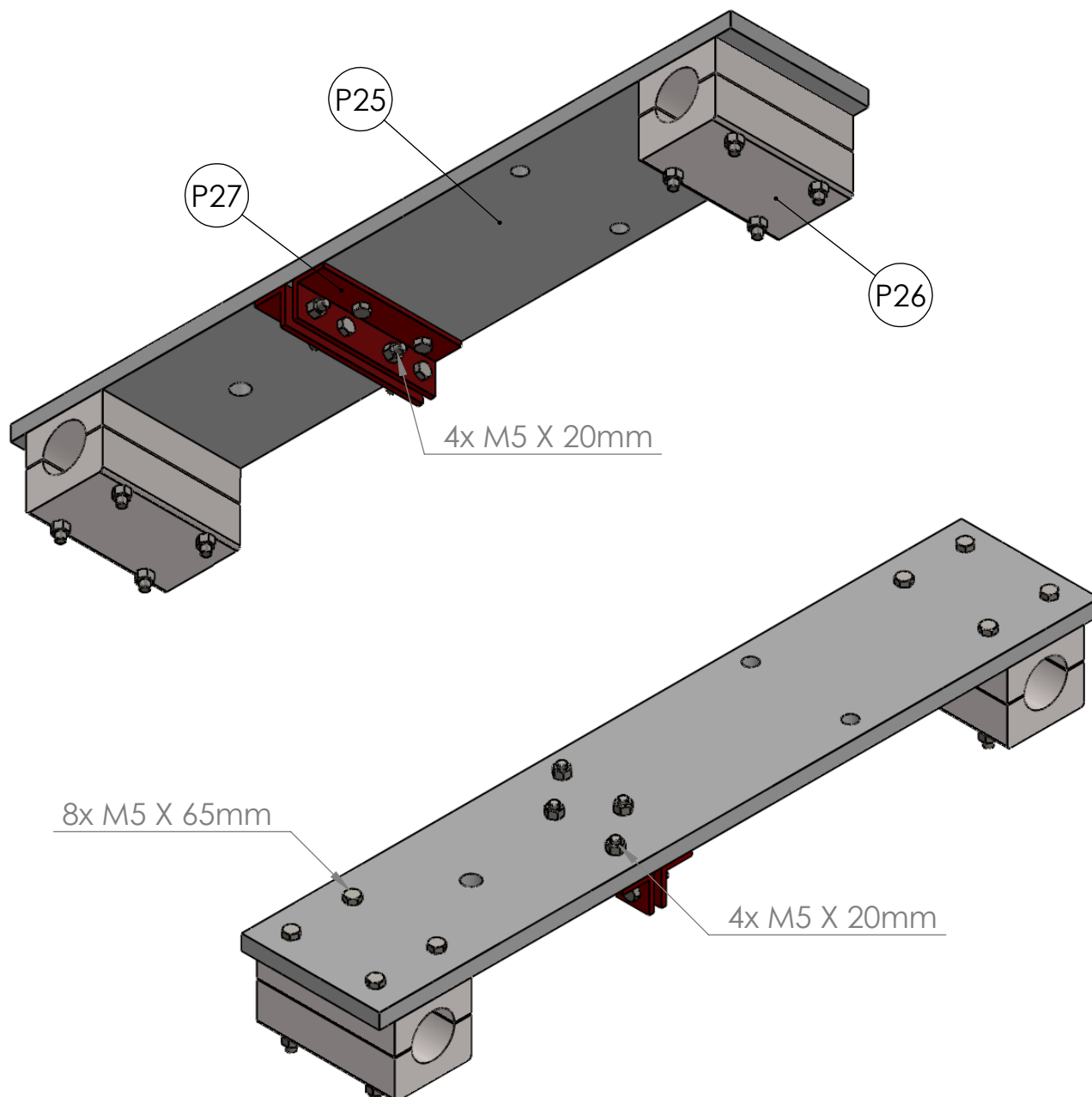
HOJA:
14/73

NOMBRE:
EJE HOR. SUPERIOR -
EXTREMO DERECHO

REFERENCIA:
EH.4

MATERIAL:
-

A4



REF	TIPO	NOMBRE
P25	PIEZA	SOPORTE CENTRAL EJE HOR. SUPERIOR
P26	PIEZA	BUJES PARA EJES HORIZONTALES
P27	PIEZA	SUJECION DE CORREA HORIZONTAL



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

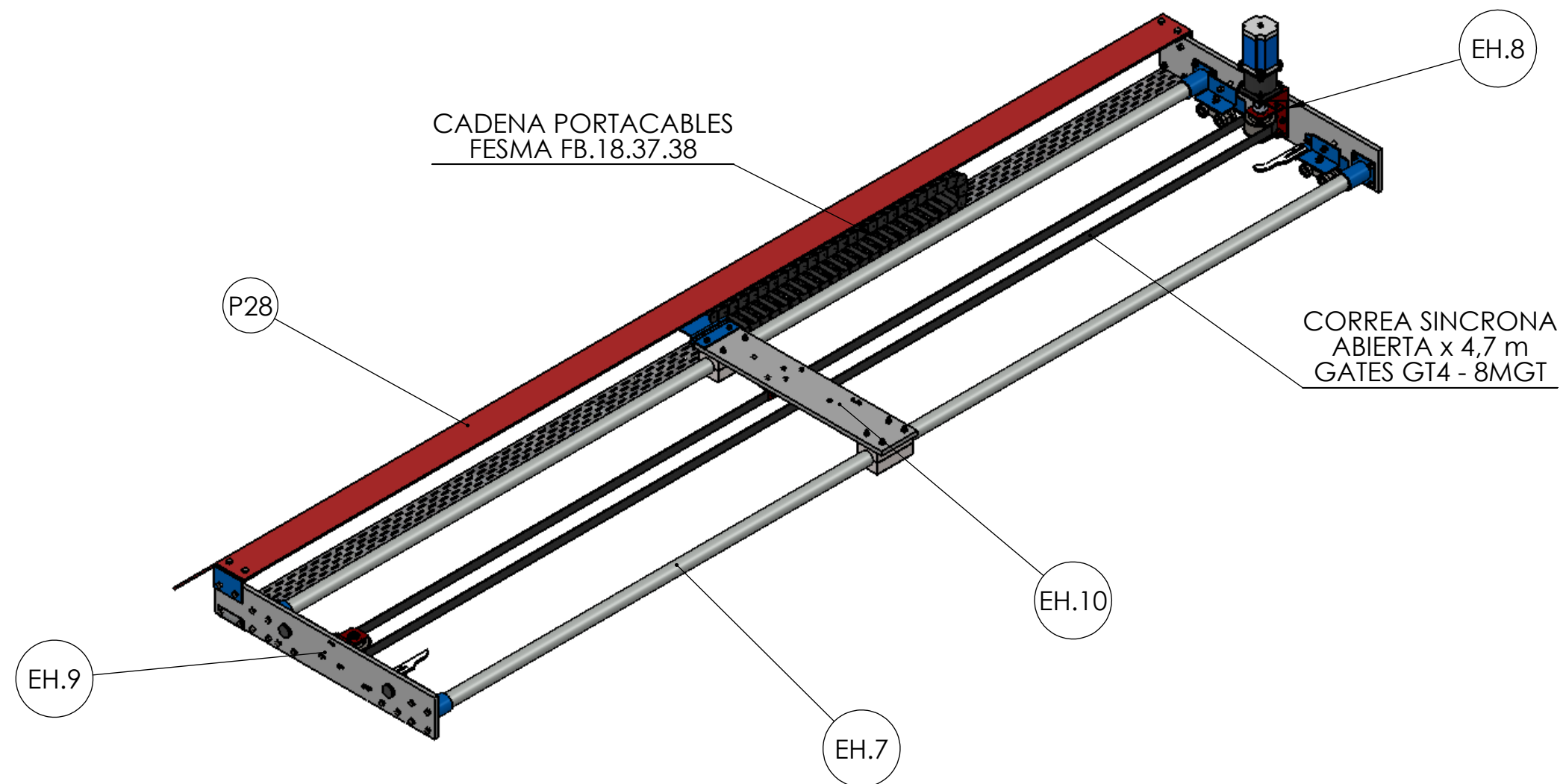
HOJA:
15/73

NOMBRE:
EJE HOR. SUPERIOR -
SOPORTE CENTRAL

REFERENCIA:
EH.5

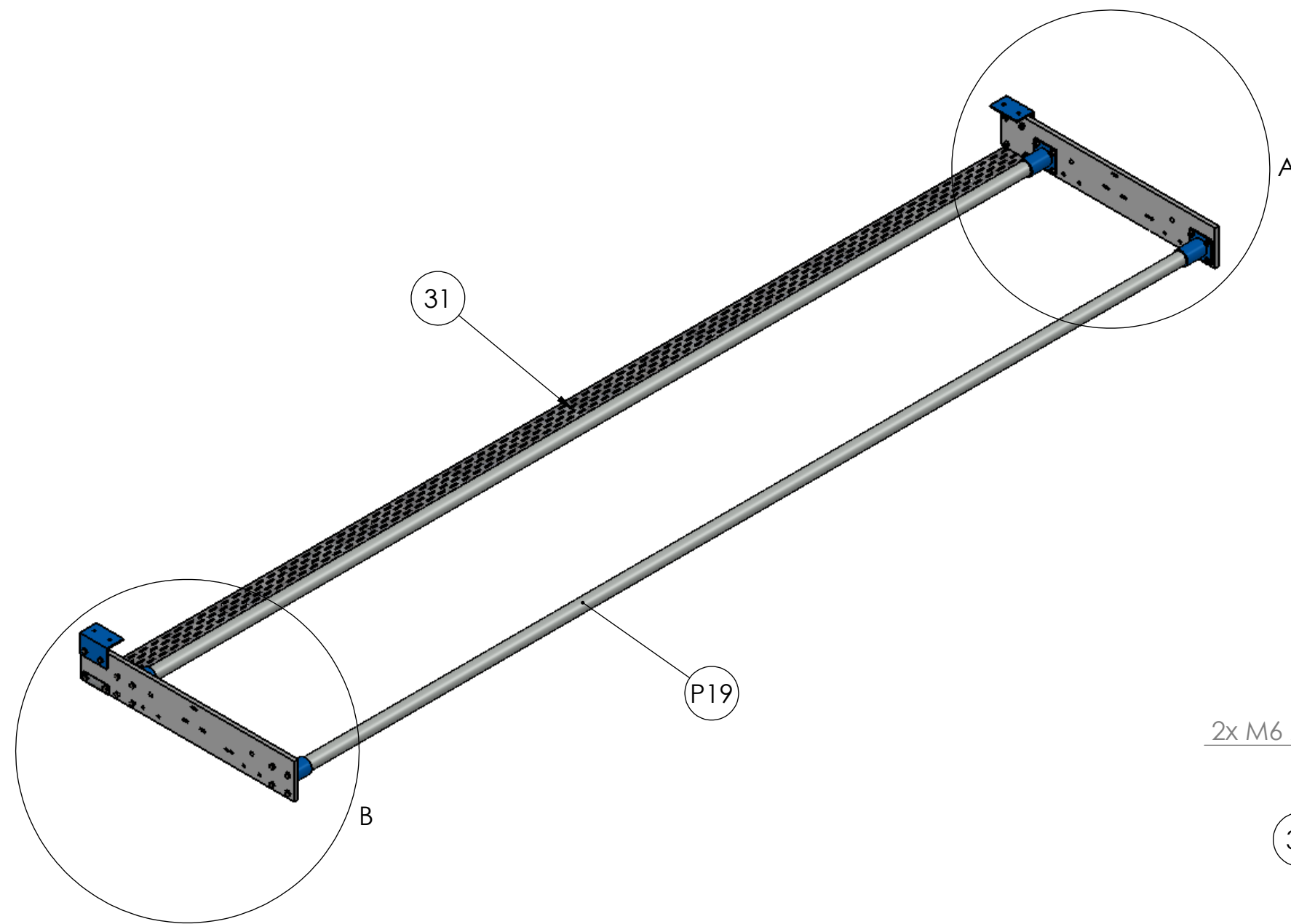
MATERIAL:
-

A4

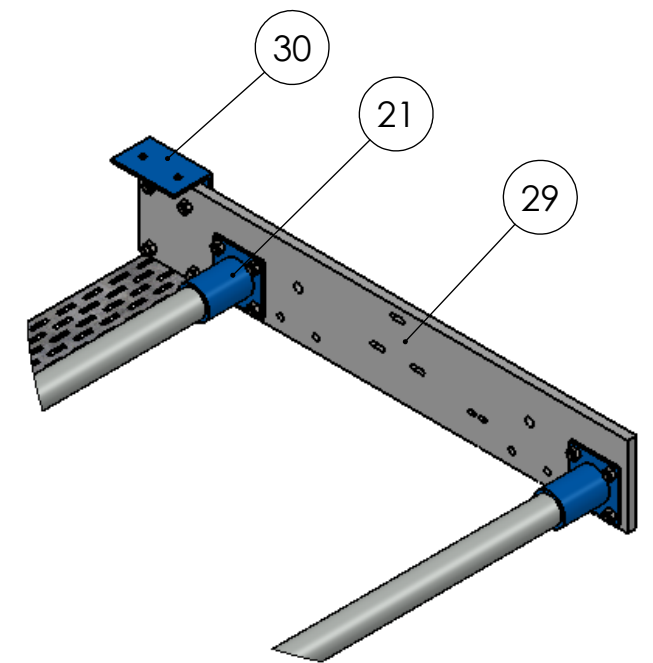


REF	TIPO	NOMBRE
EH.7	ENSAMBLAJE	EJE HOR. INFERIOR - CUERPO
EH.8	ENSAMBLAJE	EJE HOR. INFERIOR - EXTREMO IZQUIERDO
EH.9	ENSAMBLAJE	EJE HOR. INFERIOR - EXTREMO DERECHO
EH.10	ENSAMBLAJE	EJE HOR. INFERIOR - SOPORTE CENTRAL
P28	PIEZA	CHAPA DE PROTECCIÓN HORIZONTAL INFERIOR

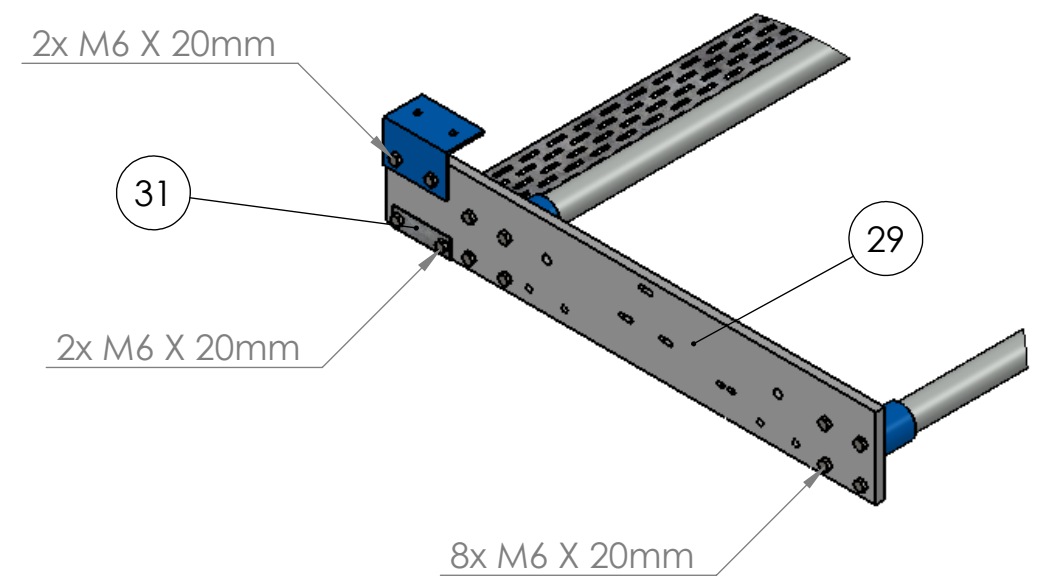
 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE HORIZONTAL INFERIOR	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: EH.6	A3
AÑO: 2022	ESCALA: S/E	HOJA: 16/73	MATERIAL: -	



DETALLE A

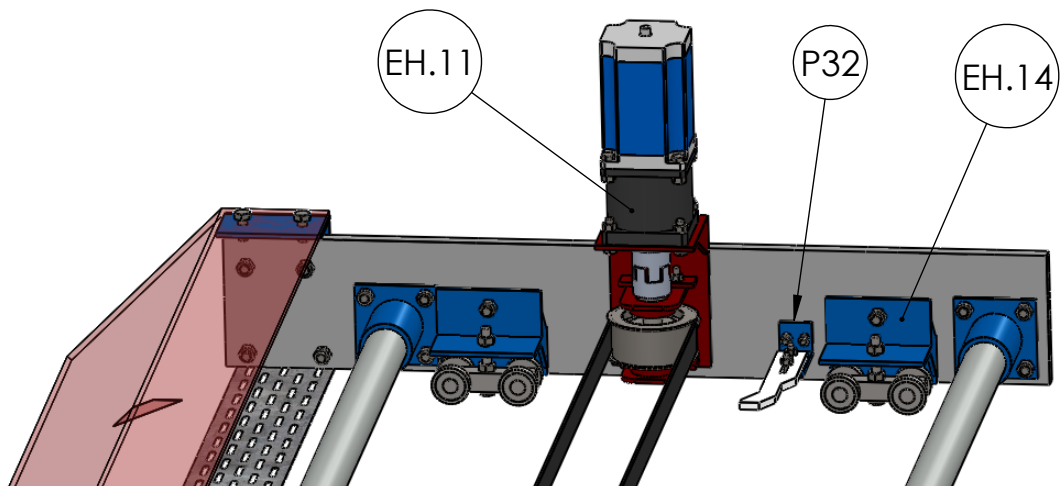


DETALLE B

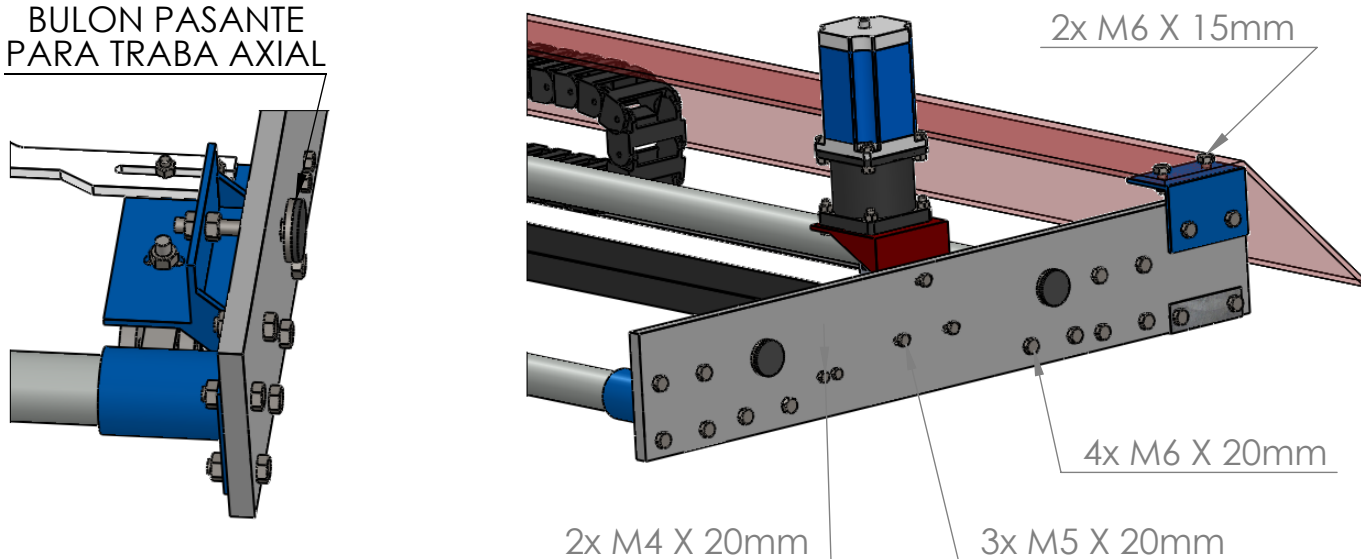


REF	TIPO	NOMBRE
P19	PIEZA	EJE HORIZONTAL GUIA
P21	PIEZA	SOPORTE DE EJE HORIZONTAL GUIA
P29	PIEZA	SOPORTE HORIZONTAL EJE INFERIOR
P30	PIEZA	SOPORTE DE CHAPA DE PROTECCIÓN HOR. INFERIOR
P31	PIEZA	SOPORTE DE CADENA PORTACABLES HORIZONTAL

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE HOR. INFERIOR - CUERPO	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: EH.7	A3
AÑO: 2022	ESCALA: S/E	HOJA: 17/73	MATERIAL: -	



BULON PASANTE
PARA TRABA AXIAL



REF	TIPO	NOMBRE
EH.11	ENSAMBLAJE	EJE HORIZONTAL - MOTOR
EH.14	ENSAMBLAJE	CARRO INFERIOR
P32	PIEZA	ACCIONAMIENTO DE SENSOR Y FIN DE CARRERA



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
18/68

NOMBRE:

EJE HOR. INFERIOR -
EXTREMO IZQUIERDO

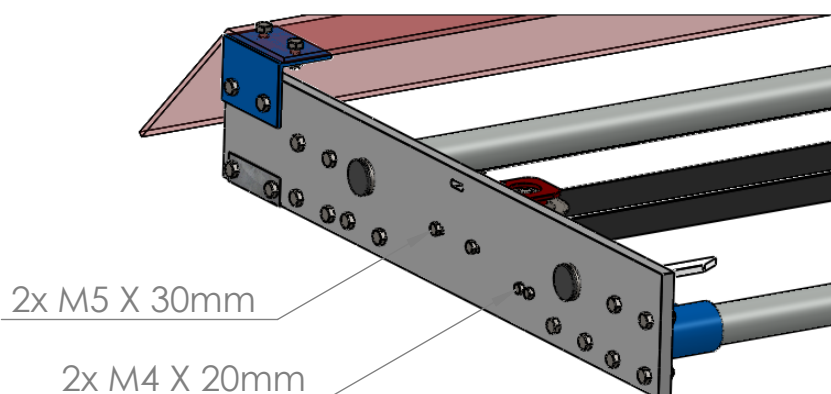
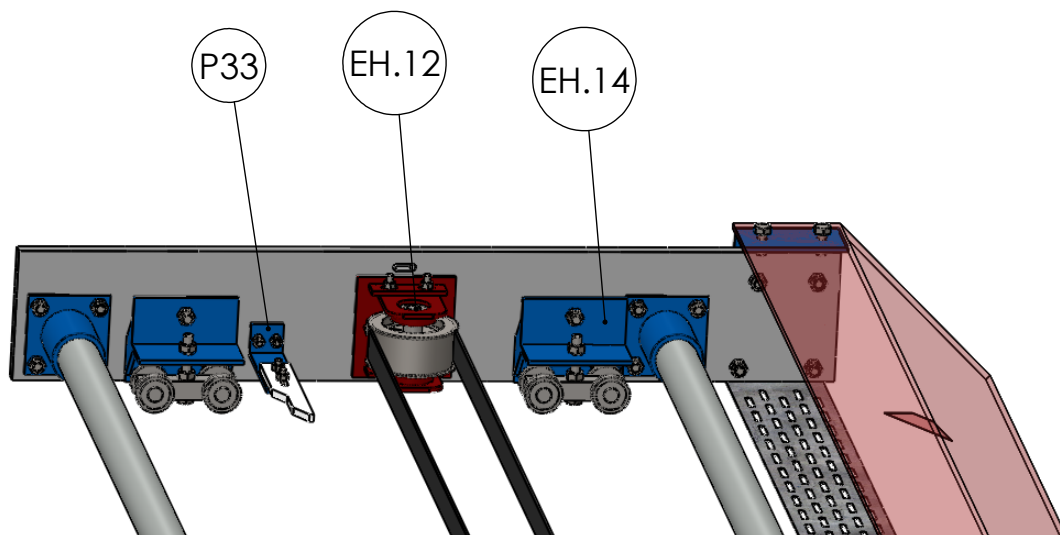
REFERENCIA:

EH.8

MATERIAL:

-

A4



REF	TIPO	NOMBRE
EH.12	ENSAMBLAJE	EJE HORIZONTAL - POLEA DE RETORNO
EH.14	ENSAMBLAJE	CARRO INFERIOR
P33	PIEZA	ACCIONAMIENTO DE FIN DE CARRERA



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

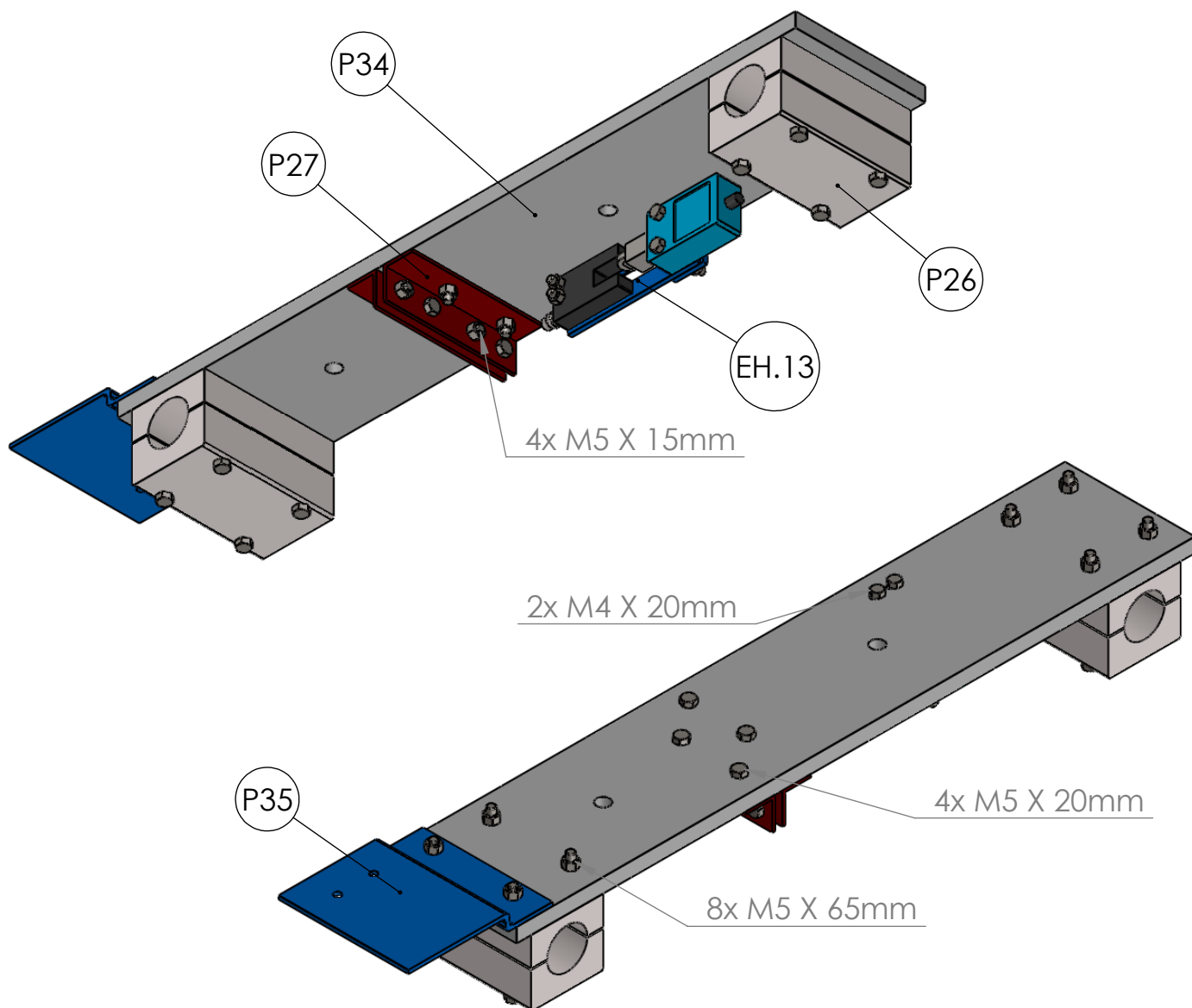
HOJA:
19/73

NOMBRE: EJE HOR. INFERIOR -
EXTREMO DERECHO

REFERENCIA: EH.9

MATERIAL: -

A4



REF	TIPO	NOMBRE
EH.13	ENSAMBLAJE	EJE HORIZONTAL - SENSOR DE POS. Y FIN DE CARR.
P26	PIEZA	BUJES PARA EJES HORIZONTALES
P27	PIEZA	SUJESION DE CORREA HORIZONTAL
P34	PIEZA	SOPORTE CENTRAL EJE HOR. INFERIOR
P35	PIEZA	SUJECIÓN CADENA PORTACABLES HORIZONTAL



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

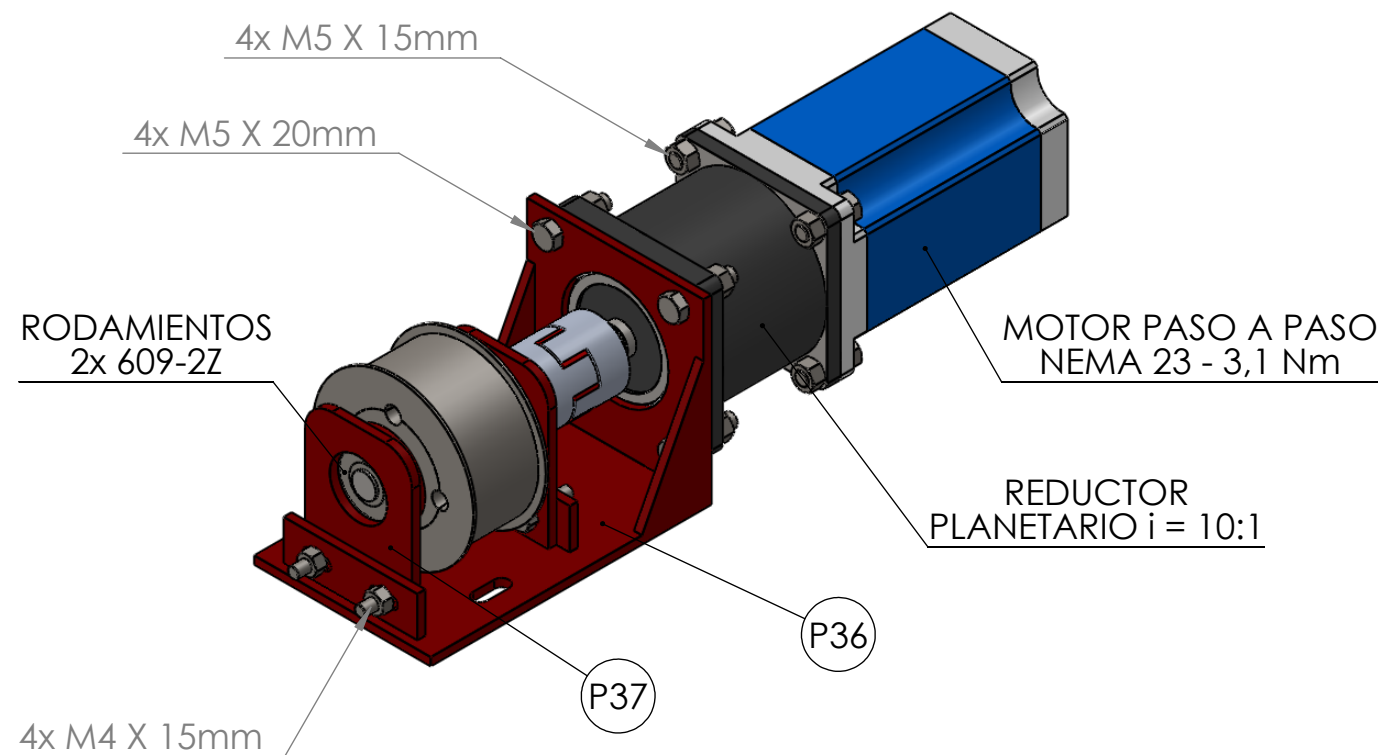
HOJA:
20/73

NOMBRE:
EJE HOR. INFERIOR -
SOPORTE CENTRAL

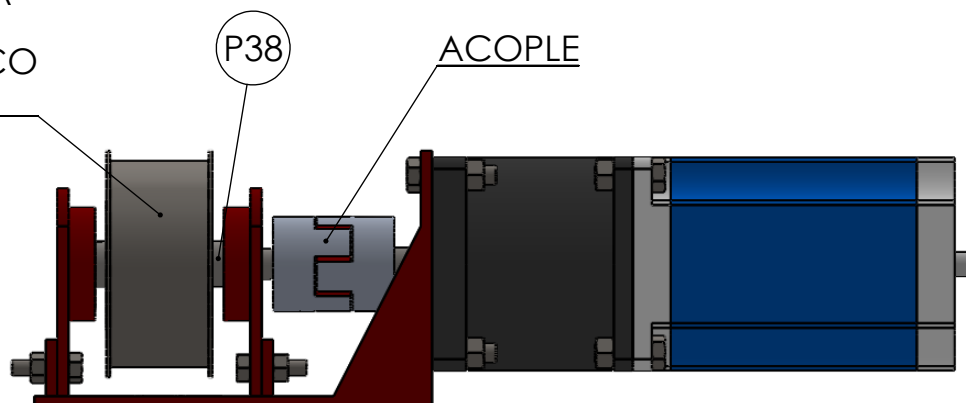
REFERENCIA:
EH.10

MATERIAL:
-

A4



POLEA SINCRONA
HTD 22-8M-20-5F
CASQUILLO CONICO
1008-12



REF	TIPO	NOMBRE
P36	PIEZA	SOPORTE MOTOR HORIZONTAL
P37	PIEZA	SOPORTE POLEA HORIZONTAL
P38	PIEZA	EJE POLEA MOTORA HORIZONTAL



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:

OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
21/73

NOMBRE:

EJE HORIZONTAL -
MOTOR

REFERENCIA:

EH.11

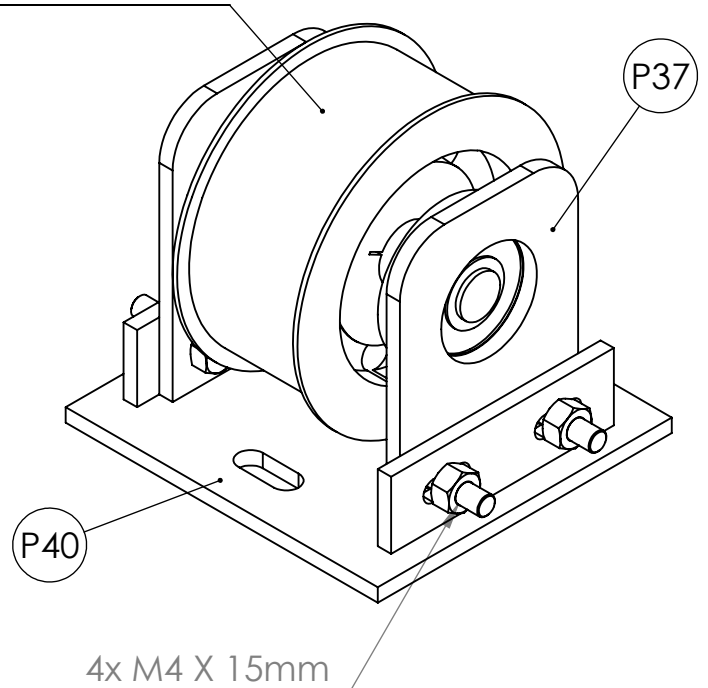
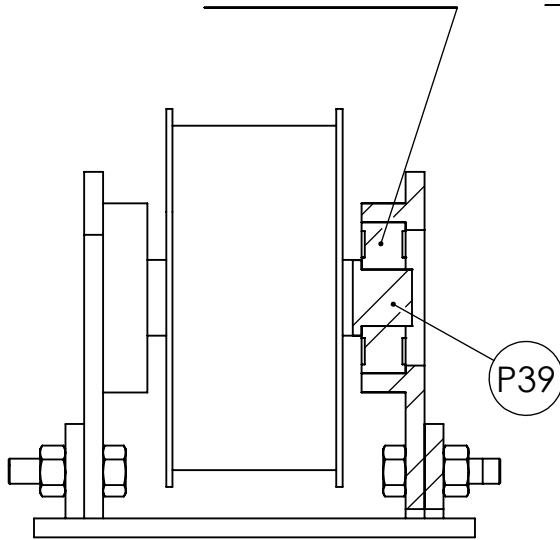
MATERIAL:

-

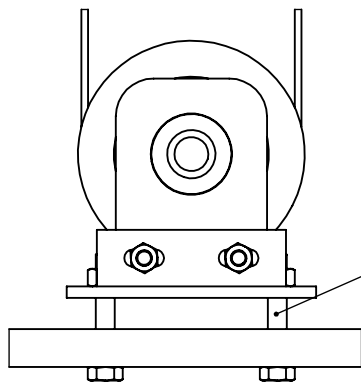
A4

RODAMIENTOS
2x 609-2Z

POLEA SINCRONA
HTD 22-8M-20-5F
CASQUILLO CONICO
1008-12



MONTAJE EN SOPORTE
HORIZONTAL DERECHO



2x M5 X 35mm
PARA REGULACION
DE POSICION
Y TENSADO
DE LA CORREA

REF	TIPO	NOMBRE
P37	PIEZA	SOPORTE POLEA HORIZONTAL
P39	PIEZA	EJE POLEA RETORNO HORIZONTAL
P40	PIEZA	SOPORTE POLEA DE RETORNO HORIZONTAL



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:

OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
22/73

NOMBRE:

EJE HORIZONTAL -
POLEA DE RETORNO

REFERENCIA:

EH.12

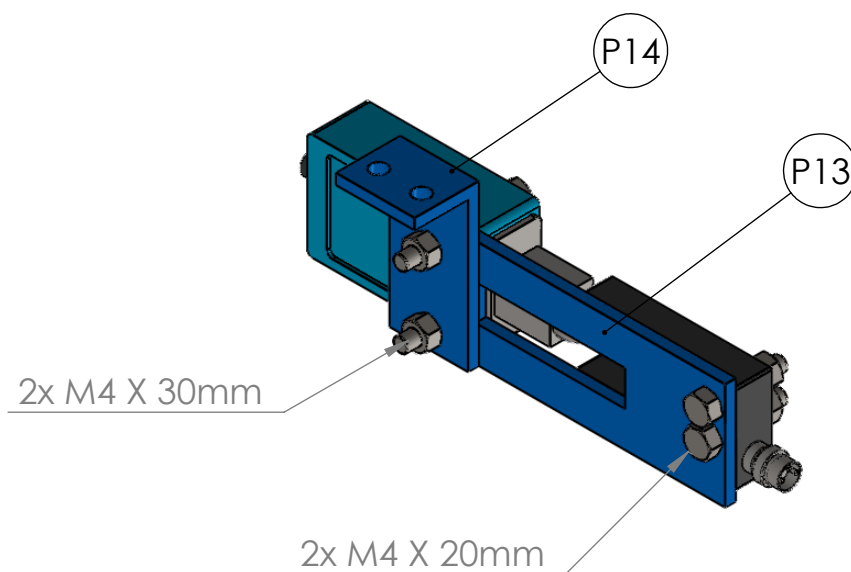
MATERIAL:

-

A4

SENSOR DE POSICION
BARRERA FOTOELECTRICA
IFM OPU200

FIN DE CARRA
ACUADOR: EMBOLO
DE ROLDANA
ABB LS21 M14D11-P01



REF	TIPO	NOMBRE
P13	PIEZA	SOPORTE DE SENSOR DE POSICIÓN Y FIN DE CARRERA
P14	PIEZA	SOPORTE DE FIN DE CARRERA



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
23/73

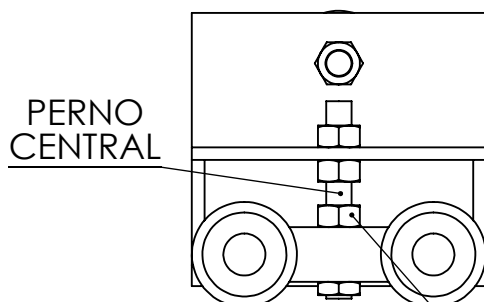
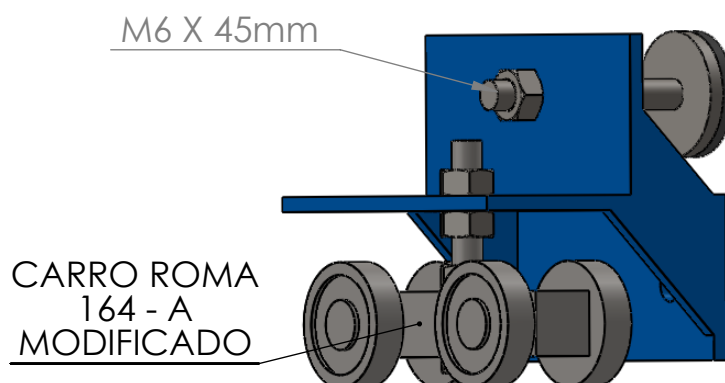
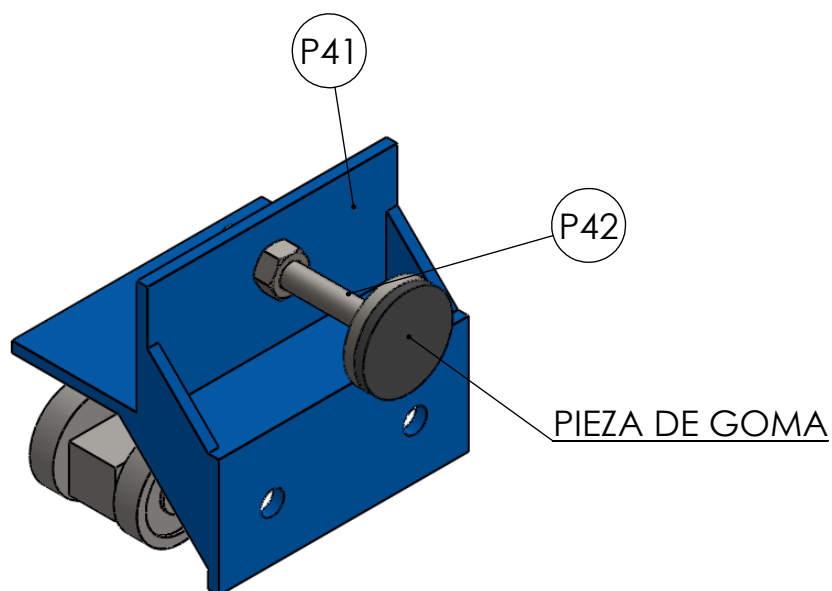
NOMBRE:
EJE HORIZONTAL - SENSOR DE
POS. Y FIN DE CARRERA

REFERENCIA:
EH.13

MATERIAL:

-

A4



ESTOS CARROS SON DISEÑADOS PARA TRABAJAR CON LA CARGA SUSPENDIDA. AQUI POR MONTARSE EN POSICION INVERTIDA DEBE REEMPLASARCE EL PERNO CENTRAL POR UN BOLON ROSCADO CON UNA TUERCA QUE HAGA DE APOYO TRANSFIRIENDO EL PESO DEL SOPORTE AL CARRO

TUERCA DE APOYO

REF	TIPO	NOMBRE
P41	PIEZA	SOPORTE CARRO INFERIOR
P42	PIEZA	TRABA AXIAL EJE HOR. INFERIOR



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

HOJA: 24/73

NOMBRE:

EJE HORIZONTAL -
CARRO INFERIOR

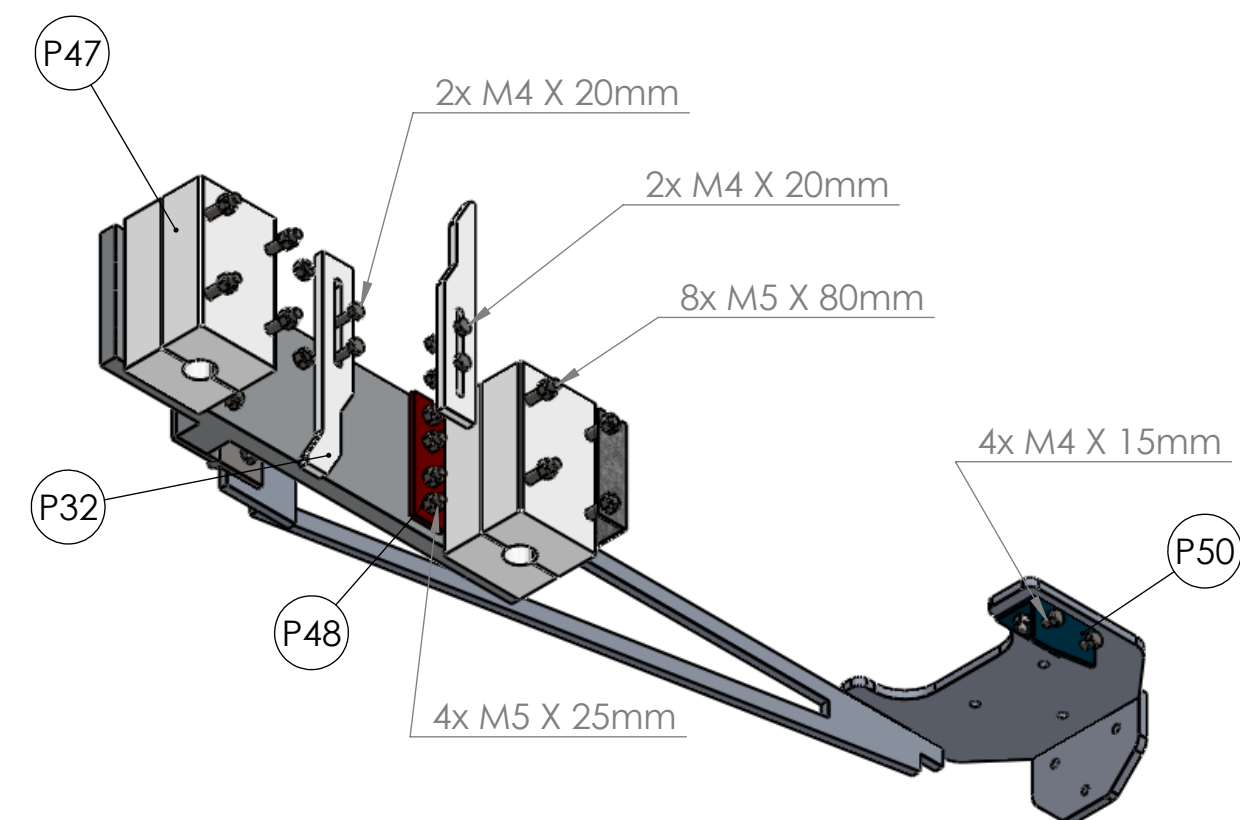
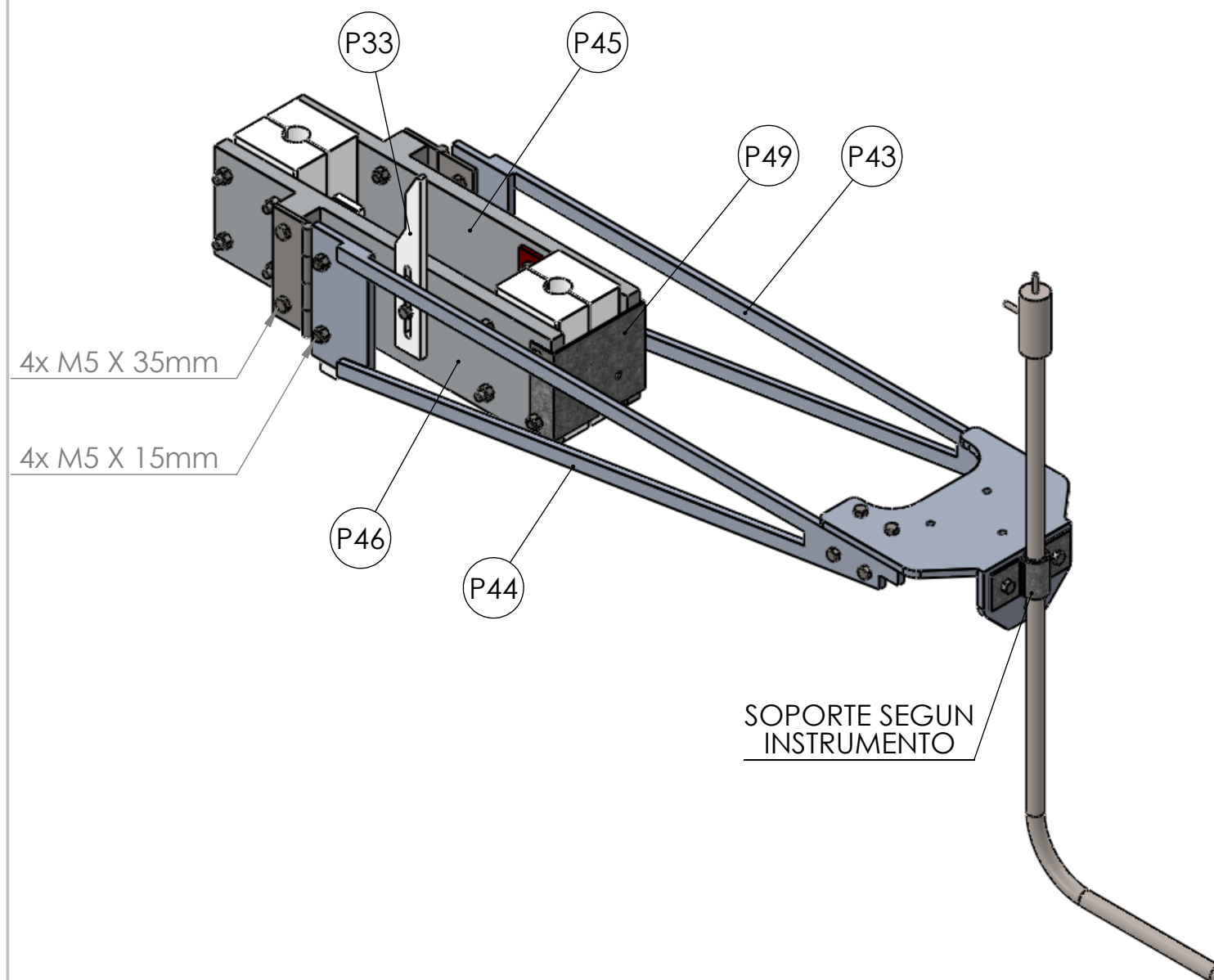
REFERENCIA:

EH.14

MATERIAL:

-

A4

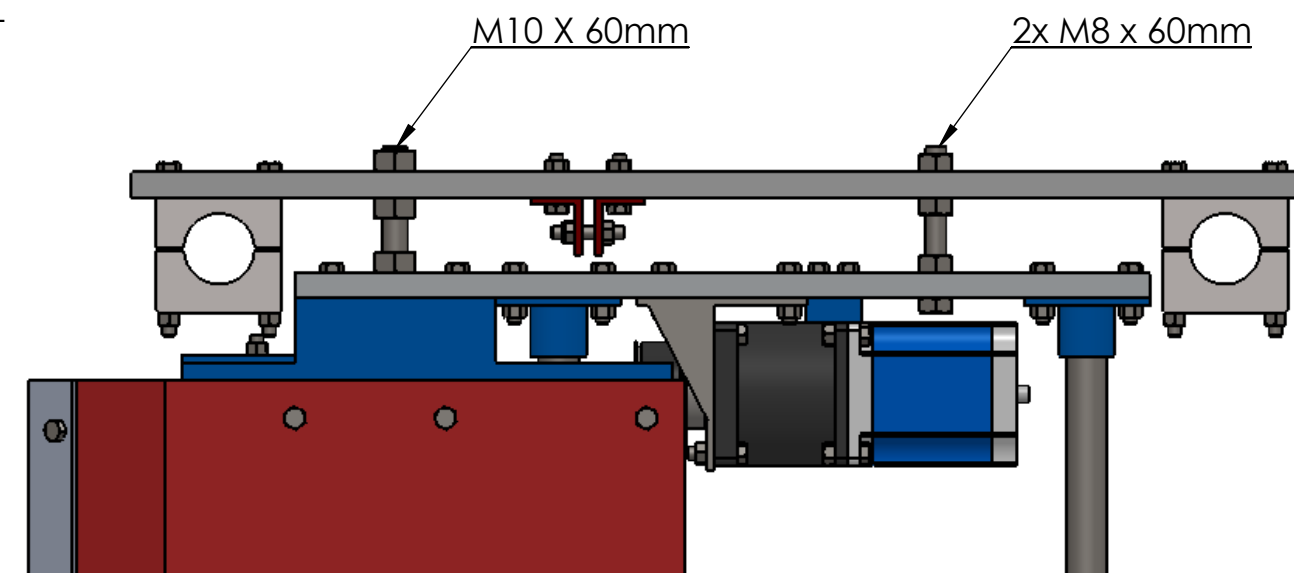


REF	TIPO	NOMBRE
P32	PIEZA	PLACA ACCIONAMIENTO VERT. INFERIOR
P33	PIEZA	PLACA ACCIONAMIENTO VERT. SUPERIOR
P43	PIEZA	POSTAINSTRUMENTO - BRAZO DERECHO
P44	PIEZA	PORTAINSTRUMENTO - BRAZO IZQUIERDO
P45	PIEZA	PORTAINSTRUMENTO - PLACA SOPORTE DERECHA
P46	PIEZA	PORTAINSTRUMENTO - PLACA SOPORTE IZQUIERDA
P47	PIEZA	BUJES PARA EJES VERTICALES
P48	PIEZA	SUJECIÓN DE CORREA VERTICAL
P49	PIEZA	SUJECIÓN DE CADENA PORTACABLES VERTICAL
P50	PIEZA	PORTAINSTRUMENTO - UNION ENTRE BRAZOS

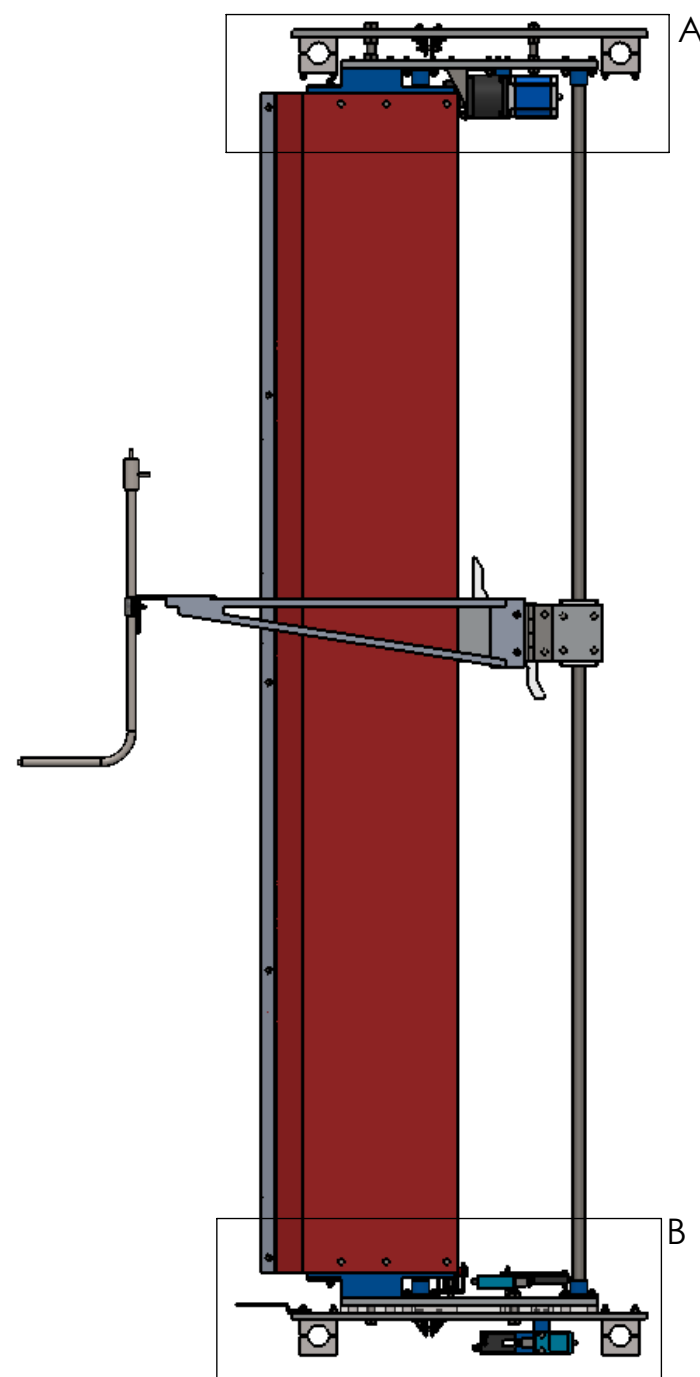
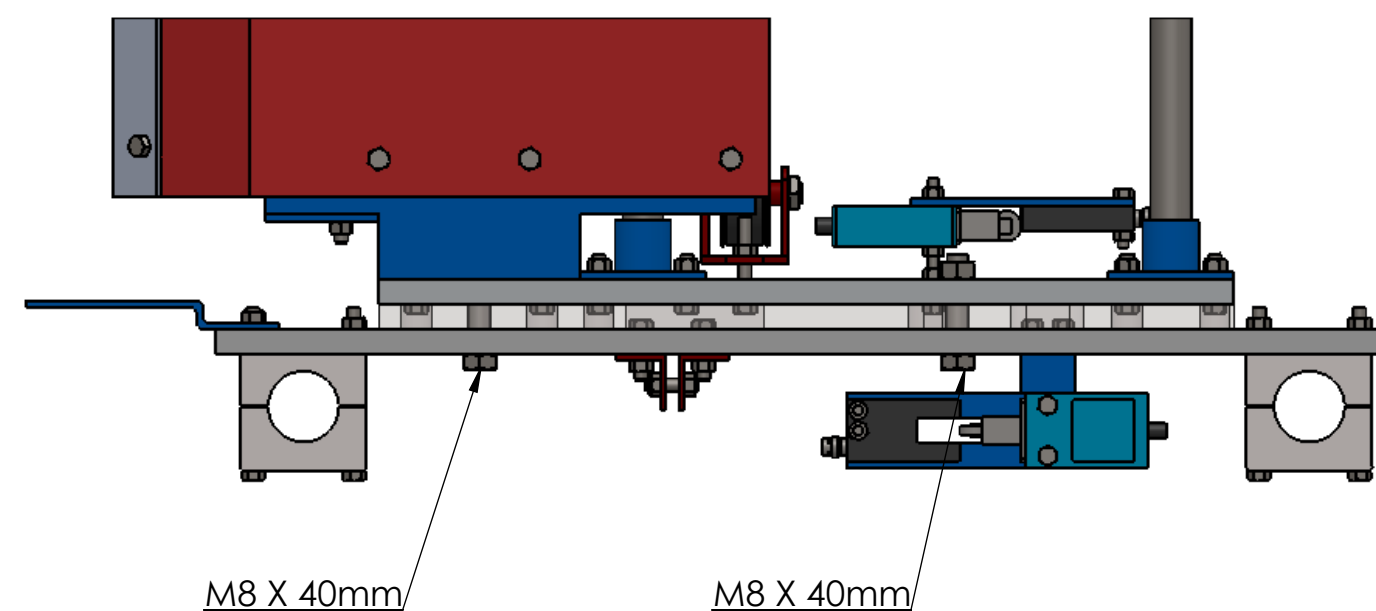
 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: PORTAINSTRUMENTOS	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: PI	A3
AÑO: 2022	ESCALA: S/E	HOJA: 25/73	MATERIAL: -	

REGULACION DE ALTURA DEL EJE VERTICAL

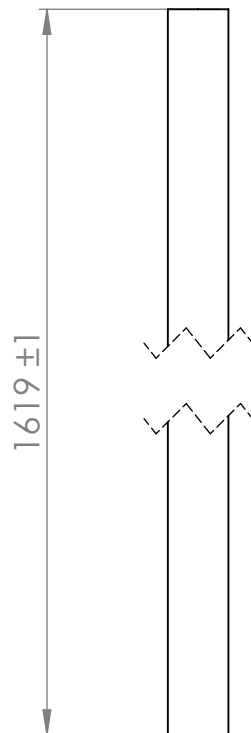
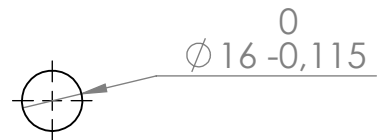
DETALLE A



DETALLE B



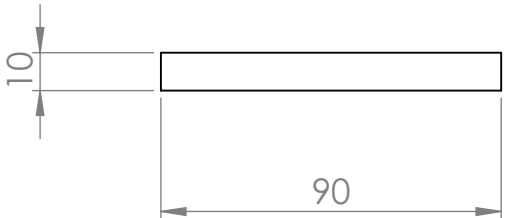
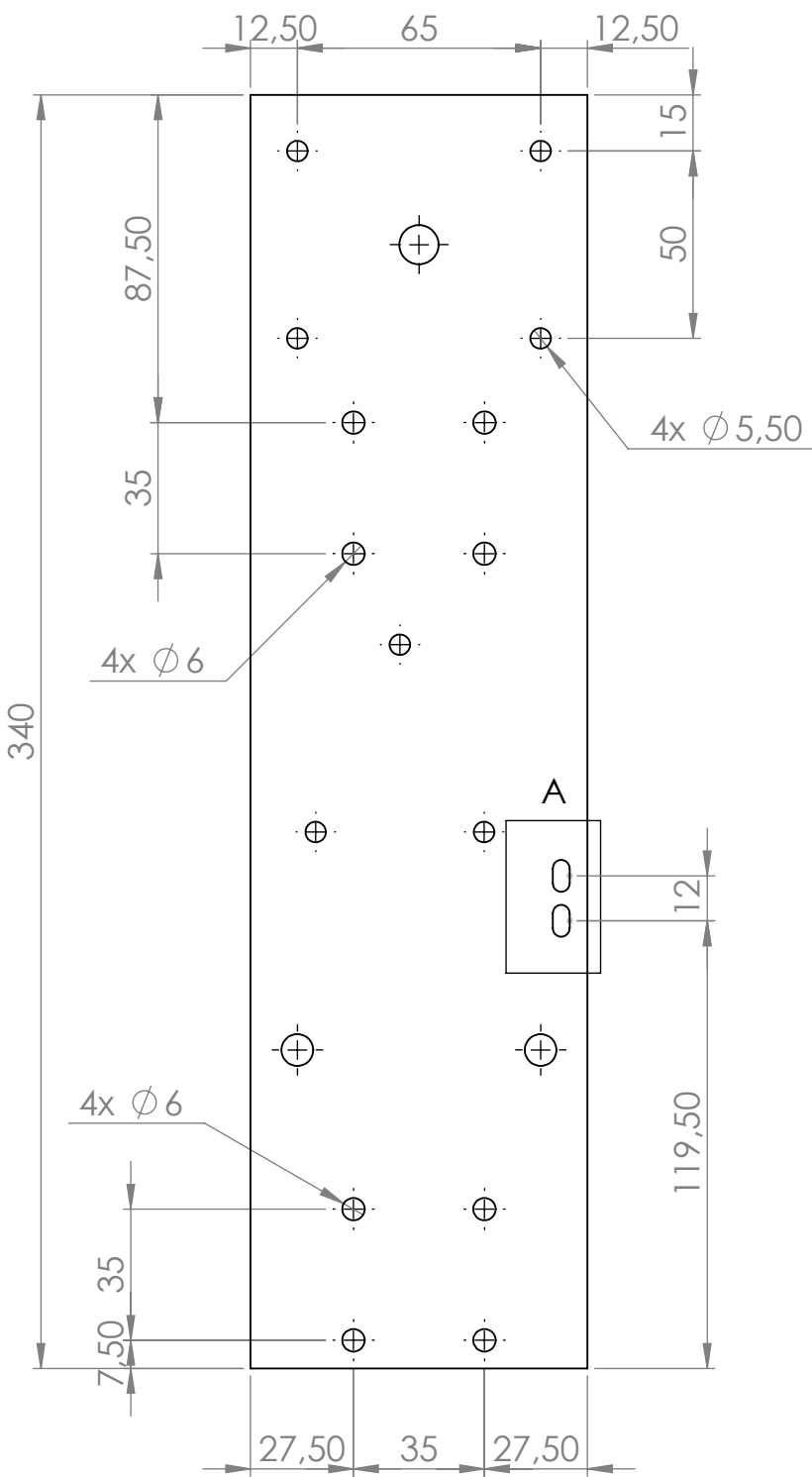
 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: PARTE MOVIL - EJE VERT. Y SOPORTES CENTRALES HOR.	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: PM	A3
AÑO: 2022	ESCALA: S/E	HOJA: 26/73	MATERIAL: -	



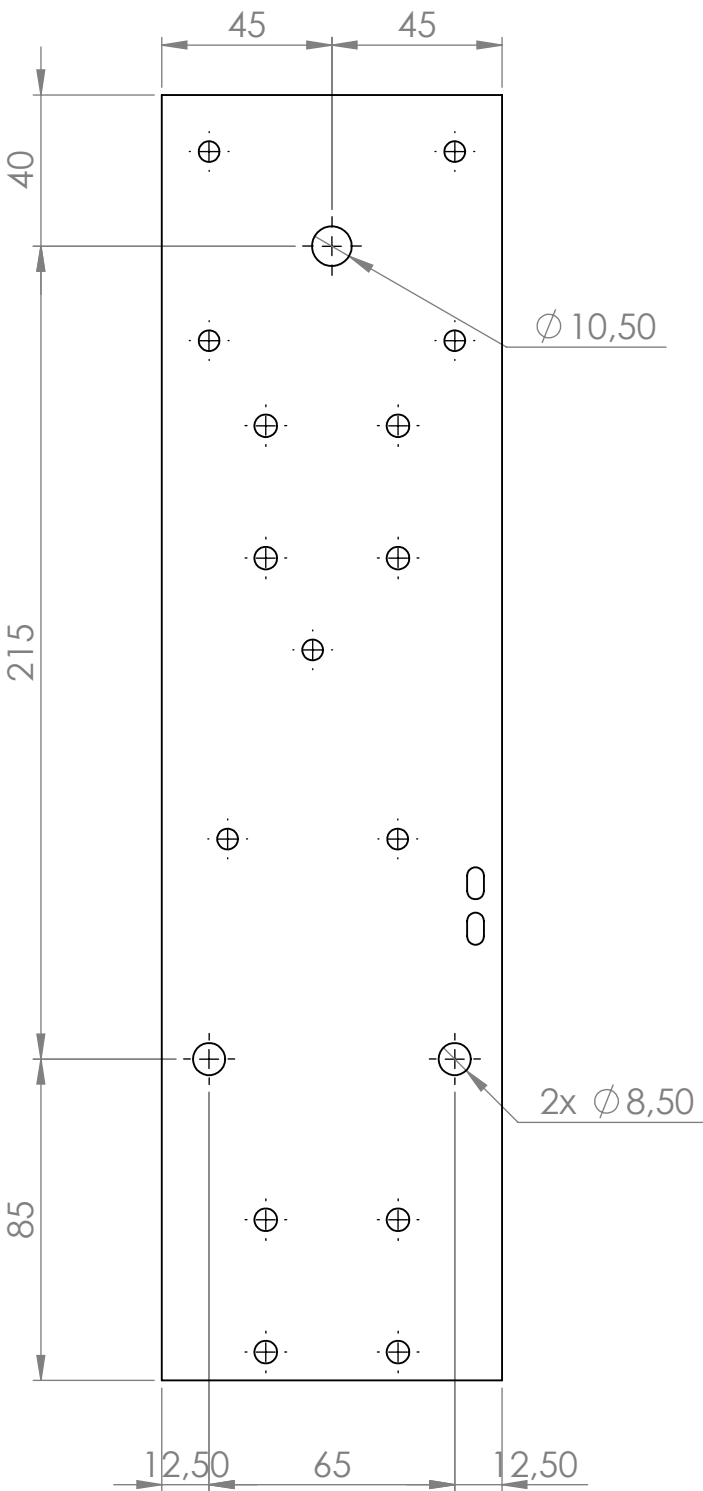
TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE VERTICAL GUÍA	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P1	A4
AÑO: 2022	ESCALA: S/E	HOJA: 27/73	MATERIAL: SAE 1010 TREFILADO	

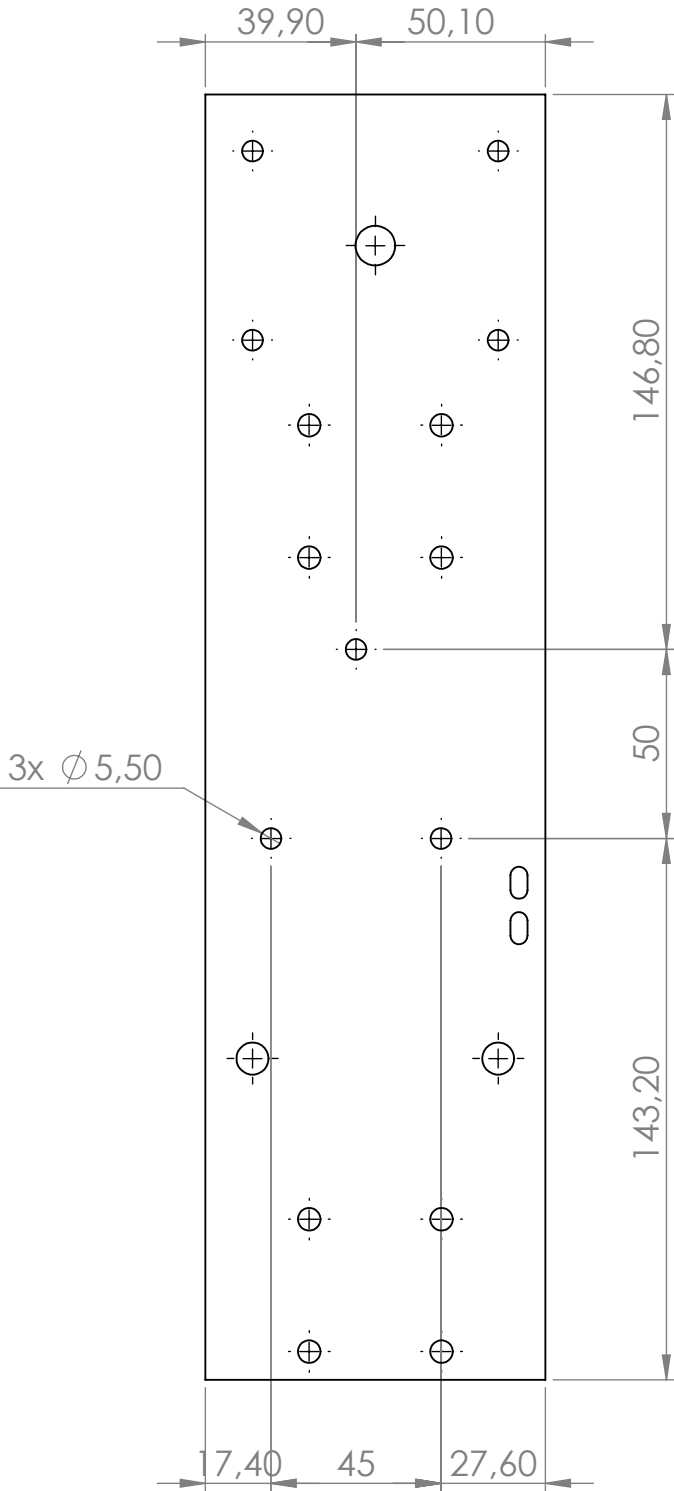
UNION A LOS SOPORTES DE EJES Y CHAPAS DE PROT



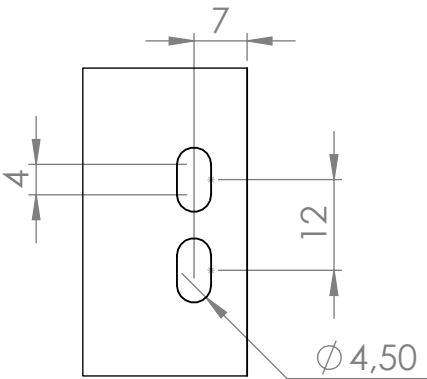
UNION AL EJE HOR. SUPERIOR



UNION DEL MOTOR VERTICAL



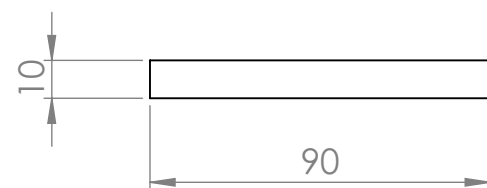
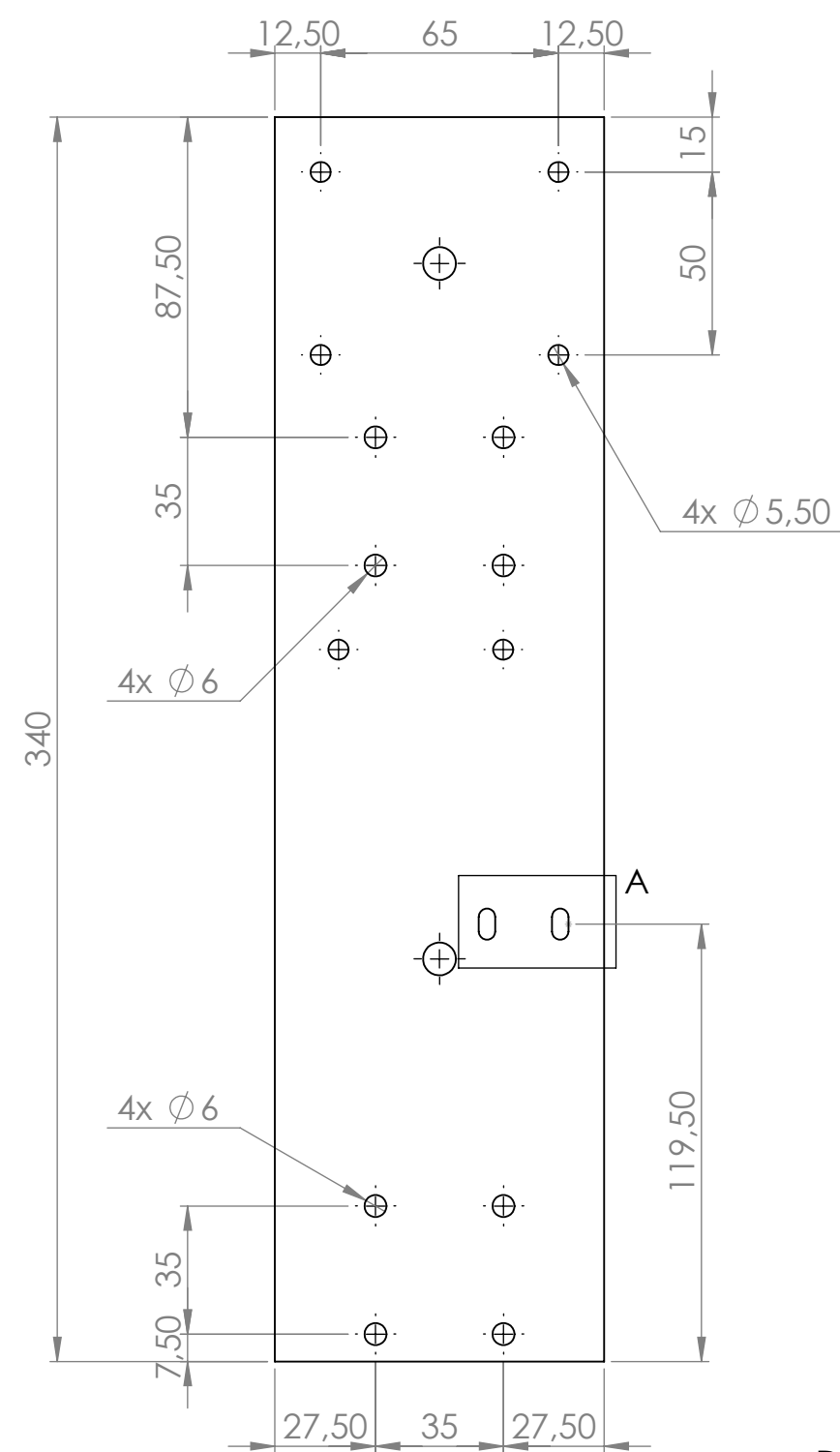
DETALLE A:
UNION A FIN
DE CARR.



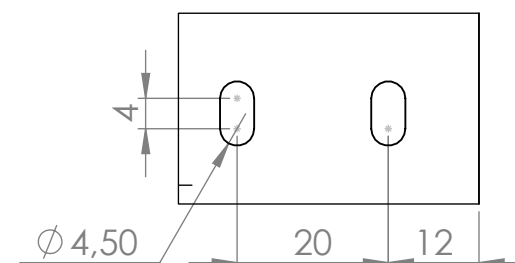
TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA	NOMBRE: SOPORTE VERTICAL SUPERIOR	
DISEÑO: OBREGON MARCELO JOAQUIN		REFERENCIA: P2
AÑO: 2022	ESCALA: S/E	HOJA: 28/73
MATERIAL: ALUMINIO		A3

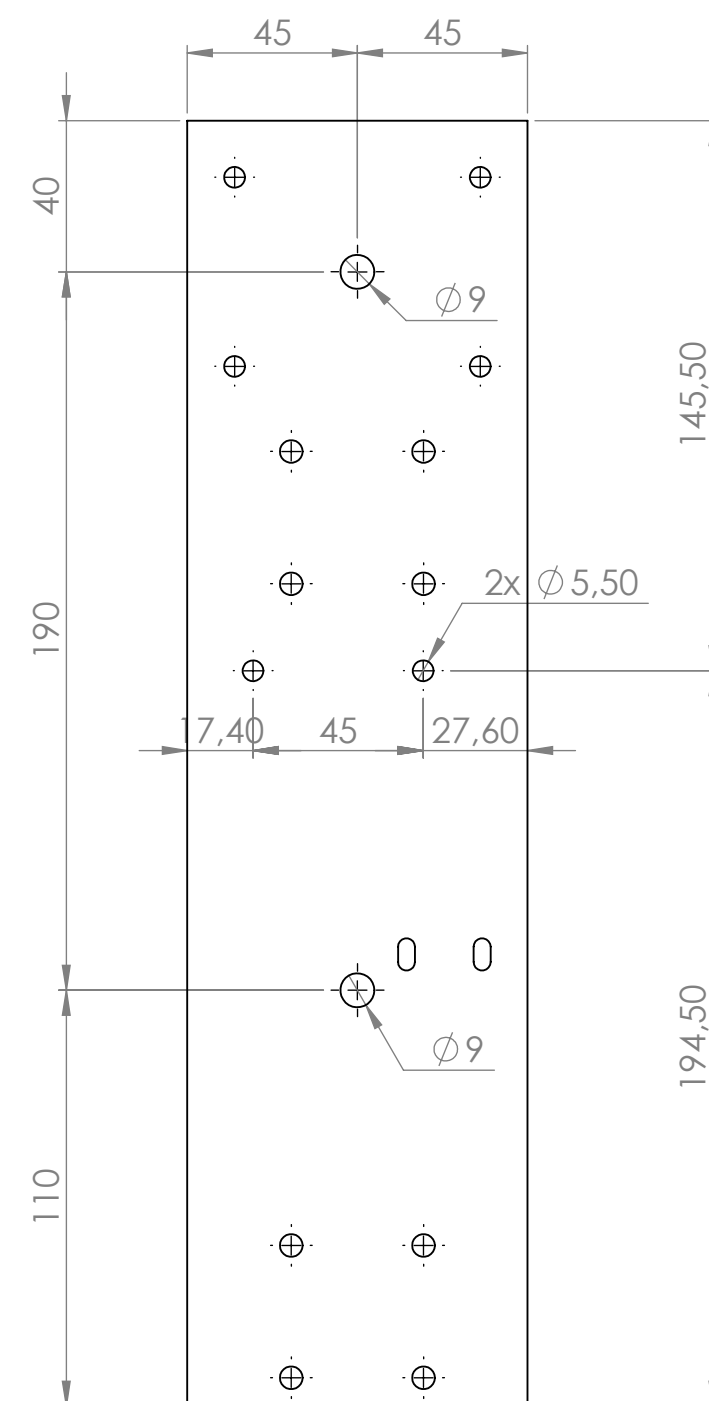
UNION A LOS SOPORTES DE EJES Y CHAPAS DE PROT



DETALLE A:
UNION A SENSOR
Y FIN DE CARR

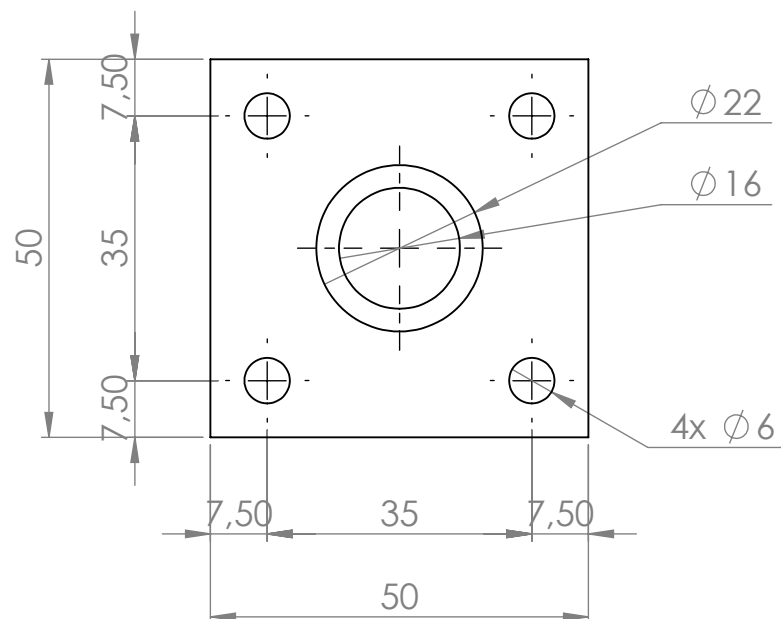
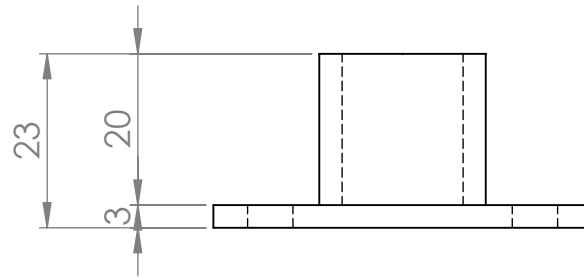
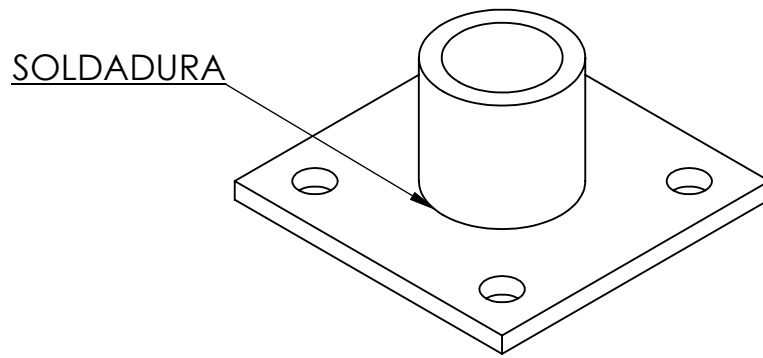


UNION AL EJE HOR. INFERIOR Y
A LA POLEA DE RETORNO



TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA	NOMBRE: SOPORTE VERTICAL INFERIOR	
DISEÑO: OBREGON MARCELO JOAQUIN		REFERENCIA: P3
AÑO: 2022	ESCALA: S/E	
HOJA: 29/73		MATERIAL: ALUMINIO
		A3



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE

FACULTAD DE INGENIERIA

DISEÑO:

OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
30/73

NOMBRE:

SOPORTE DE EJE
VERTICAL GUIA

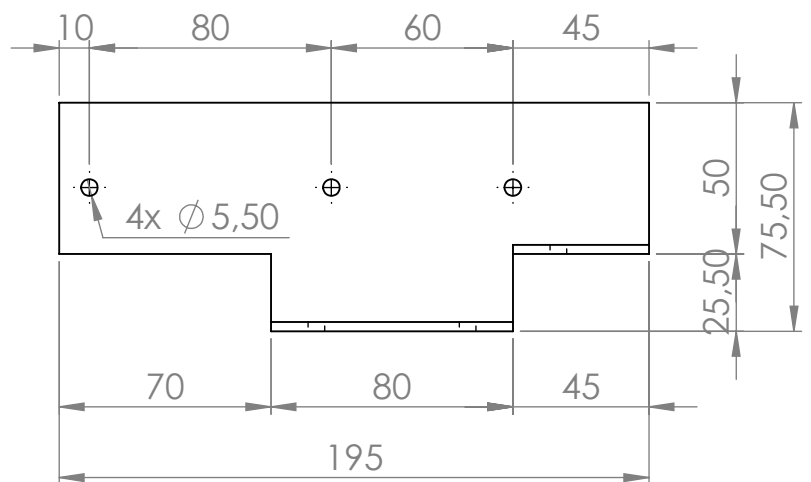
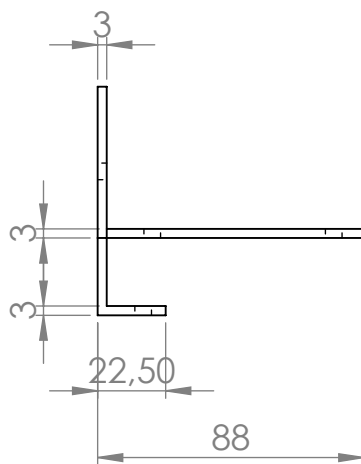
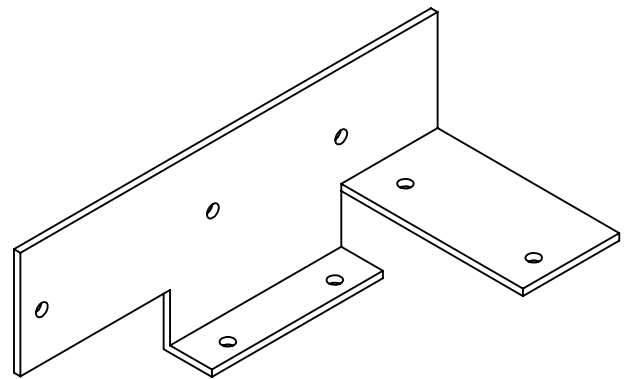
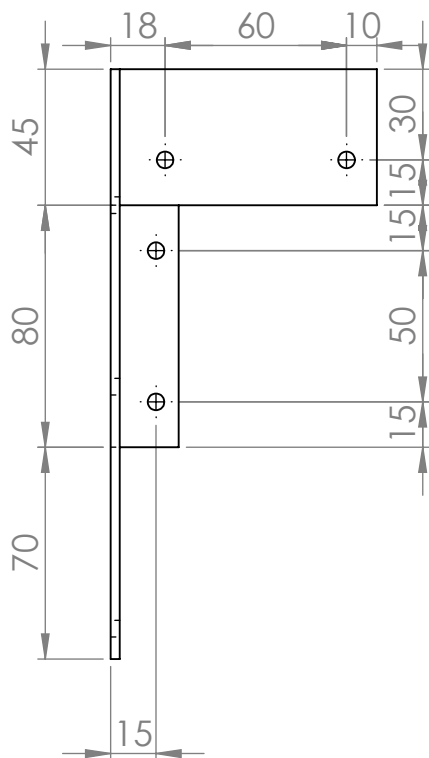
REFERENCIA:

P4

MATERIAL:

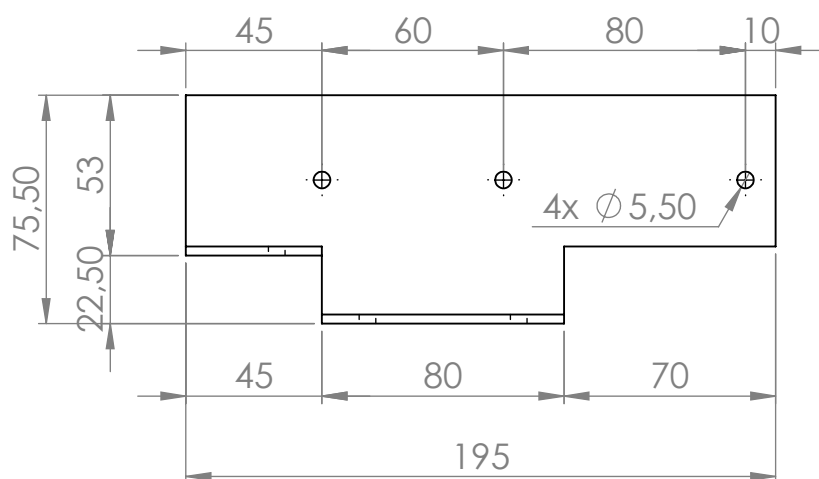
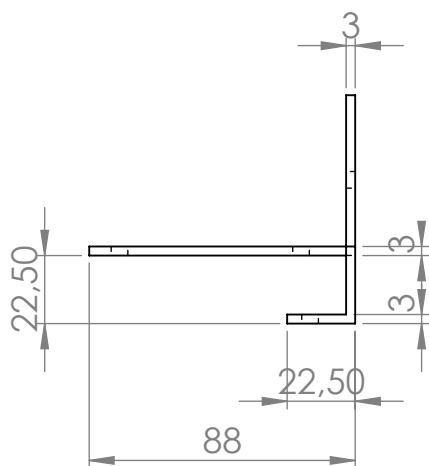
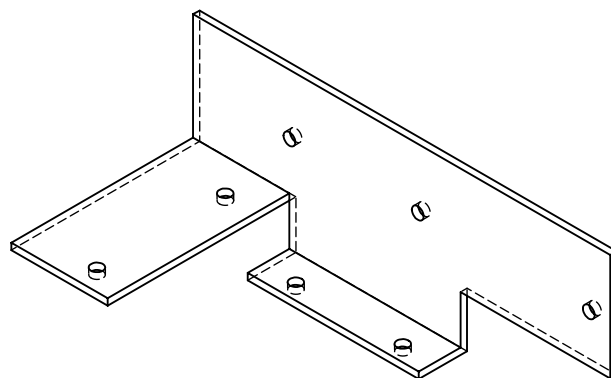
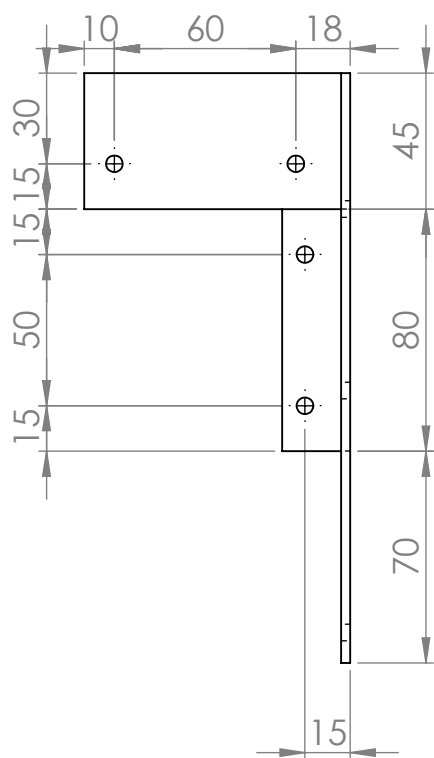
SAE 1040

A4



TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: SOPORTE DE CHAPA DE PROTECCION VERTICAL 1	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P5	A4
AÑO: 2022	ESCALA: S/E	HOJA: 31/73	MATERIAL: SAE 1010	



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

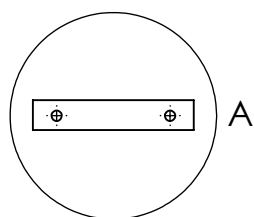
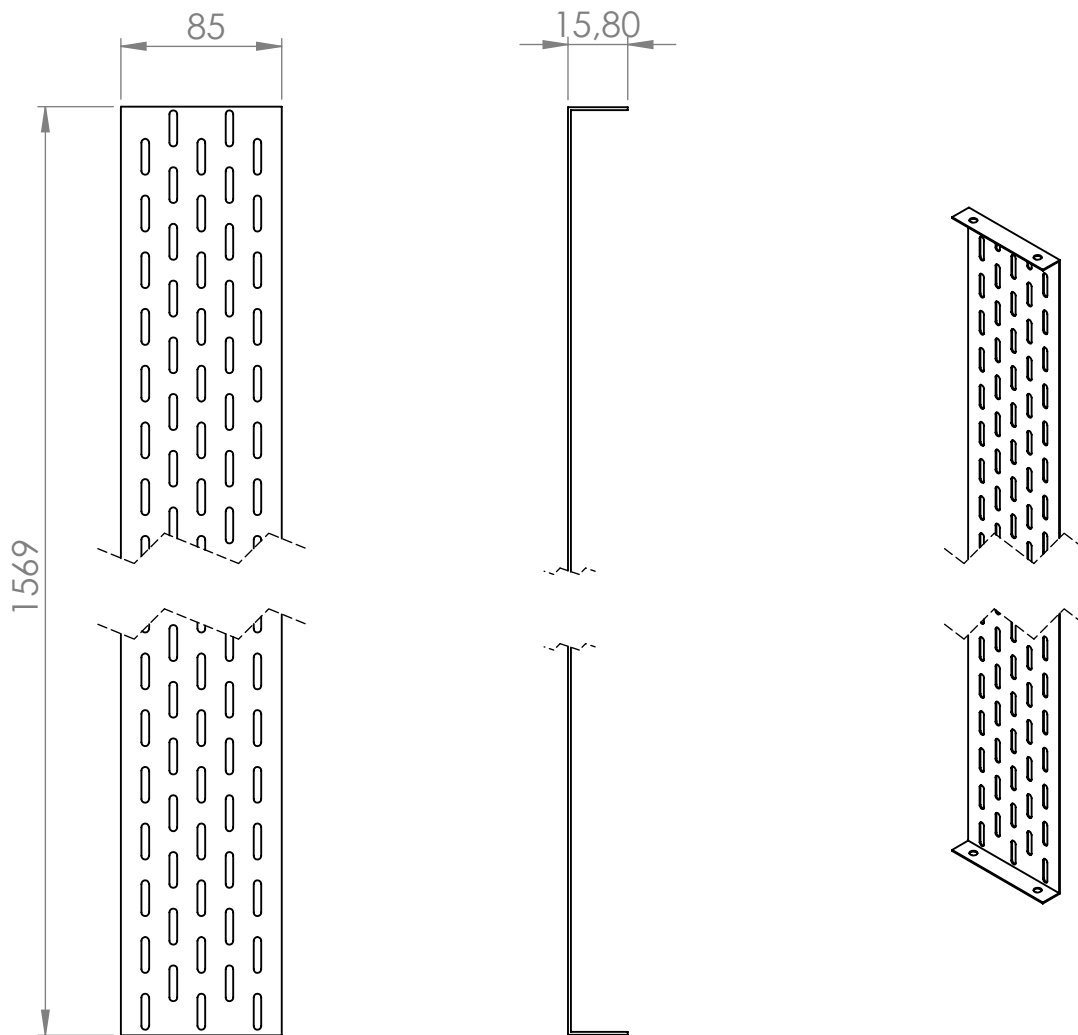
HOJA: 32/73

NOMBRE: SOPORTE DE CHAPA DE
PROTECCION VERTICAL 2

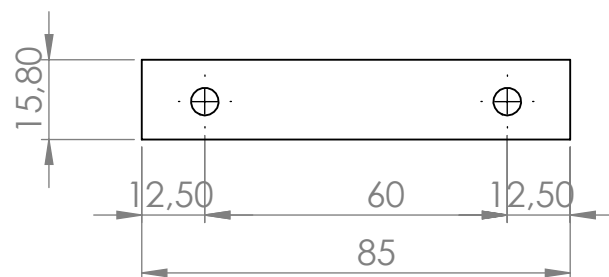
REFERENCIA: P6

MATERIAL: SAE 1010

A4

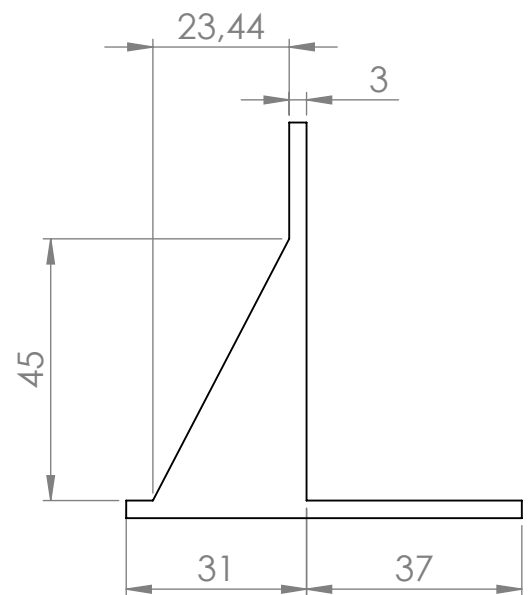
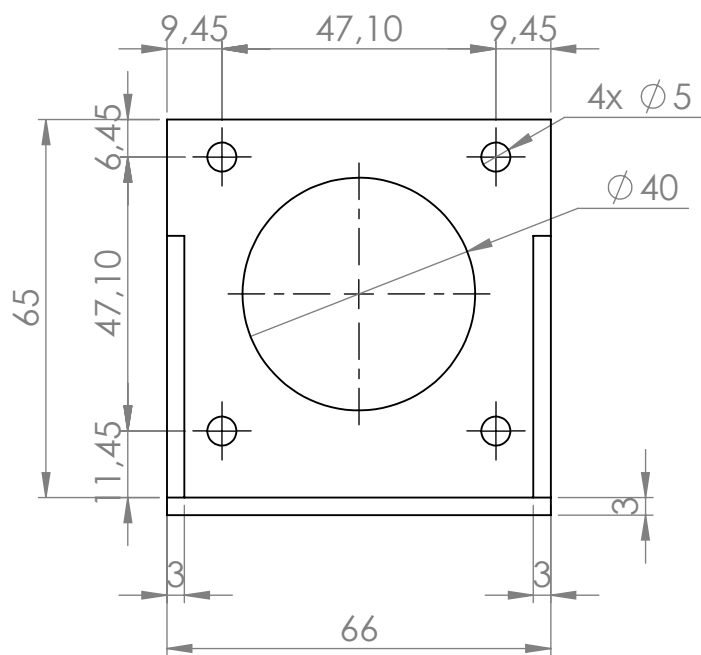
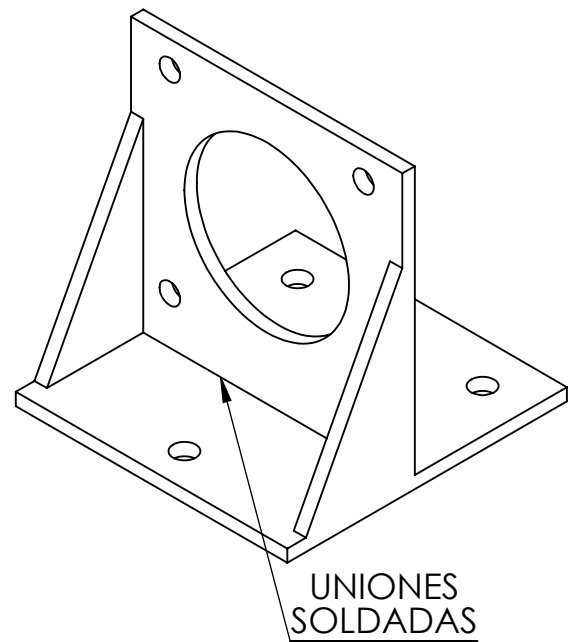
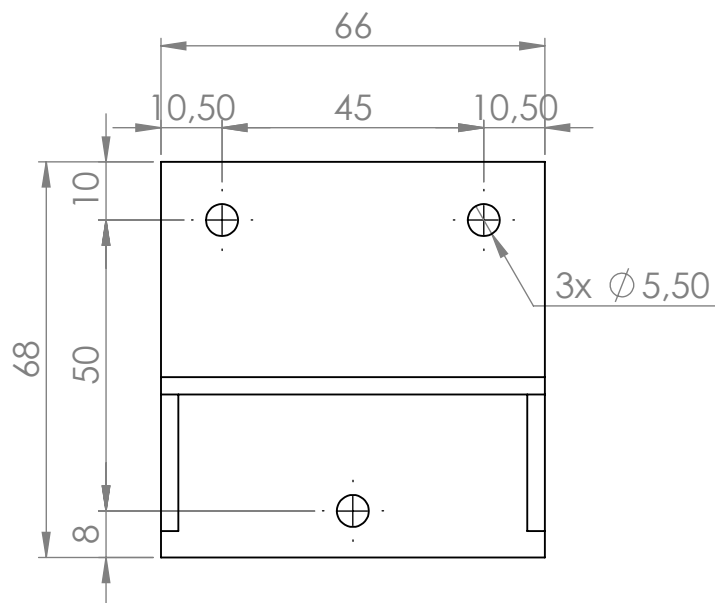


DETALLE A



TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: SOPORTE DE CADENA PORTACABLES VERTICAL	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P7	A4
AÑO: 2022	ESCALA: S/E	HOJA: 33/73	MATERIAL: CHAPA DE ACERO GALVANIZADO	



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

HOJA: 35/73

NOMBRE:

SOPORTE
MOTOR VERTICAL

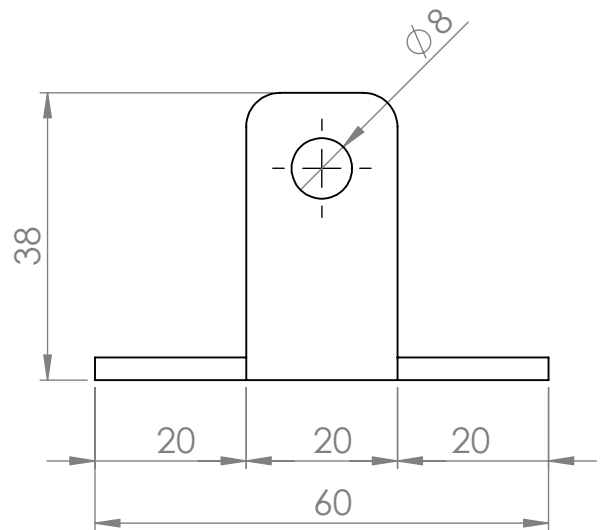
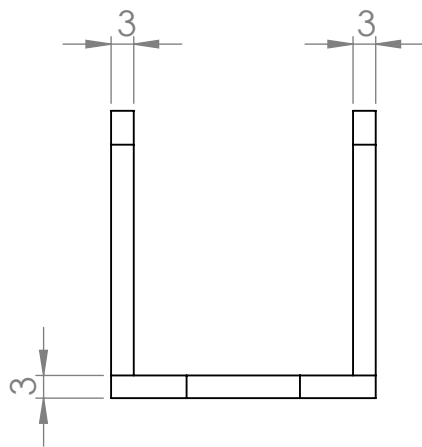
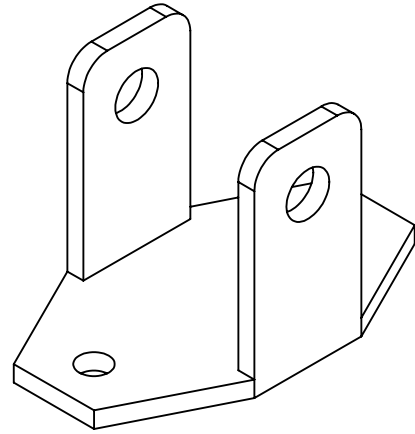
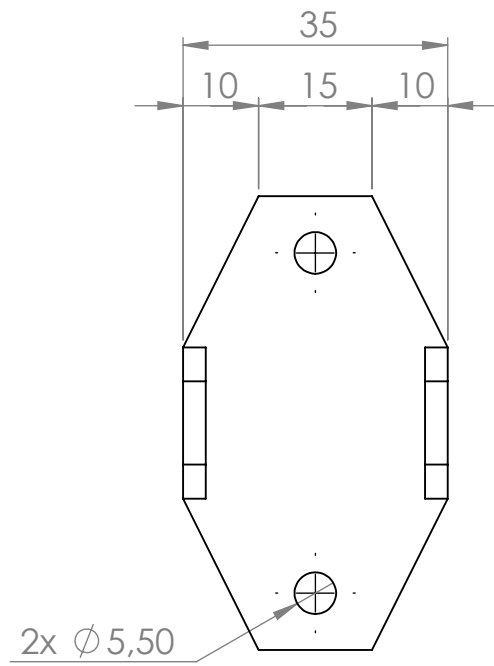
REFERENCIA:

P9

MATERIAL:

SAE 1010

A4



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE

FACULTAD DE INGENIERIA

DISEÑO:

OBREGON MARCELO JOAQUIN

AÑO:

2022

ESCALA:

S/E

HOJA:

36/73

NOMBRE:

SOPORTE POLEA
DE RETORNO VERTICAL

REFERENCIA:

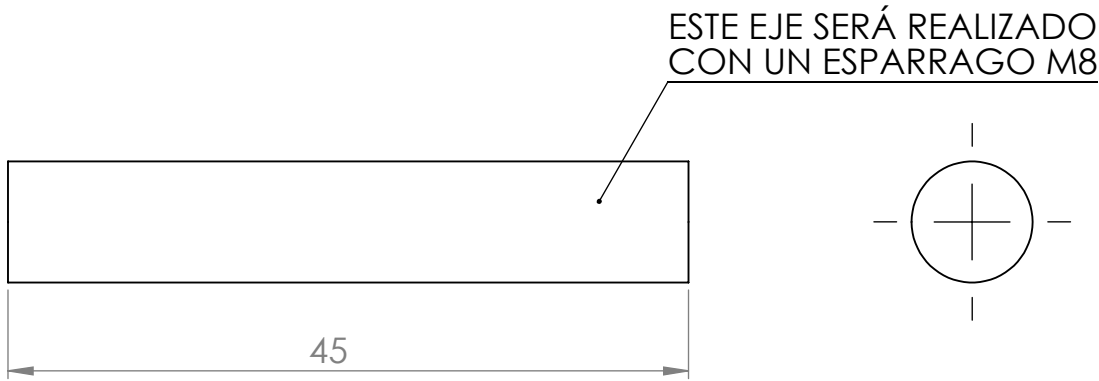
P10

MATERIAL:

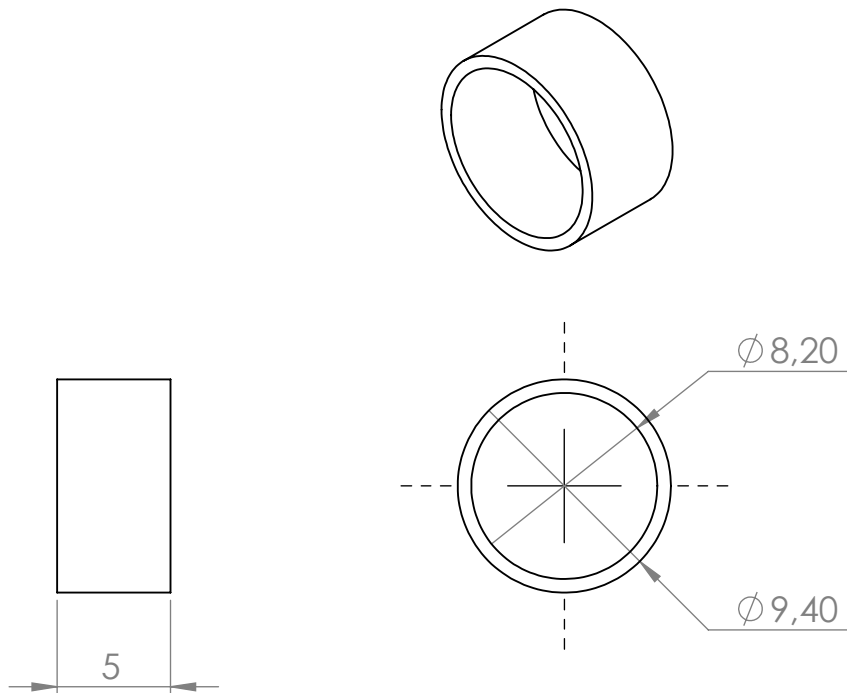
SAE 1010

A4

EJE POLEA DE RETORNO

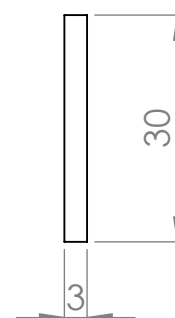
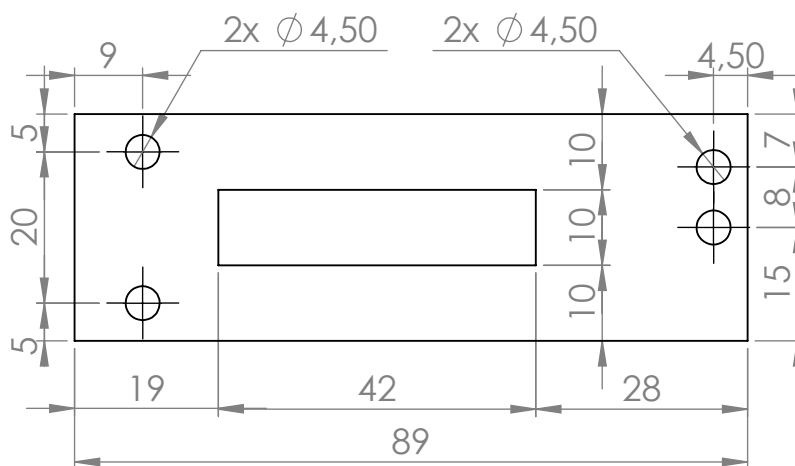
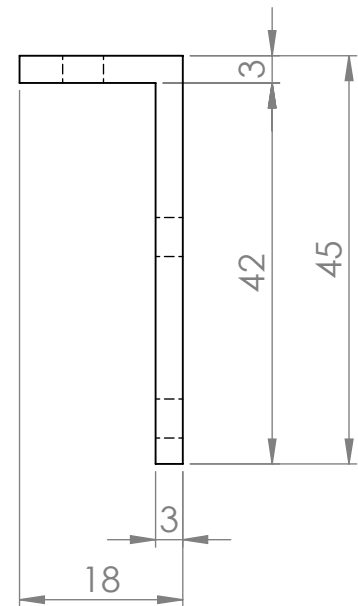
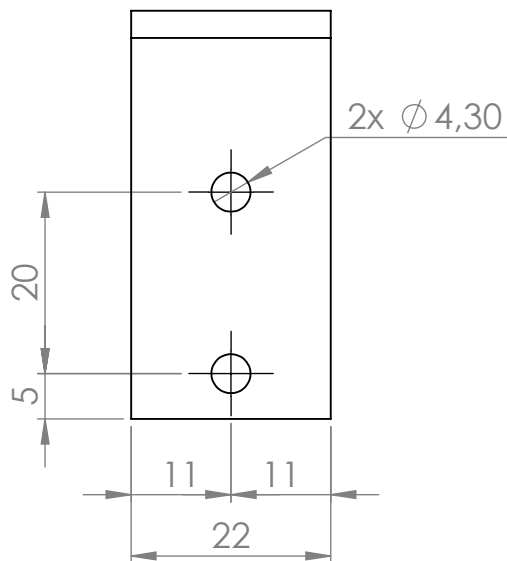
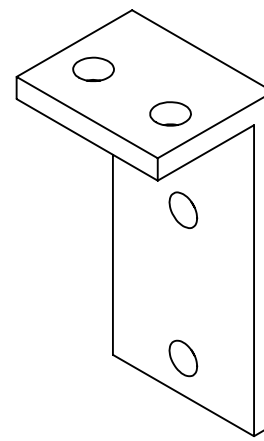
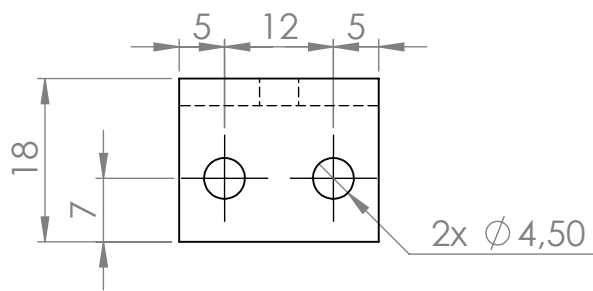


CASQUILLO POLEA DE RETORNO



TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE Y CASQUILO POLEA DE RETORNO VERTICAL	
			REFERENCIA: P11 Y P12	A4
			MATERIAL: SAE 1010	
DISEÑO: OBREGON MARCELO JOAQUIN				
AÑO: 2022	ESCALA: S/E	HOJA: 37/73		



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

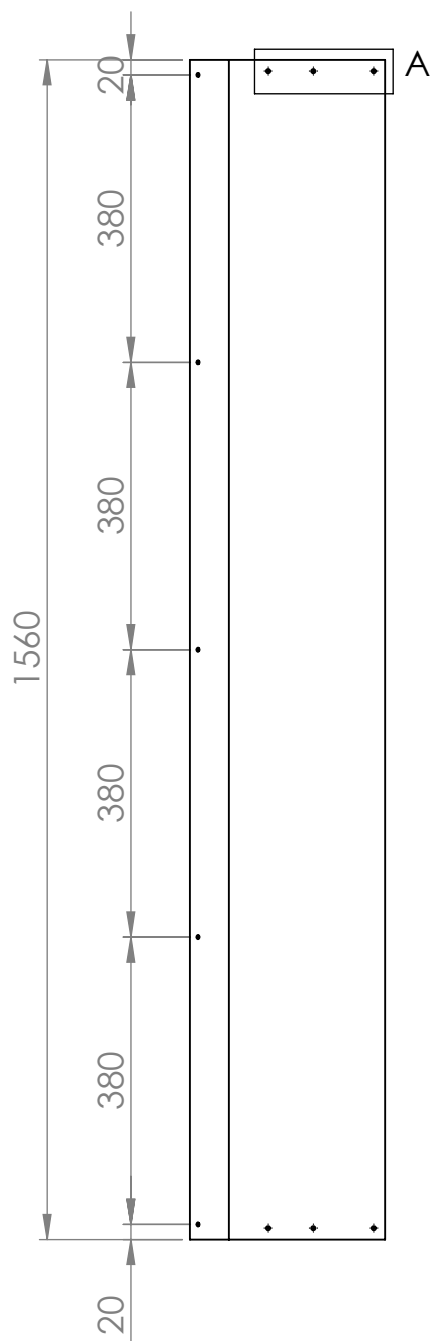
HOJA:
38/73

NOMBRE:
SOPORTE DE SENSOR
Y FINES DE CARRERA

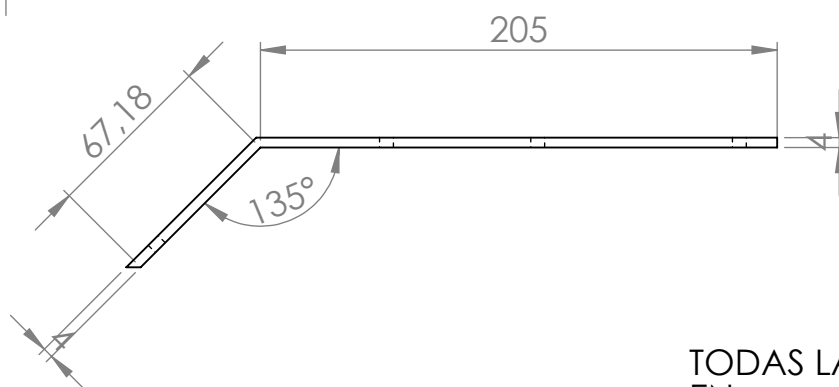
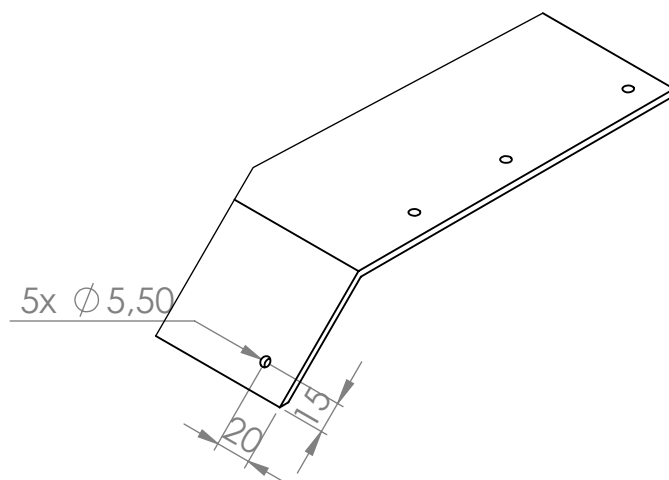
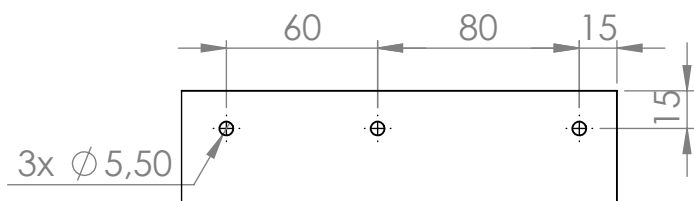
REFERENCIA:
P13 Y P14

MATERIAL:
SAE 1010

A4



DETALLE A



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

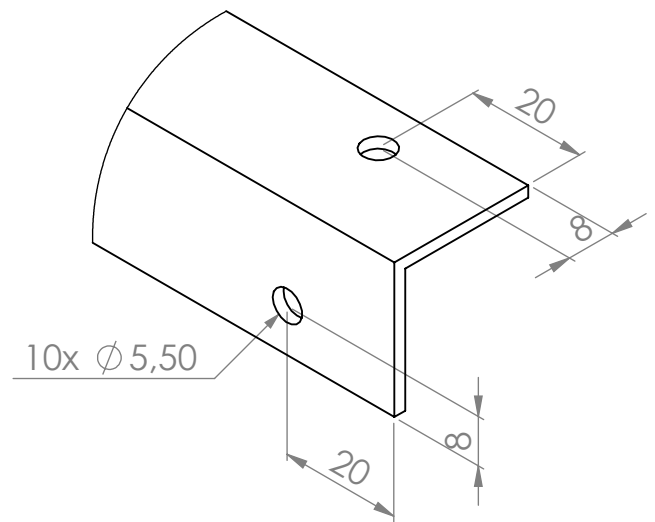
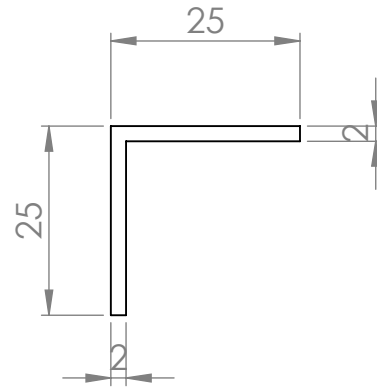
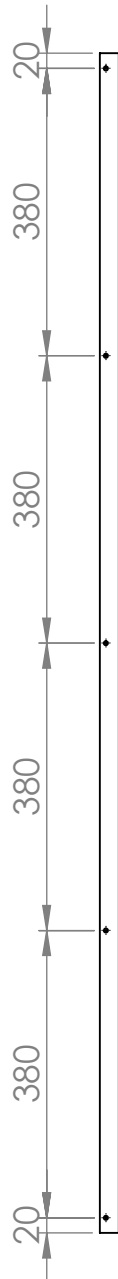
HOJA:
39/73

NOMBRE:
CHAPA DE PROTECCIÓN
VERTICAL - MITAD

REFERENCIA:
P15

MATERIAL:
POLICARBONATO
COMPACTO

A4



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE

FACULTAD DE INGENIERIA

DISEÑO:

OBREGON MARCELO JOAQUIN

AÑO:

2022

ESCALA:

S/E

HOJA:

40/73

NOMBRE:

PERFIL FRONTAL

REFERENCIA:

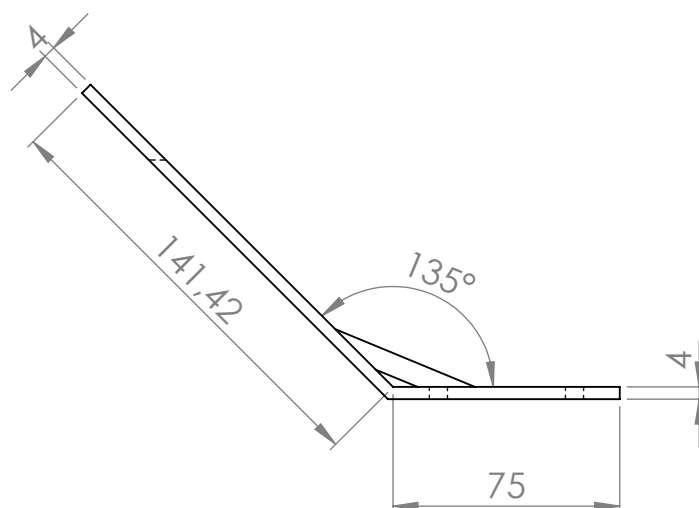
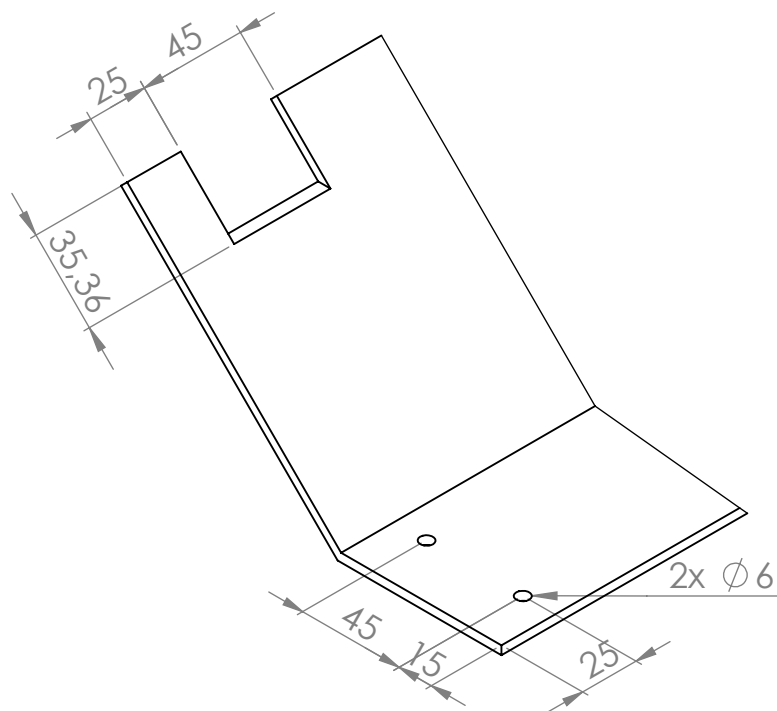
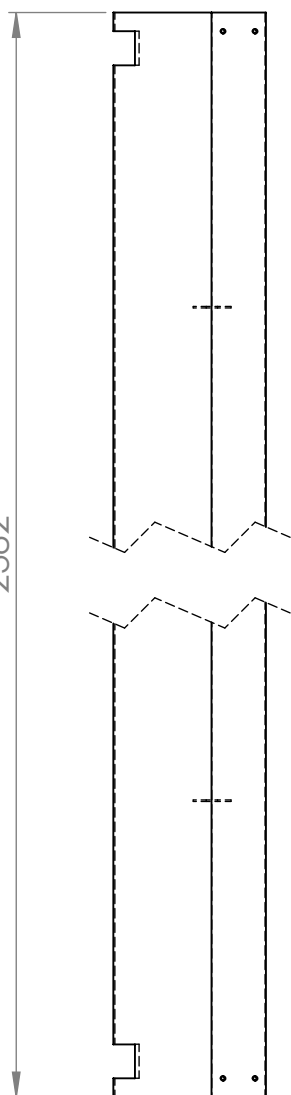
P16

MATERIAL:

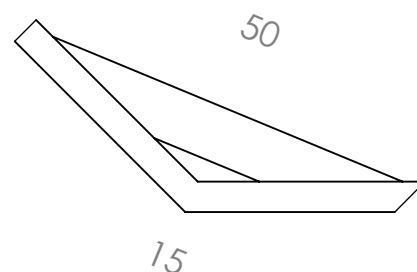
ALUMINIO

A4

2382

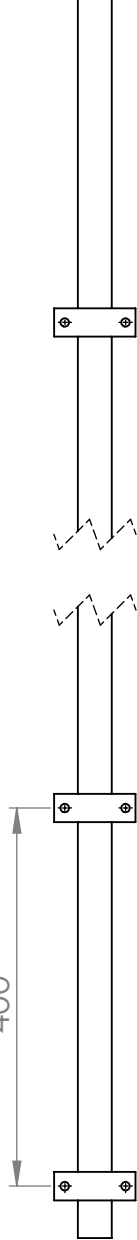
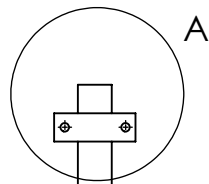


NERVIOS
CANTIDAD: 5
SEPARACION: 40mm



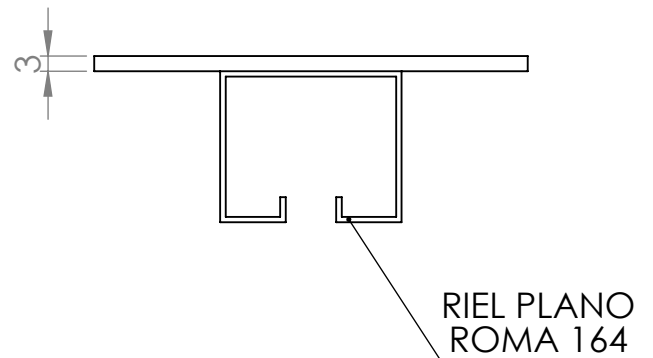
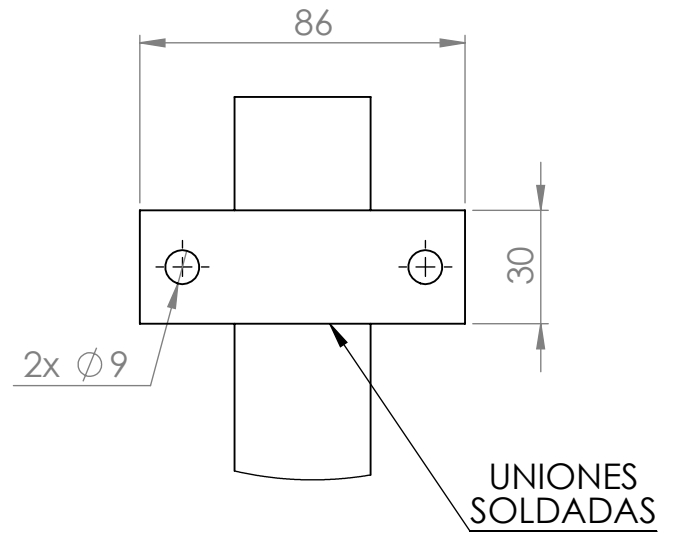
TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: CHAPA DE PROTECCIÓN HORIZONTAL SUPERIOR	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P17	A4
AÑO: 2022	ESCALA: S/E	HOJA: 41/73	MATERIAL: POLICARBONATO COMPACTO	



LONGITUD TOTAL
SEGUN NECESIDAD

DETALLE A



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
42/73

NOMBRE:

RIEL PLANO

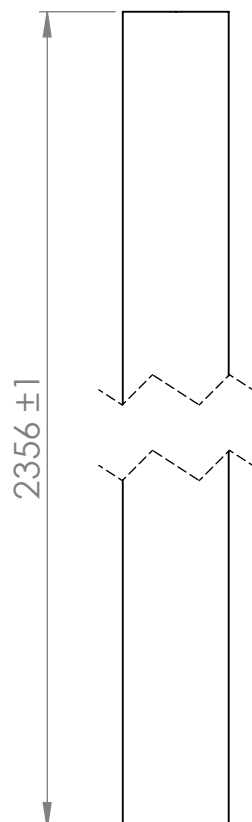
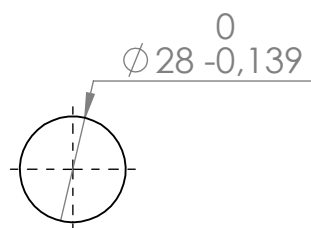
REFERENCIA:

P18

MATERIAL:

ACERO
GALVANIZADO

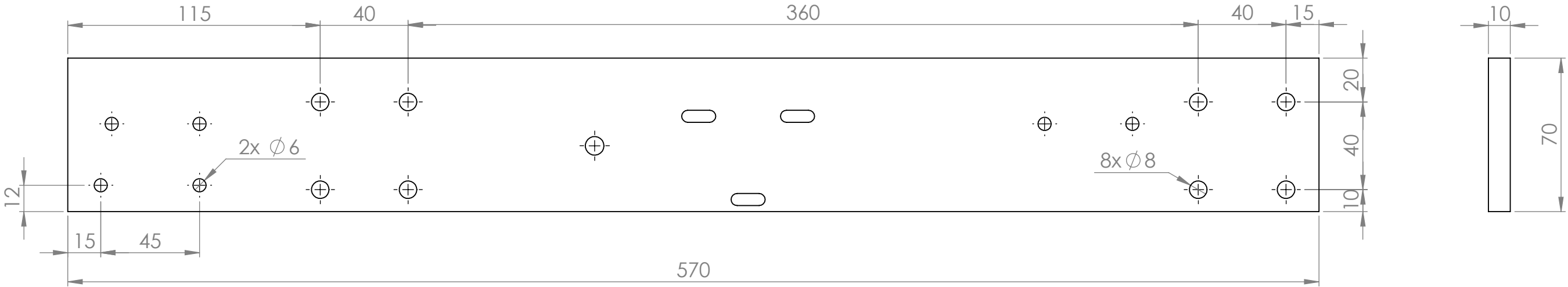
A4



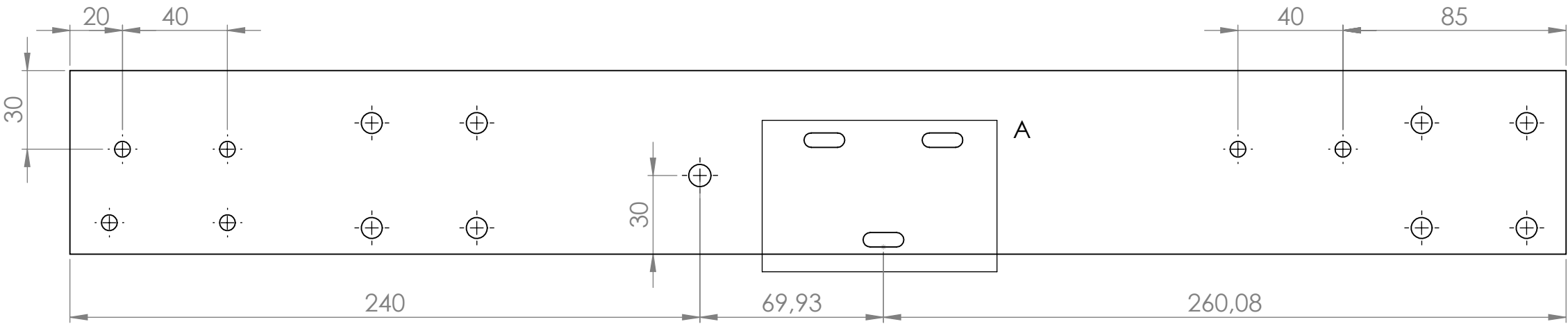
TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJE HORIZONTAL GUÍA	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P19	A4
AÑO: 2022	ESCALA: S/E	HOJA: 43/73	MATERIAL: SAE 1010 TREFILADO	

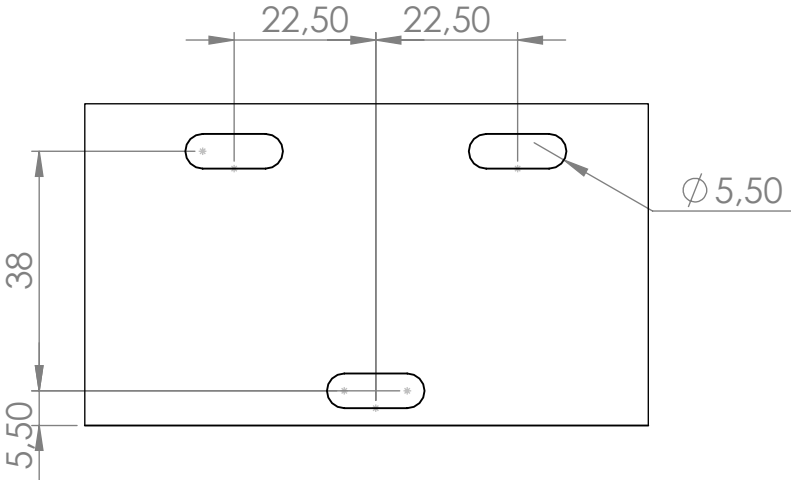
UNION A LOS SOPORTES DE EJES Y CHAPA DE PROT



UNION AL SOPORTE DE CARRO Y TRABA AXIAL



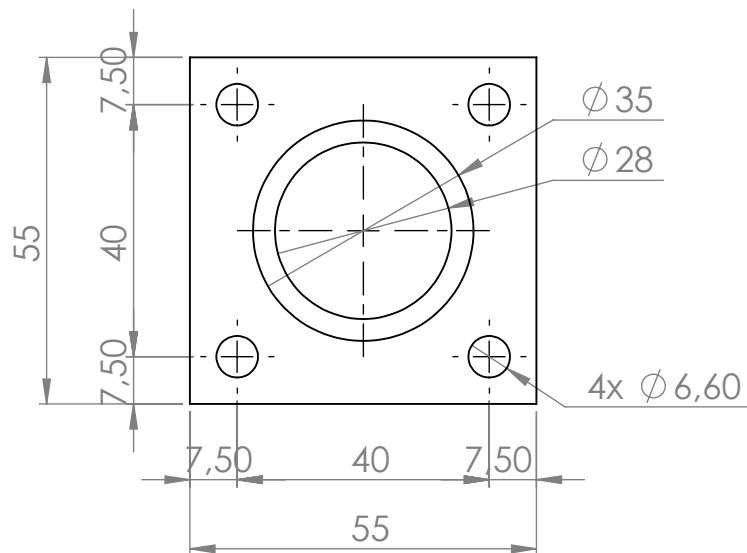
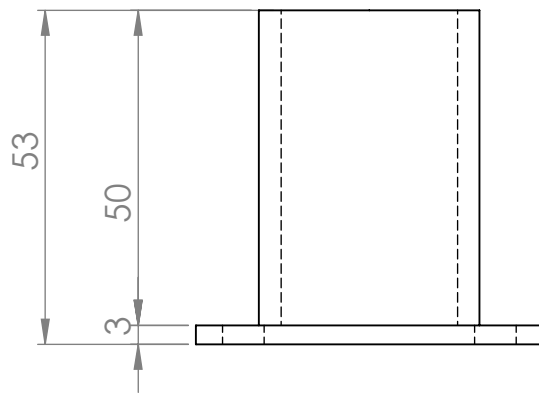
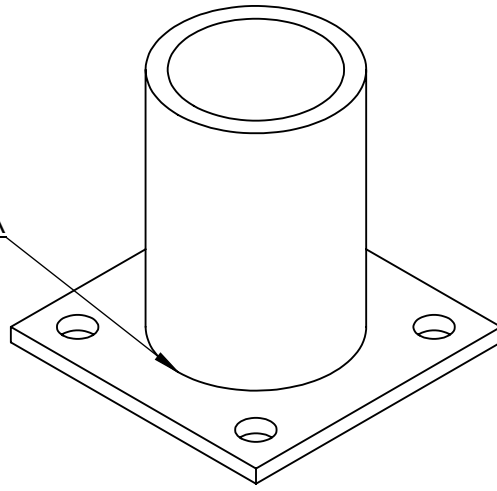
DETALLE A
SUJECION DEL MOTOR
Y POLEA DE RETORNO



TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA	NOMBRE: SOPORTE HORIZONTAL EJE SUPERIOR	
DISEÑO: OBREGON MARCELO JOAQUIN		REFERENCIA: P20
AÑO: 2022	ESCALA: S/E	HOJA: 44/73
MATERIAL: ALUMINIO		A3

SOLDADURA



TODAS LAS SIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
45/73

NOMBRE:

SOPORTE DE EJE
HORIZONTAL GUIA

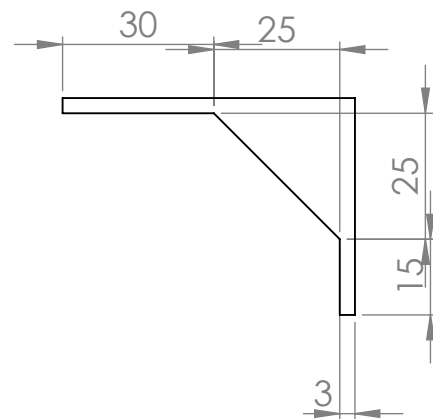
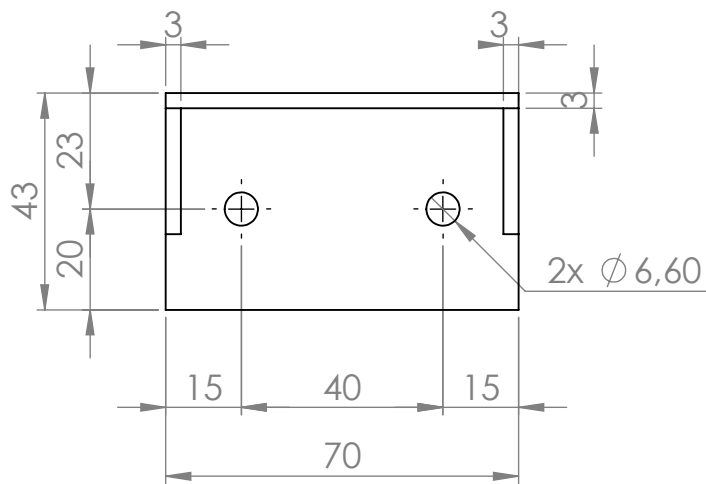
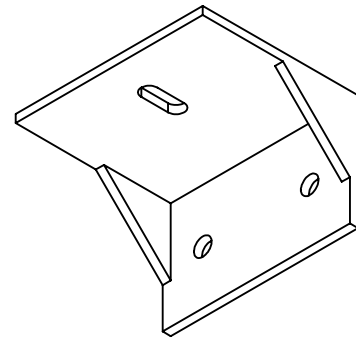
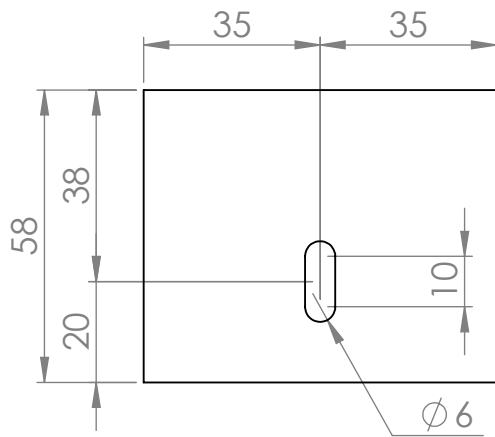
REFERENCIA:

P21

MATERIAL:

SAE 1040

A4



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

HOJA: 46/73

NOMBRE:

SOPORTE CARRO
SUPERIOR

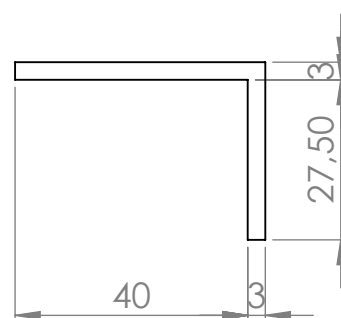
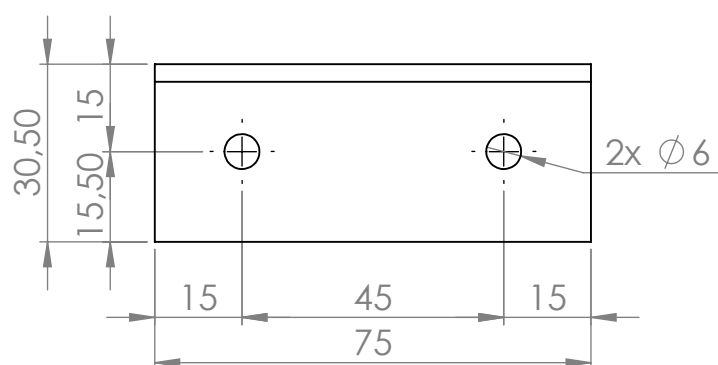
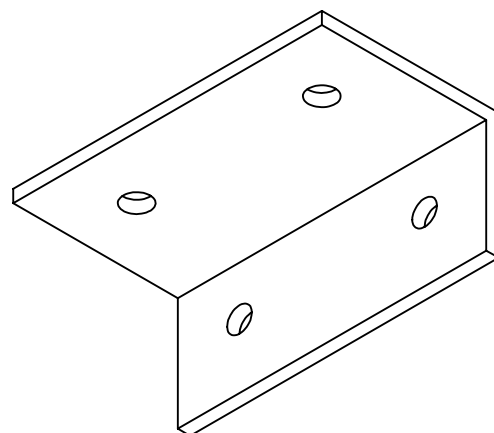
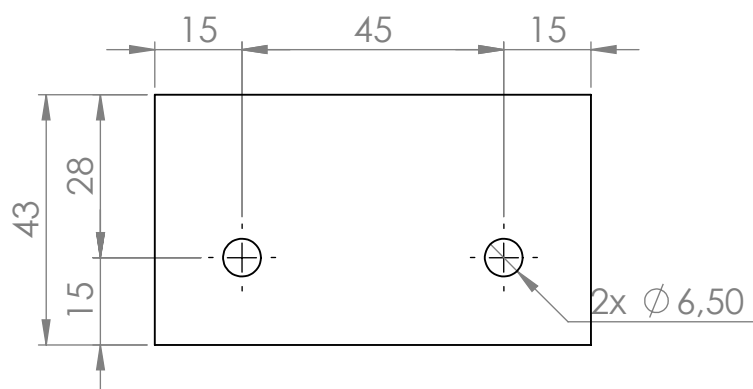
REFERENCIA:

P22

MATERIAL:

SAE 1010

A4



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

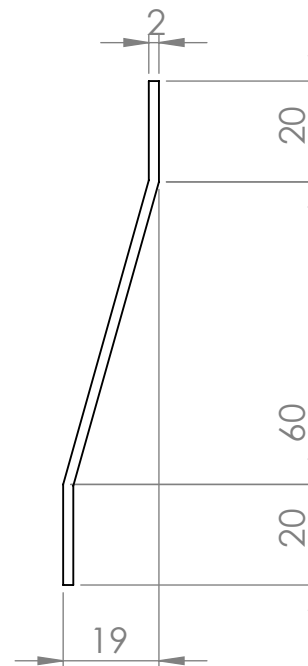
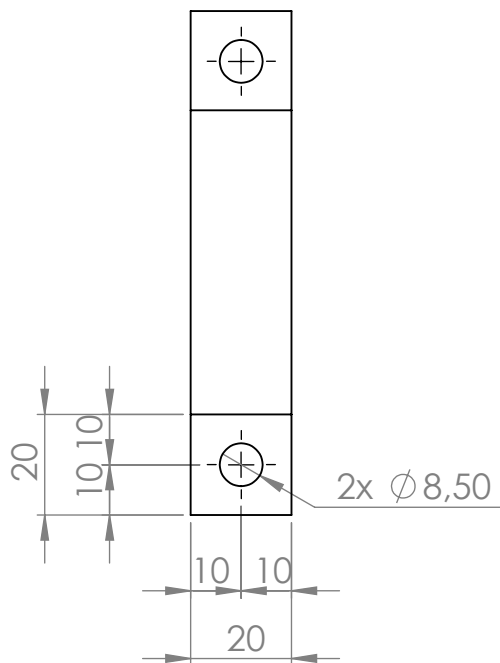
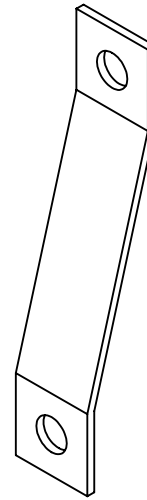
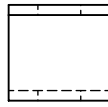
HOJA:
47/73

NOMBRE:
SOPORTE DE CHAPA
DE PROT. HOR SUPERIOR

REFERENCIA:
P23

MATERIAL:
SAE 1010

A4



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

HOJA: 48/73

NOMBRE:

TRABA AXIAL
EJE HOR. SUPERIOR

REFERENCIA:

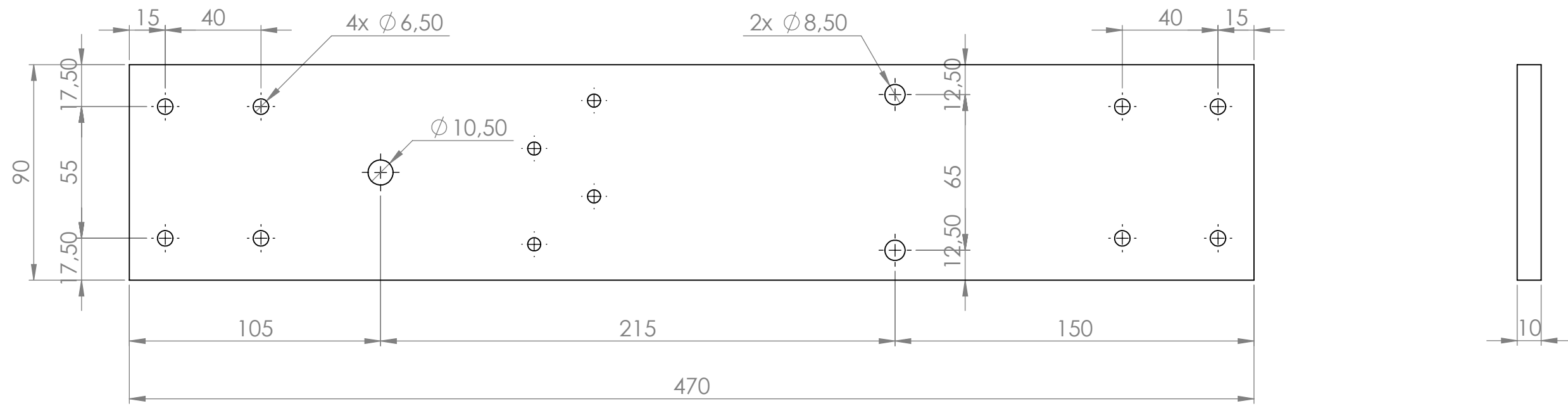
P24

MATERIAL:

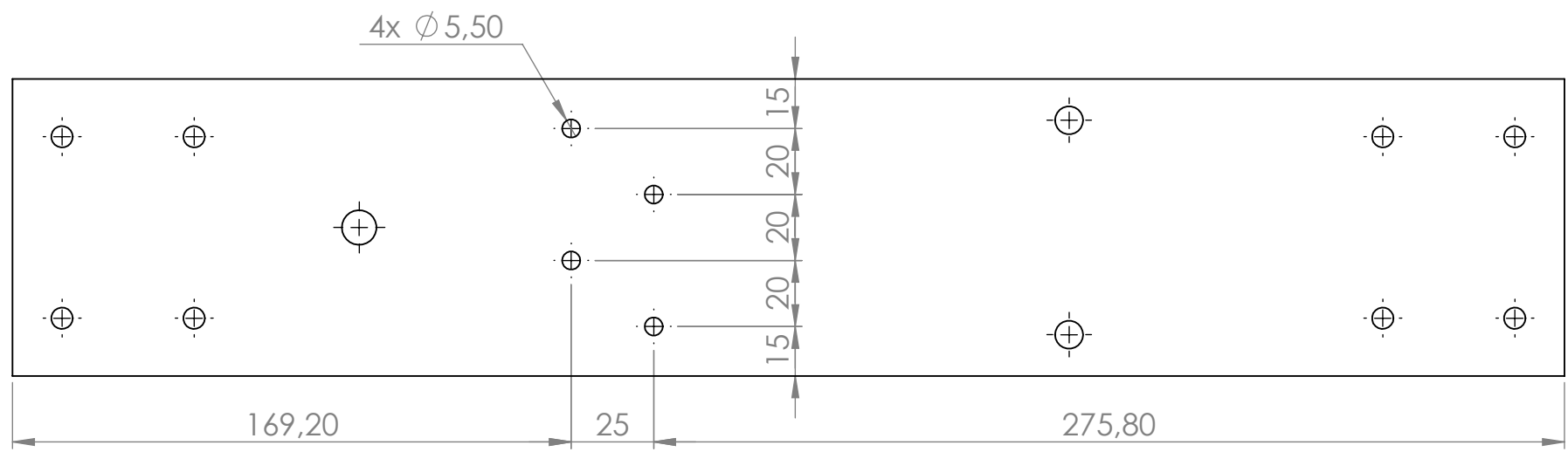
SAE 1010

A4

UNION A BUJES Y EJE VERTICAL

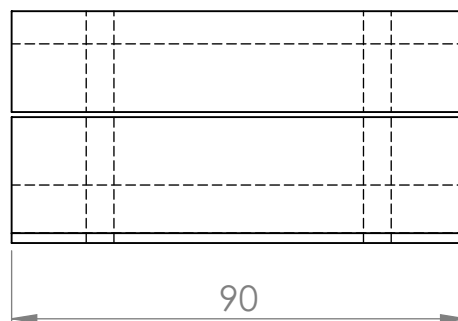
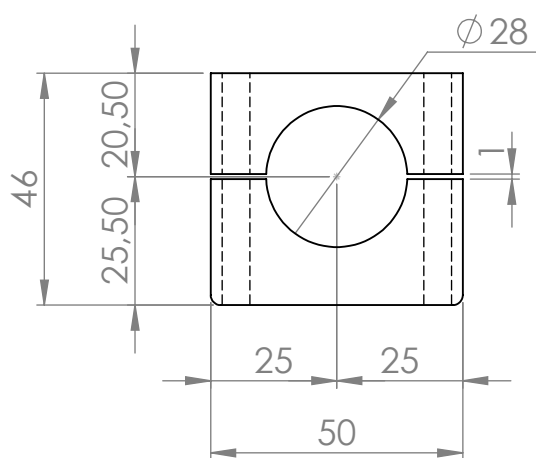
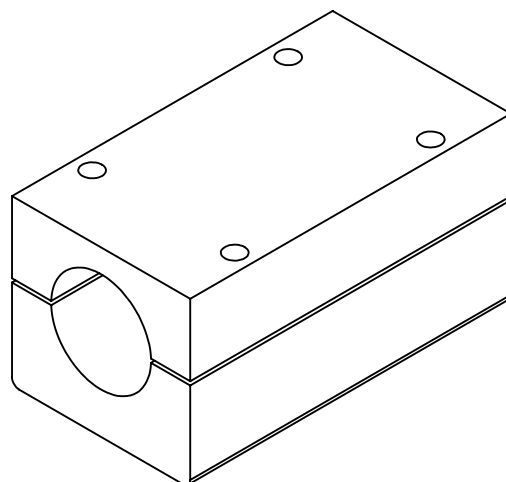
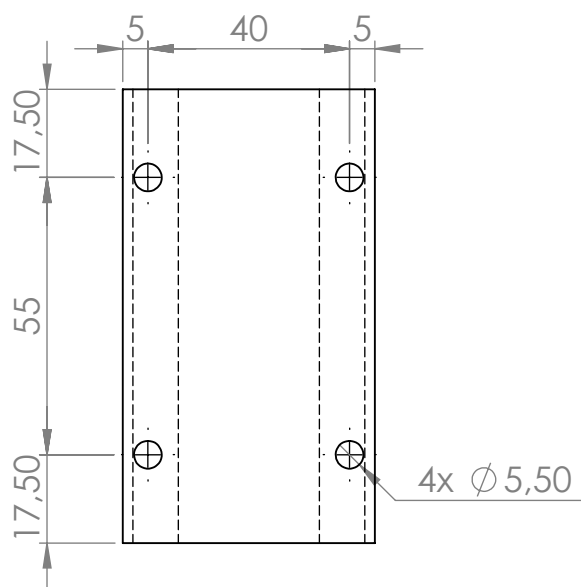


UNION A CORREA HORIZONTAL



TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA	NOMBRE: SOPORTE CENTRAL EJE HOR. SUPERIOR	
DISEÑO: OBREGON MARCELO JOAQUIN		A3
AÑO: 2022	ESCALA: S/E	
HOJA: 49/73		MATERIAL: ALUMINIO



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

HOJA: 50/73

NOMBRE:

BUJES PARA EJES
HORIZONTALES

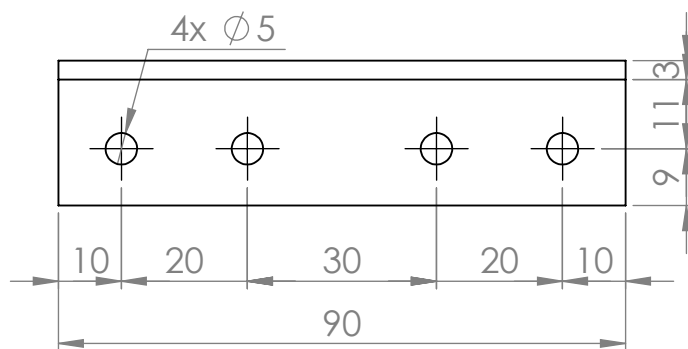
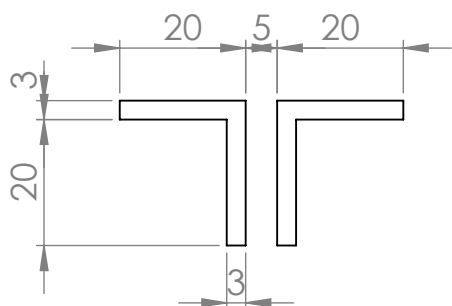
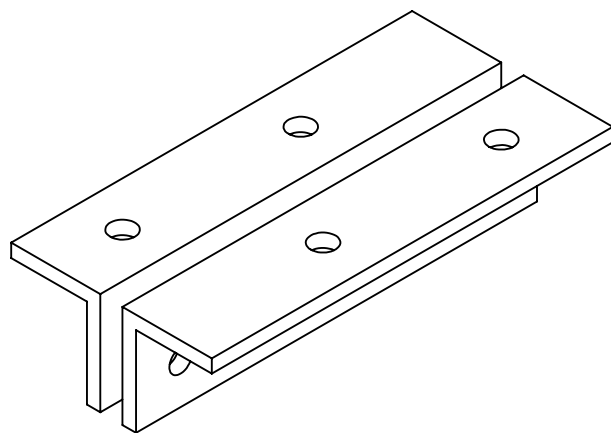
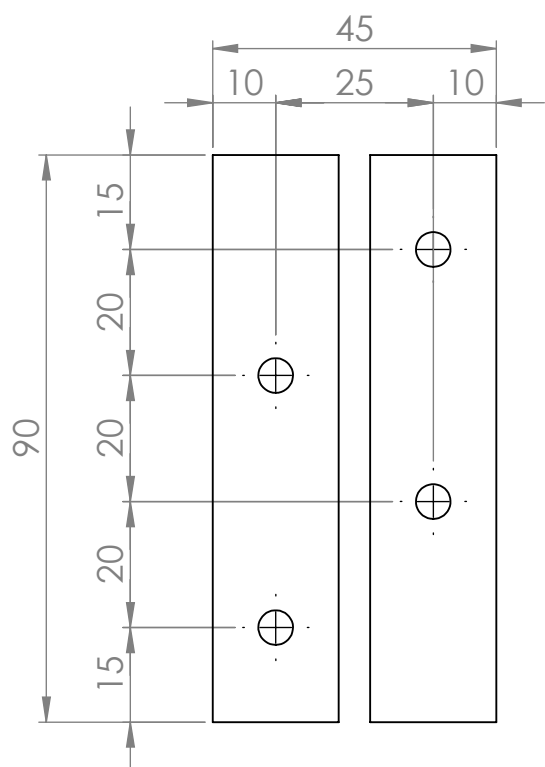
REFERENCIA:

P26

MATERIAL:

GRILON

A4



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

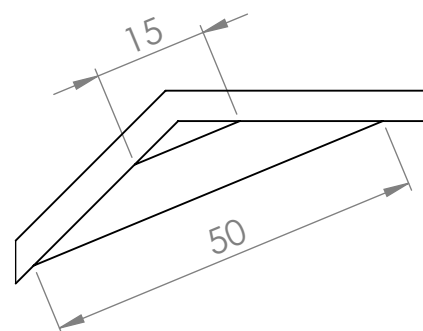
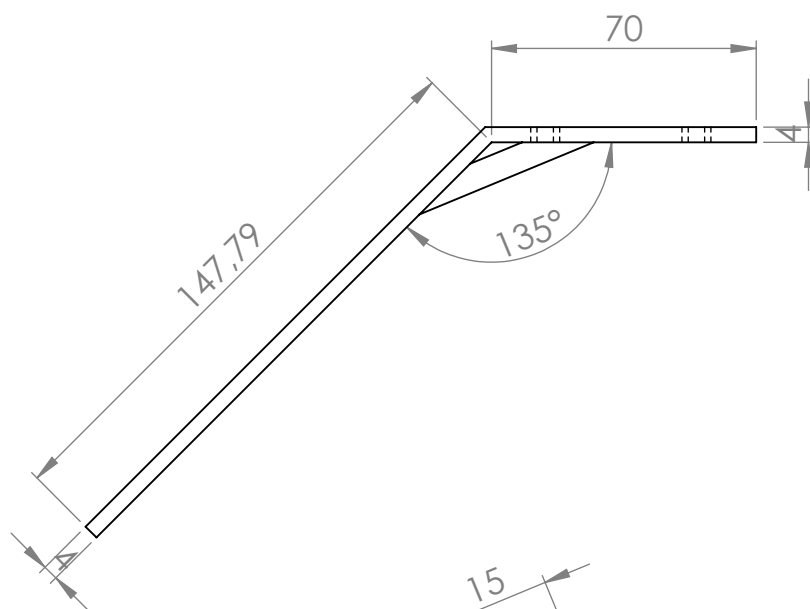
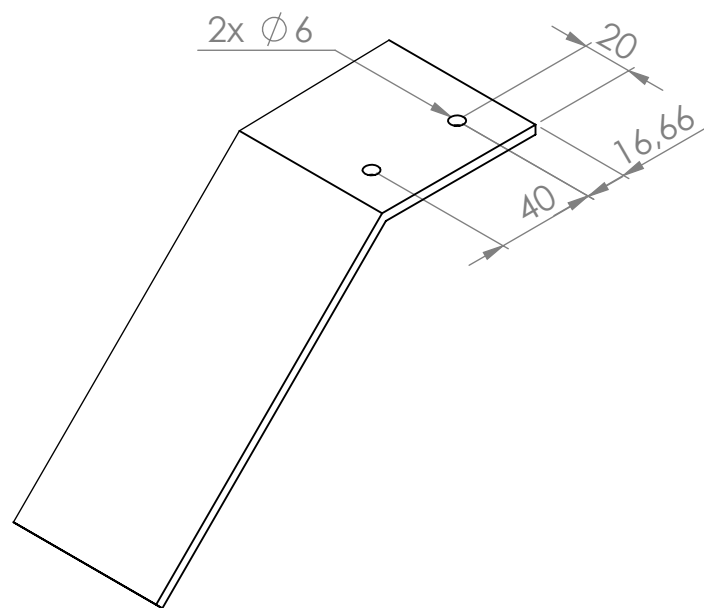
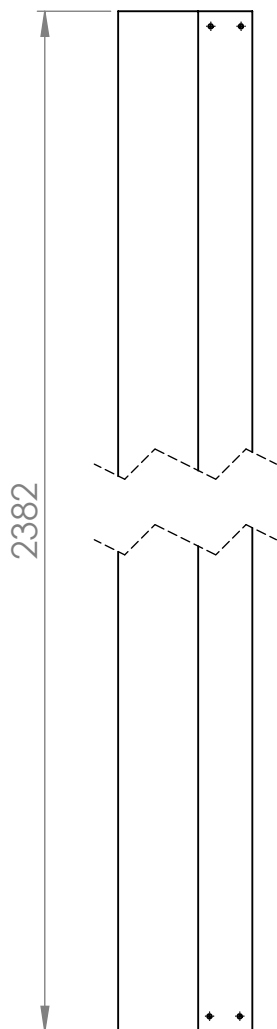
HOJA: 51/73

NOMBRE: SUJECIÓN DE
CORREA HORIZONTAL

REFERENCIA: P27

MATERIAL: SAE 1010

A4



NERVIOS
CANTIDAD: 5
SEPARACION: 40mm

TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE

FACULTAD DE INGENIERIA

DISEÑO:

OBREGON MARCELO JOAQUIN

AÑO:

2022

ESCALA:

S/E

HOJA:

52/73

NOMBRE:

CHAPA DE PROTECCIÓN
HORIZONTAL INFERIOR

REFERENCIA:

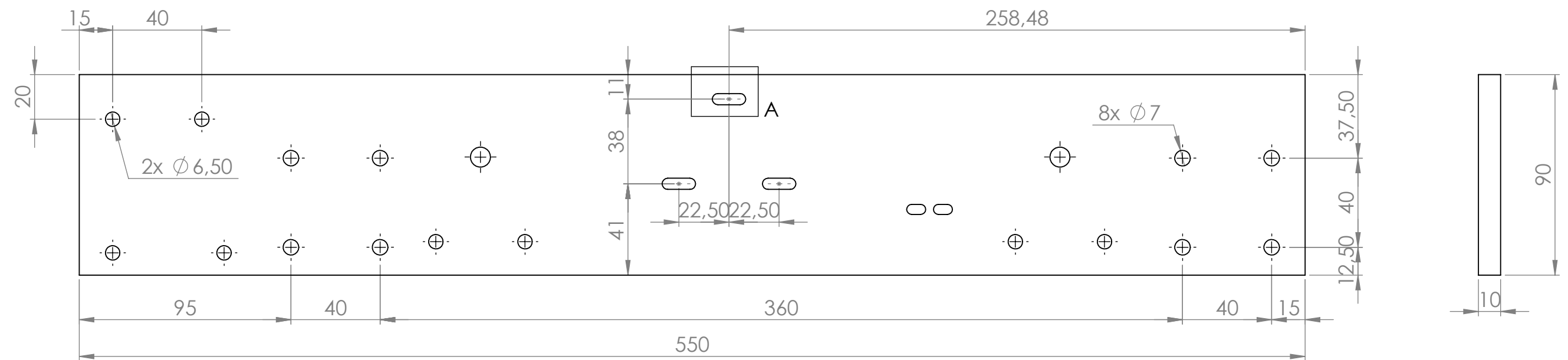
P28

MATERIAL:

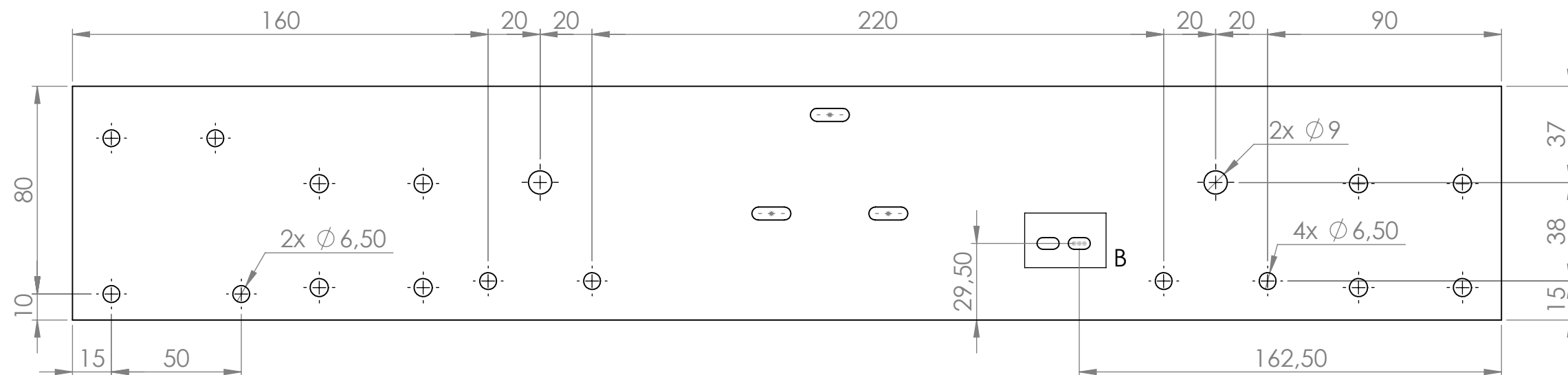
POLICARBONATO
COMPACTO

A4

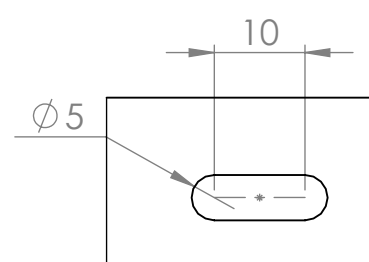
UNION A LOS SOPORTES DE EJES Y CHAPA DE PROT



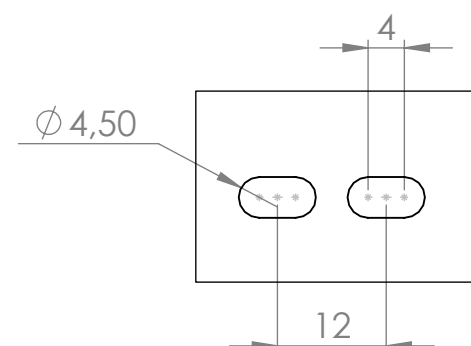
UNION AL SOPORTE DE CARRO Y CADENA PORTACABLES



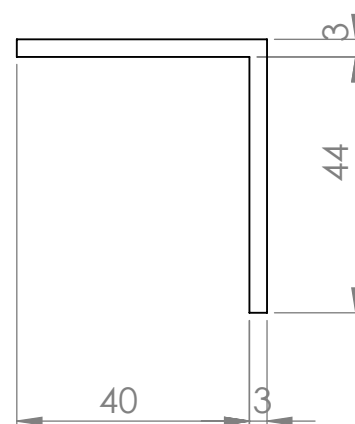
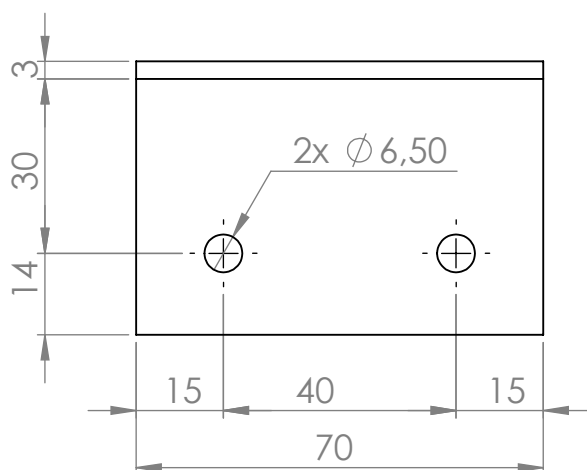
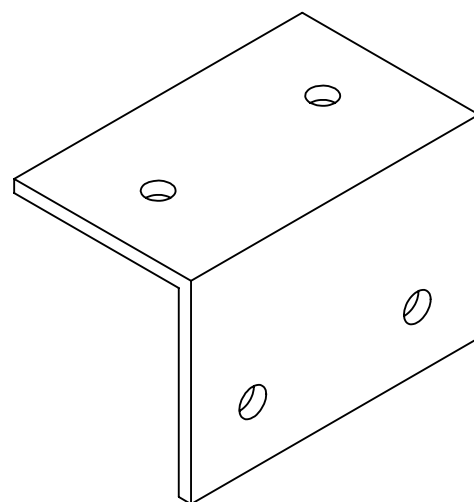
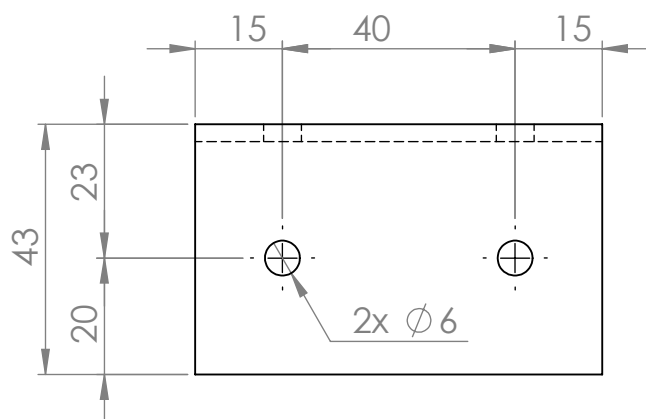
DETALLE A SUJECCIÓN DEL MOTOR



DETALLE B
SUJECCIÓN PLACA
DE ACCIONAMIENTO



TODAS LAS DIMENSIONES
EN mm



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

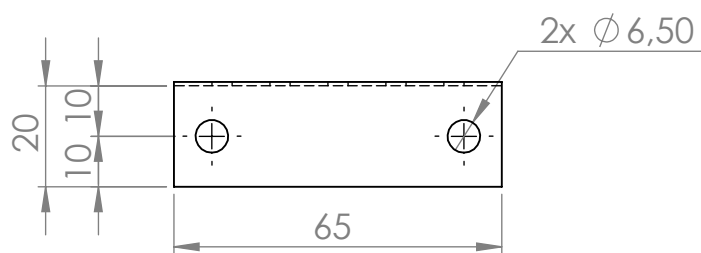
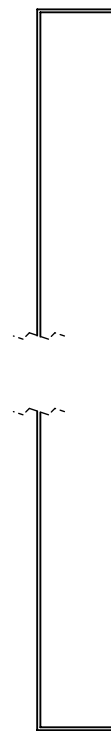
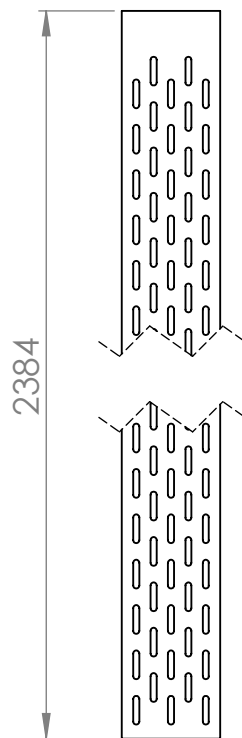
HOJA: 54/73

NOMBRE: SOPORTE DE CHAPA
DE PROT. HOR. INFERIOR

REFERENCIA: P30

MATERIAL: SAE 1010

A4



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

HOJA: 55/73

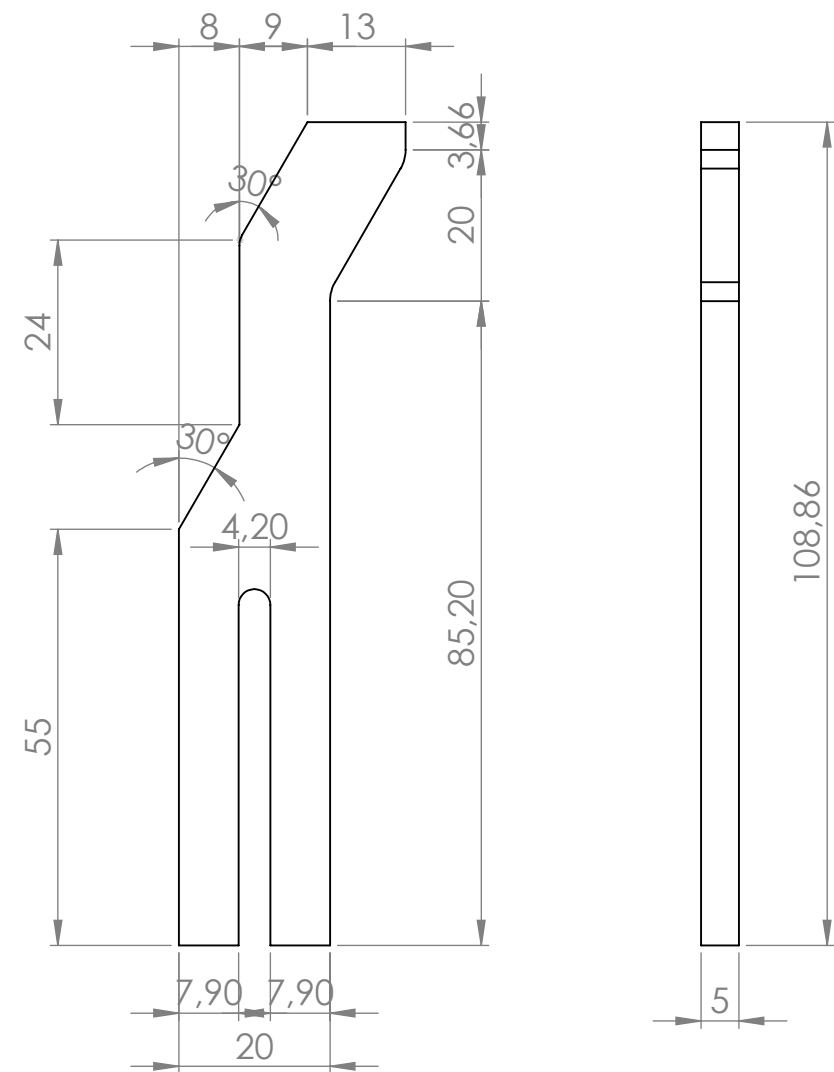
NOMBRE: SOPORTE DE CADENA
PORTACABLES HORIZONTAL

REFERENCIA: P31

MATERIAL: CHAPA DE
ACERO GALVANIZADO

A4

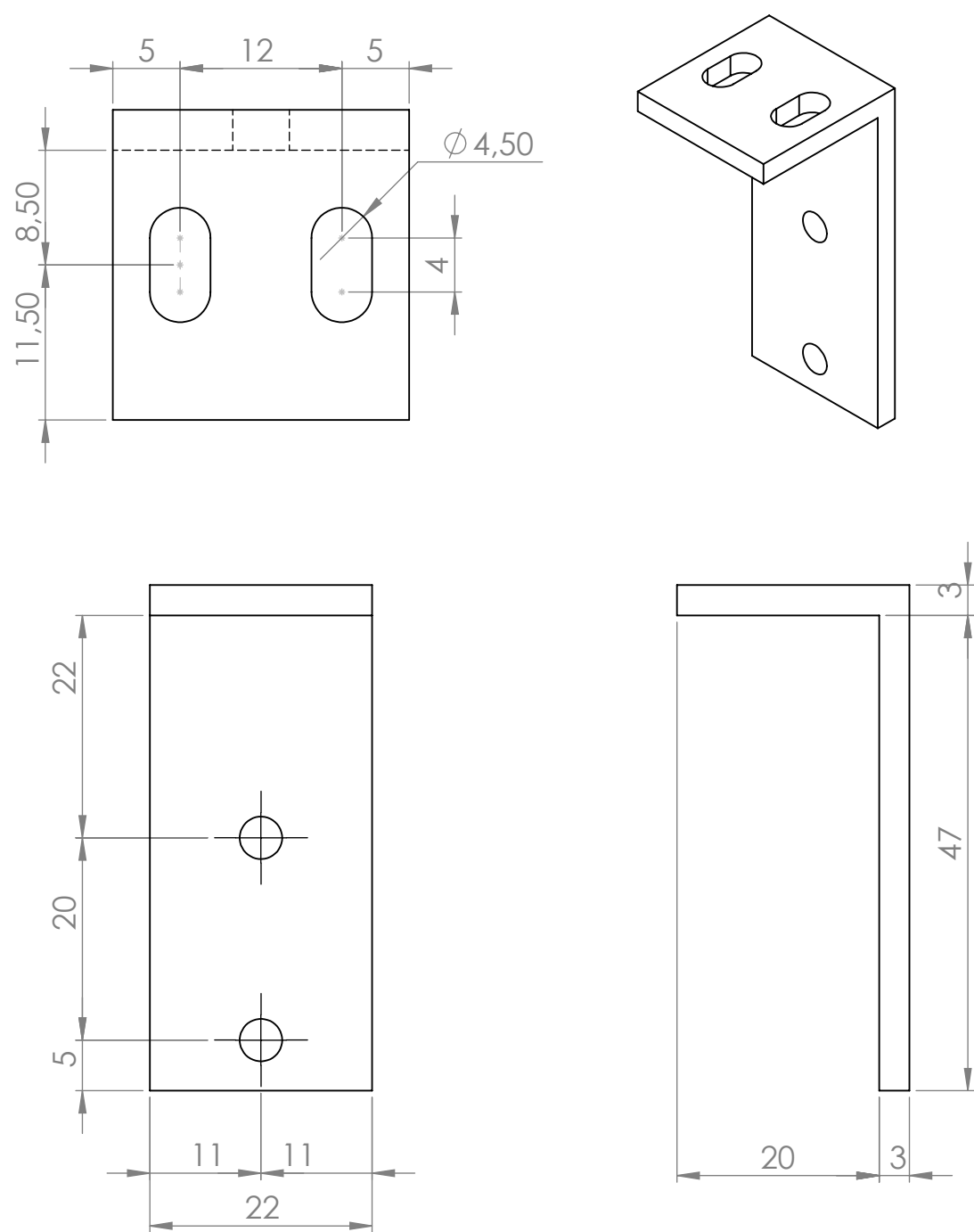
PLACA DE ACCIONAMIENTO
DE SENSOR Y FIN DE CARRERA



MATERIAL: PVC

UNION ENTRE PLACA Y SOPORTE:
2x M4 X 15mm

SOPORTE DE PLACA DE ACCIONAMIENTO
(PARA EJE HORIZONTAL)

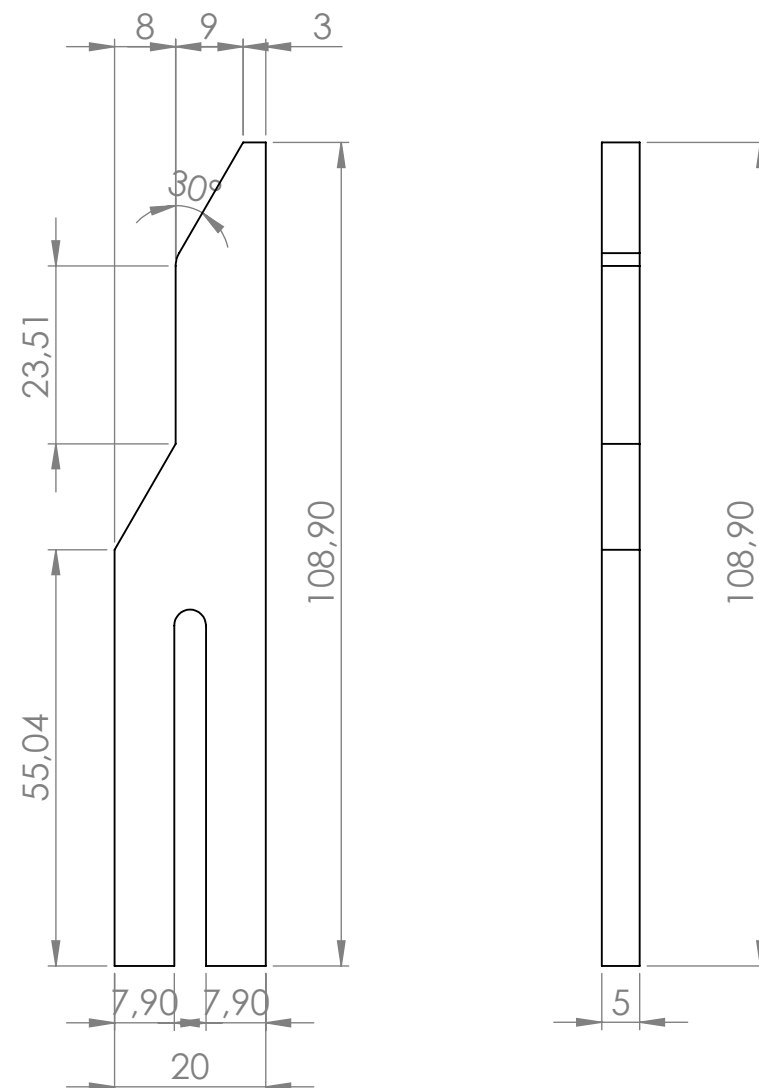


MATERIAL: SAE 1010

TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: ACCIONAMIENTO DE SENSOR Y FIN DE CARRERA	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P32	A3
AÑO: 2022	ESCALA: S/E	HOJA: 56/73	MATERIAL: SEGUN INDICACIÓN	

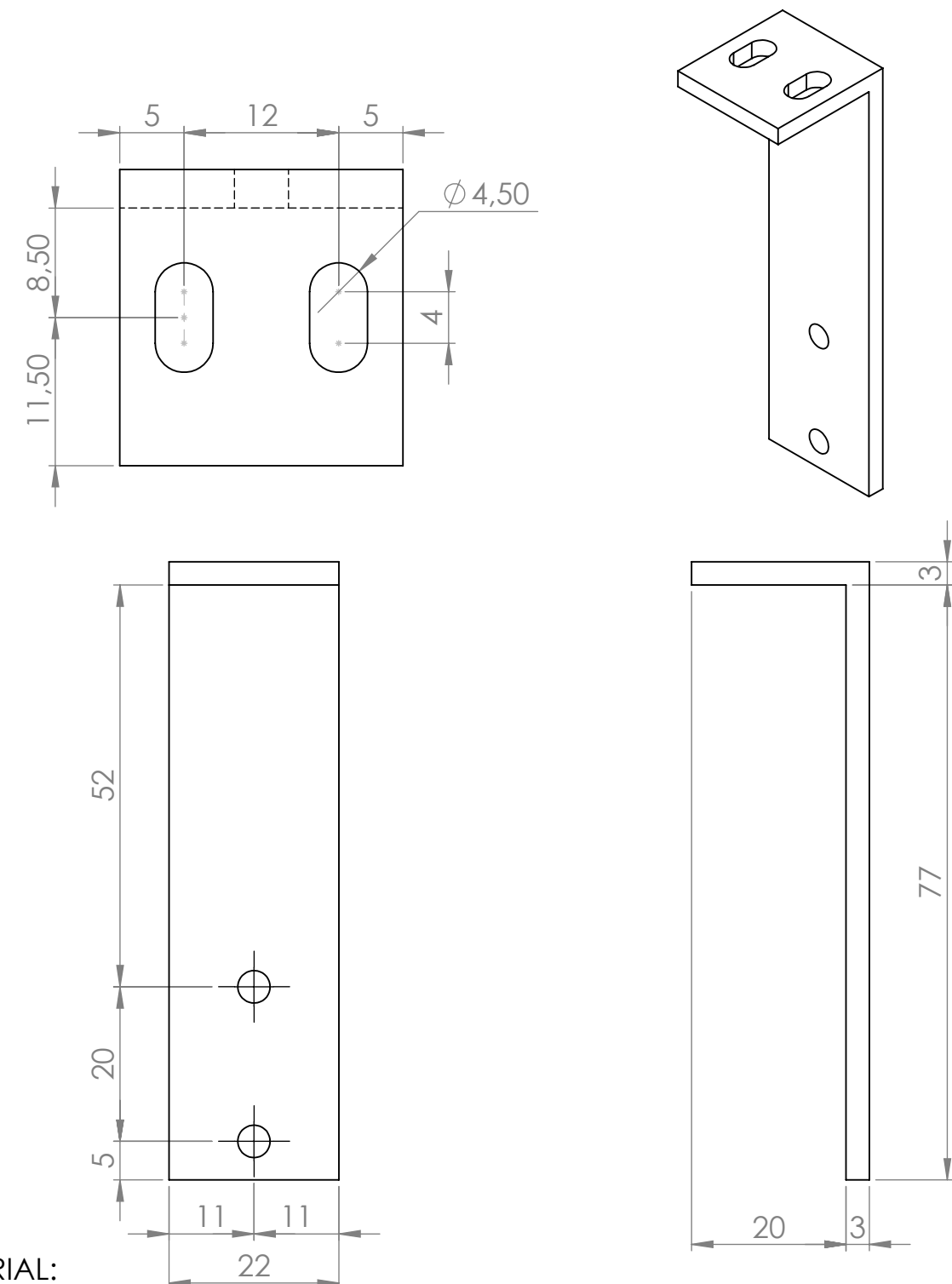
PLACA DE ACCIONAMIENTO DE FIN DE CARRERA



MATERIAL: PVC

UNION ENTRE PLACA Y SOPORTE:
2x M4 X 15mm

SOPORTE DE PLACA DE ACCIONAMIENTO (PARA EJE HORIZONTAL)

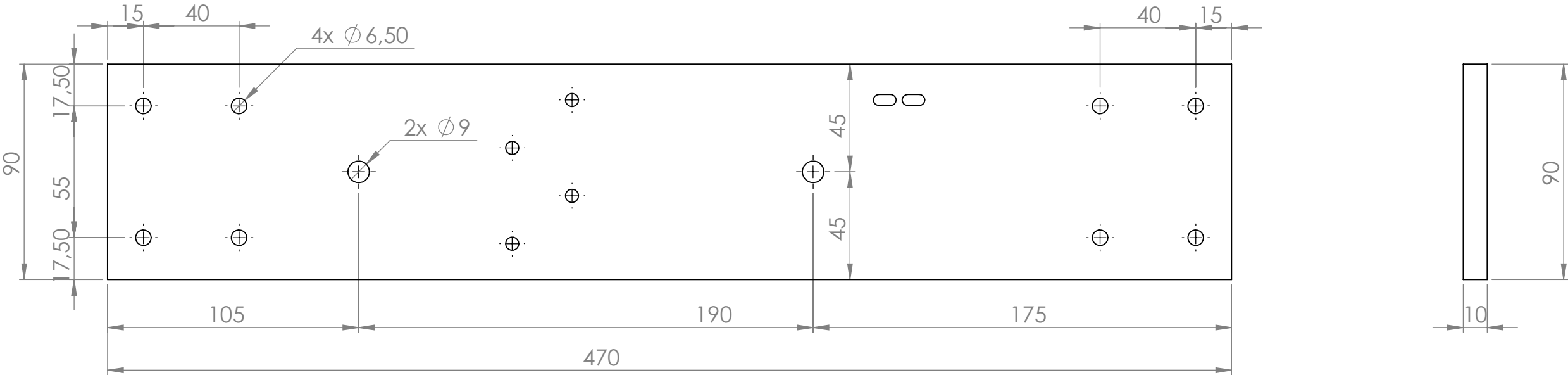


MATERIAL:
SAE 1010

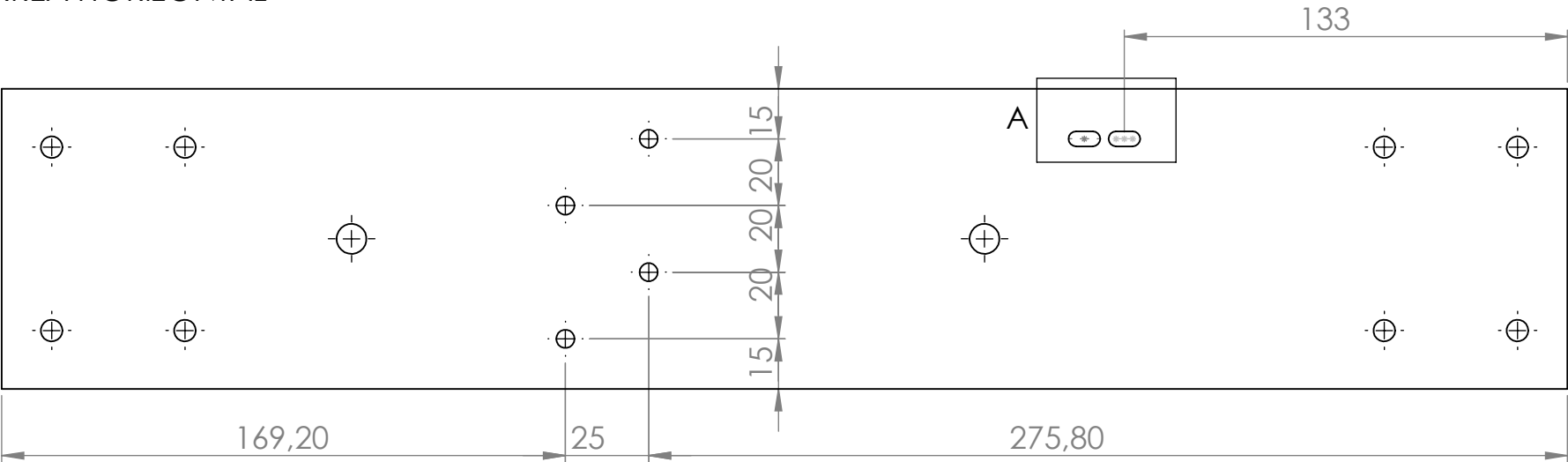
TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: ACCIONAMIENTO DE FIN DE CARRERA	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P33	A3
AÑO: 2022	ESCALA: S/E	HOJA: 57/73	MATERIAL: SEGUN INDICACIÓN	

UNION A BUJES Y EJE VERTICAL

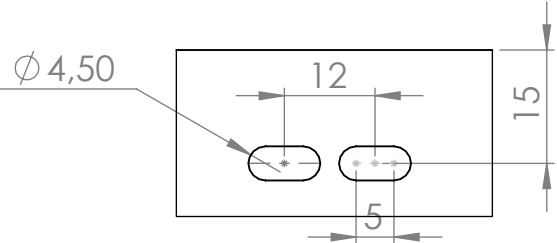


UNION A CORREA HORIZONTAL



TODAS LAS DIMENSIONES
EN mm

DETALLE A
UNION A SENSOR
Y FIN DE CARRERA



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

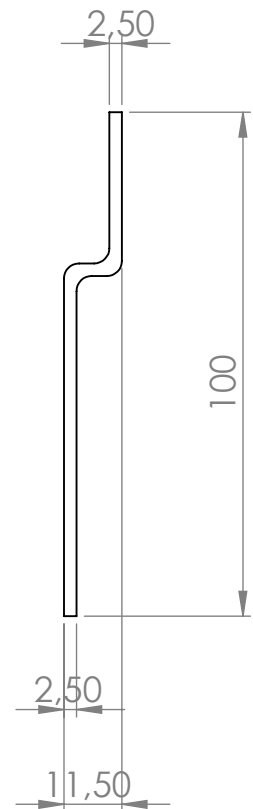
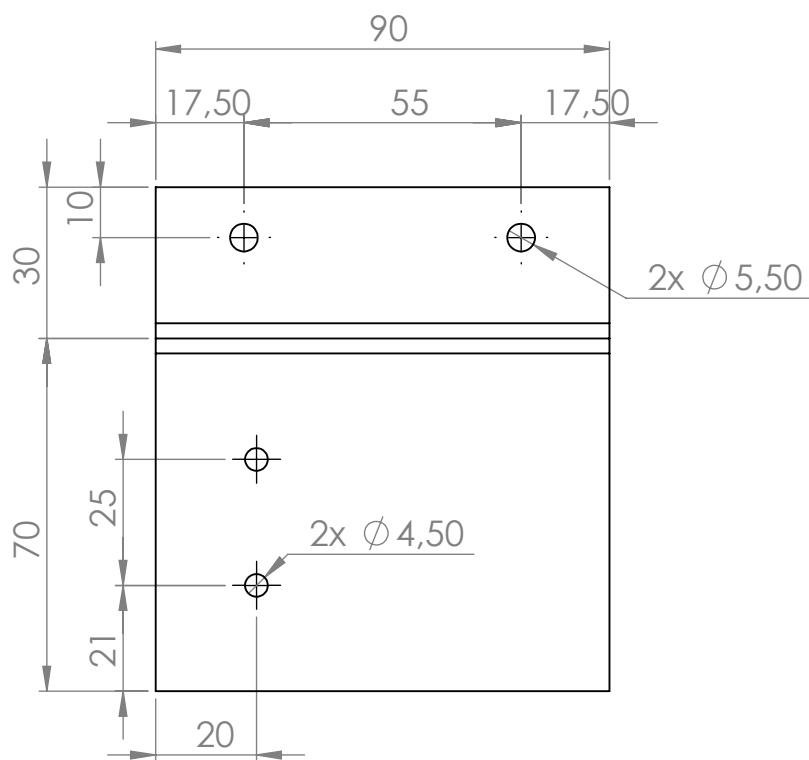
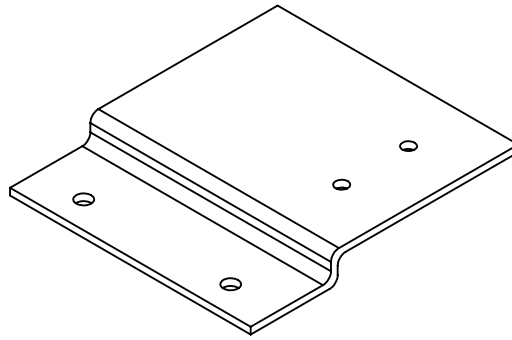
AÑO: 2022 ESCALA: S/E HOJA: 58/73

NOMBRE: SOPORTE CENTRAL
EJE HOR. INFERIOR

REFERENCIA: P34

MATERIAL: ALUMINIO

A3



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
59/73

NOMBRE:

SUJECCIÓN CADENA
PORTACABLES HOR.

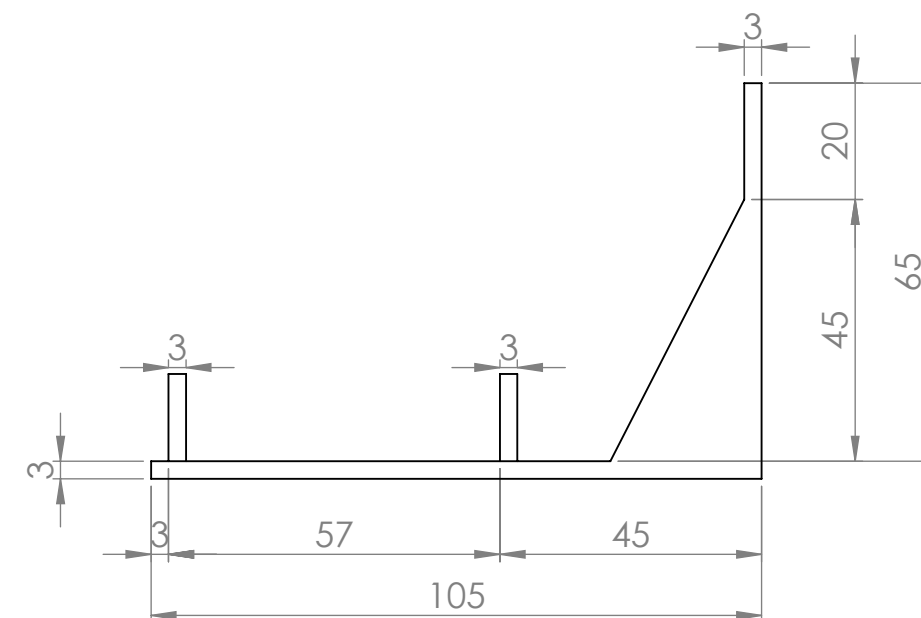
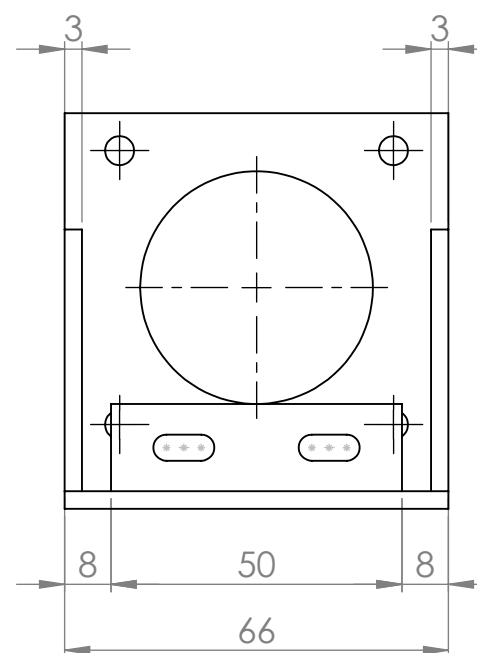
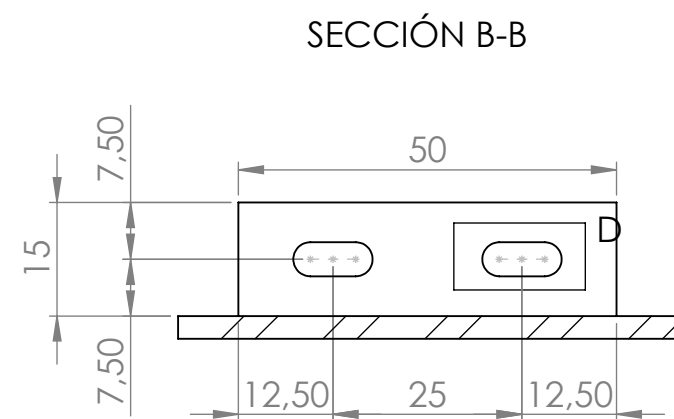
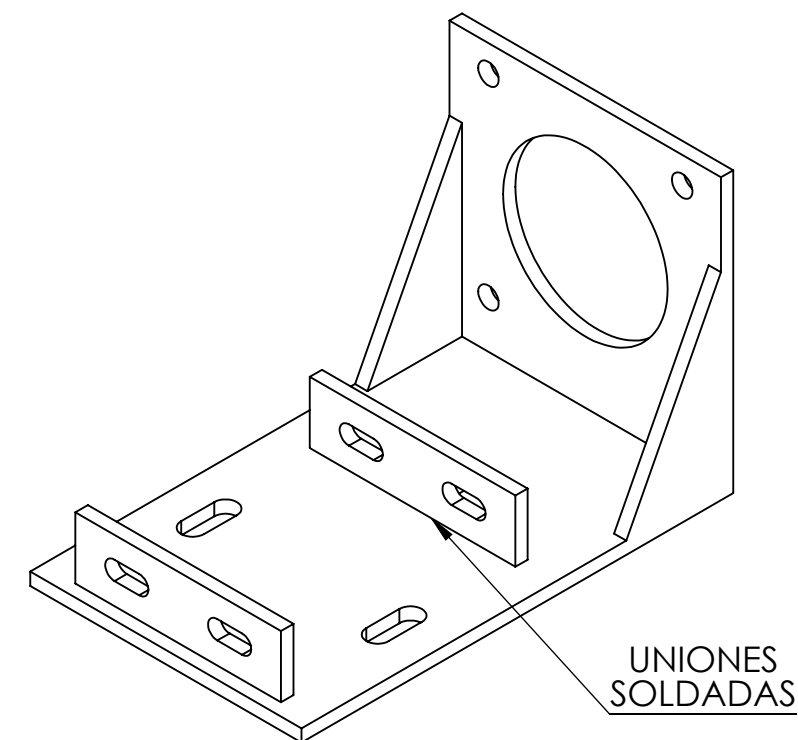
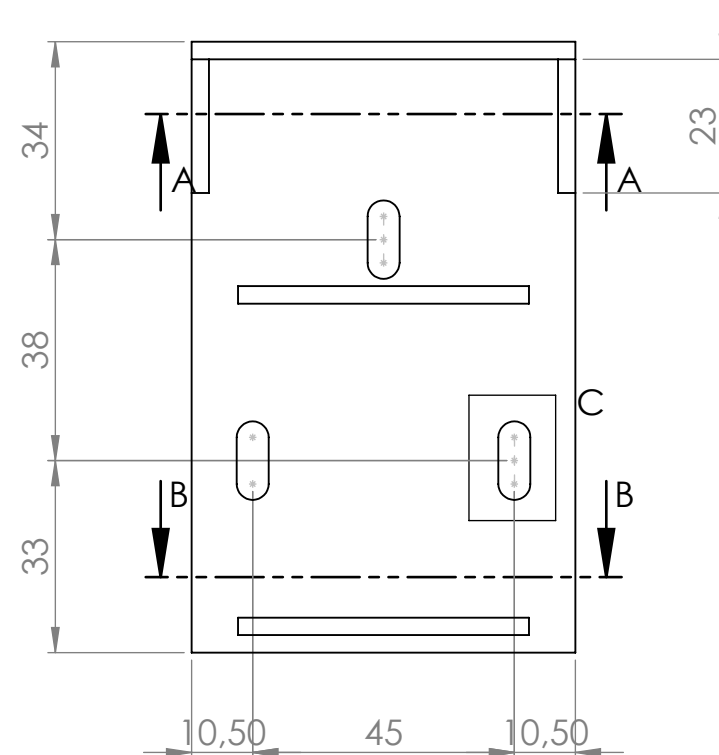
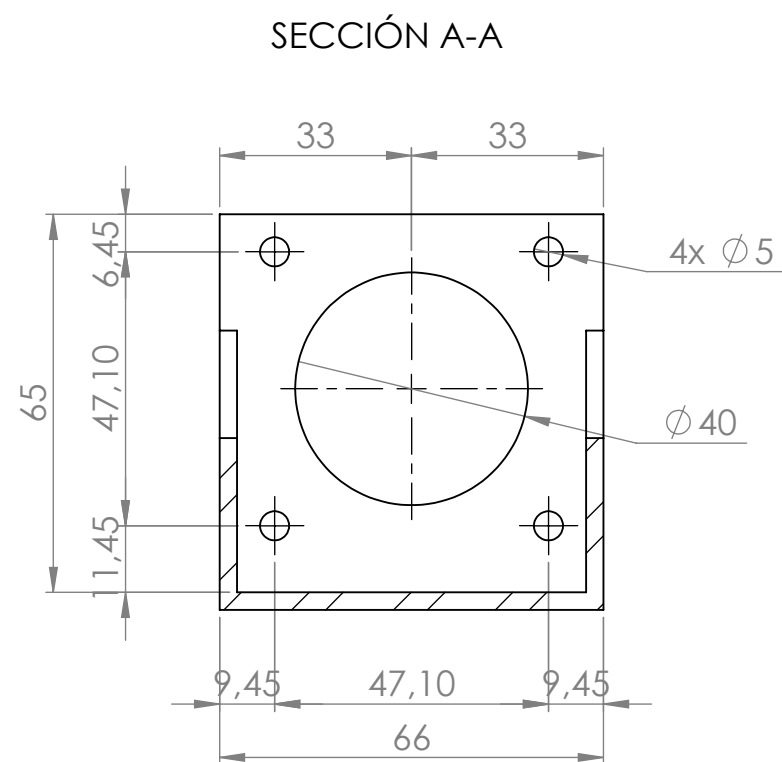
REFERENCIA:

P35

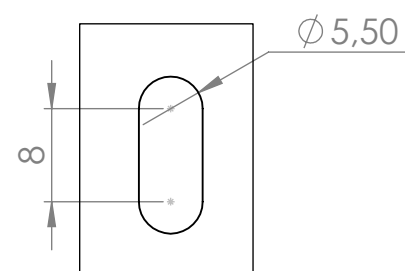
MATERIAL:

SAE 1010

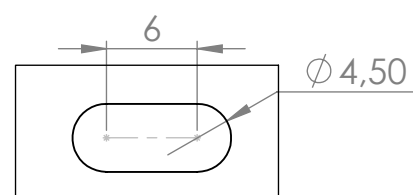
A4



DETALLE C

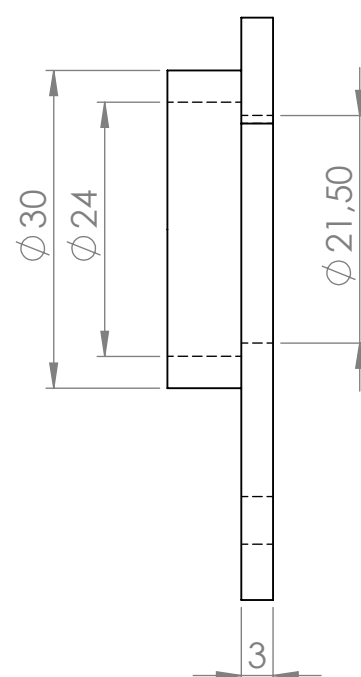
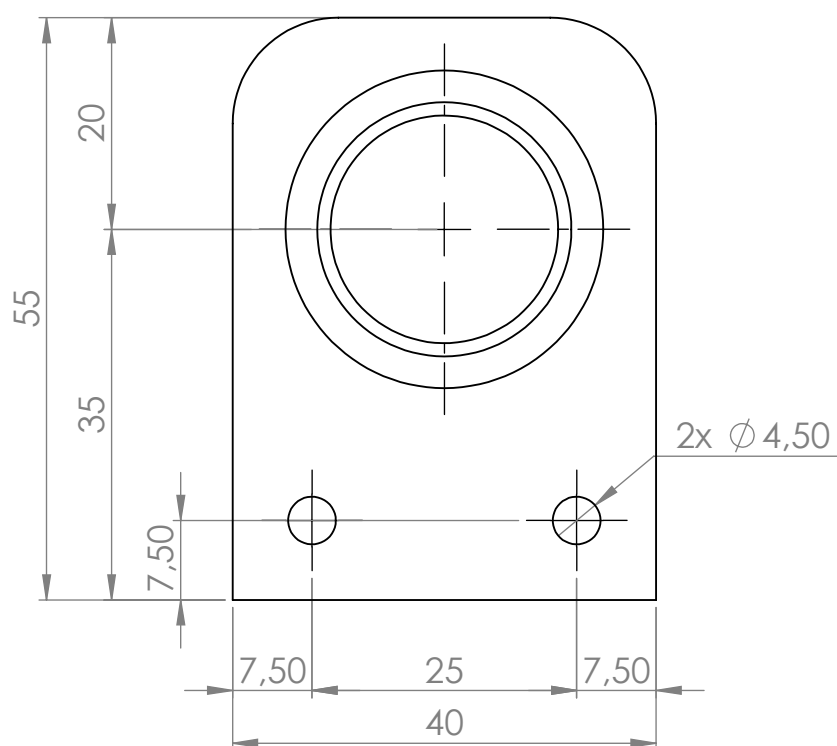
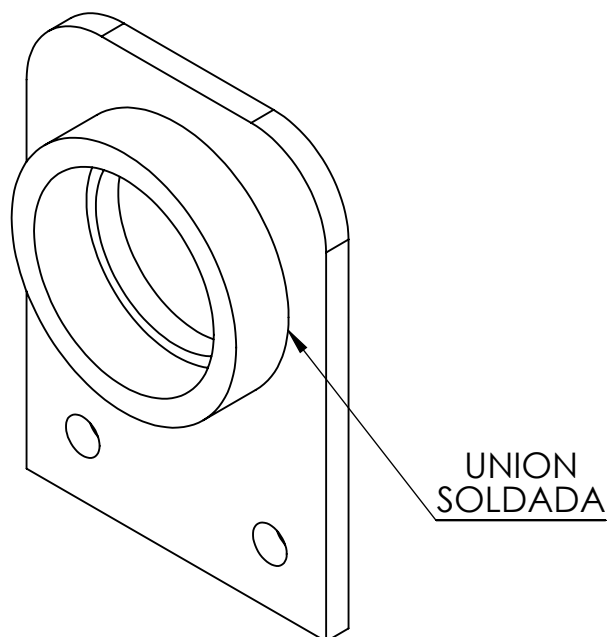


DETALLE D



TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: SOPORTE MOTOR HORIZONTAL	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P36	A3
AÑO: 2022	ESCALA: S/E	HOJA: 60/73	MATERIAL: SAE 1010	



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
61/73

NOMBRE:

SOPORTE POLEA
HORIZONTAL

REFERENCIA:

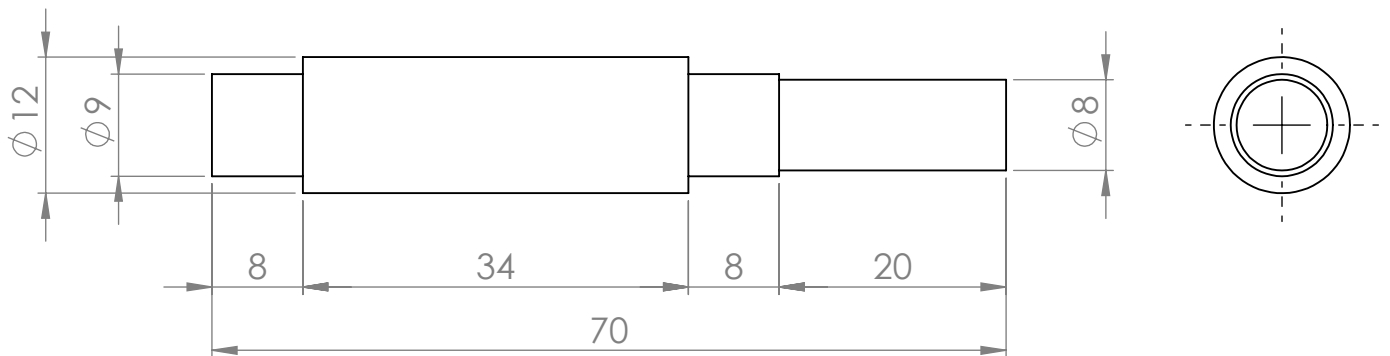
P37

MATERIAL:

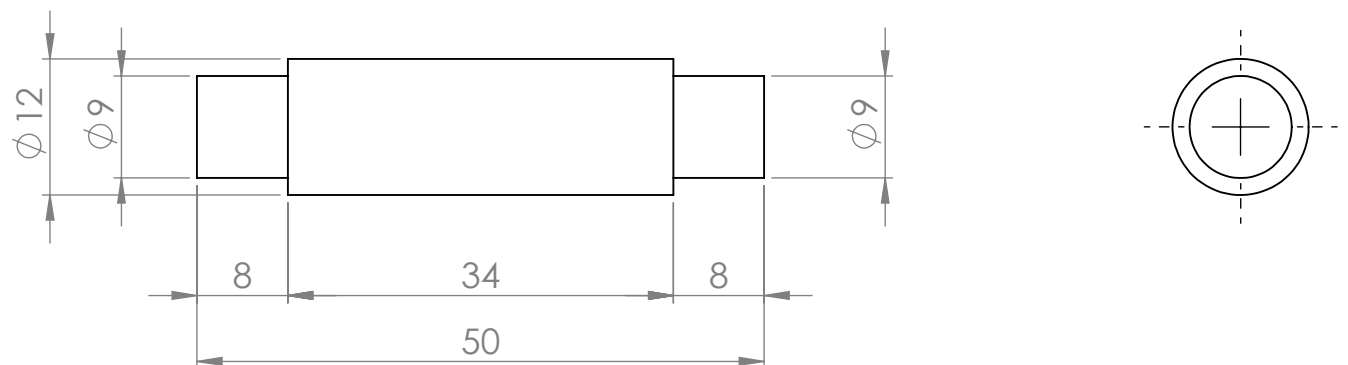
SAE 1010

A4

EJE POLEA MOTORA HORIZONTAL

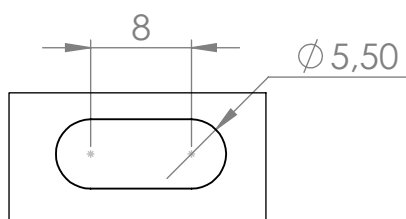
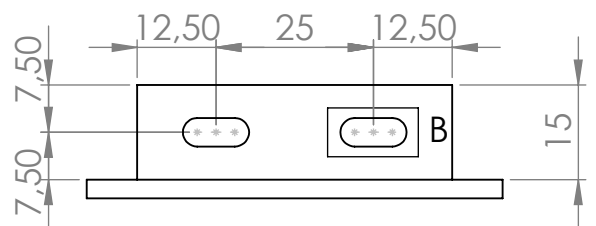
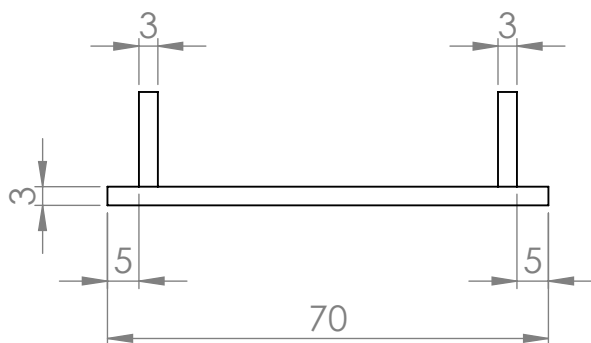
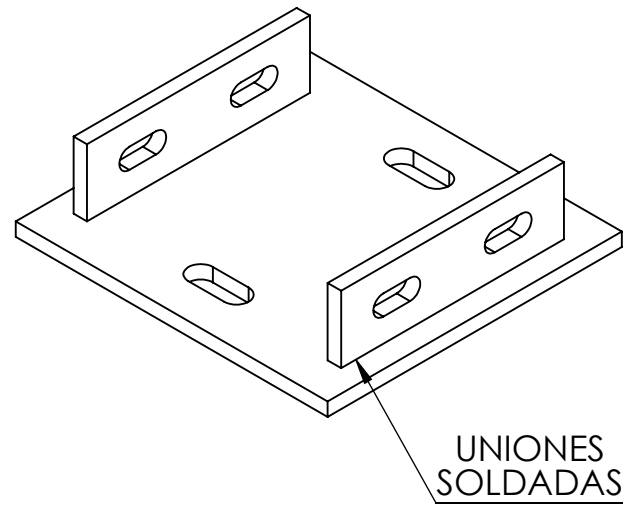
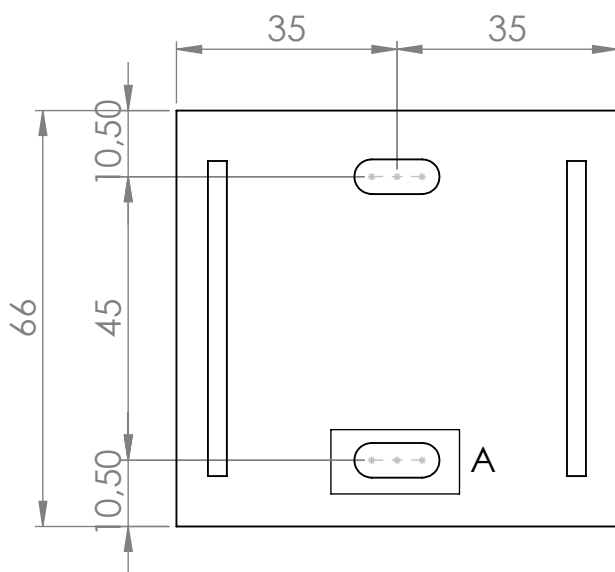


EJE POLEA DE RETORNO HORIZONTAL

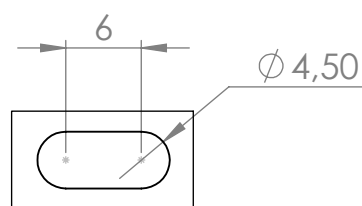


TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: EJES POLEAS HORIZONTALES	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P38 Y P39	A4
AÑO: 2022	ESCALA: S/E	HOJA: 62/73		



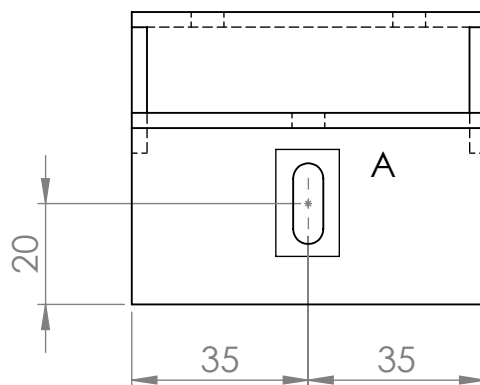
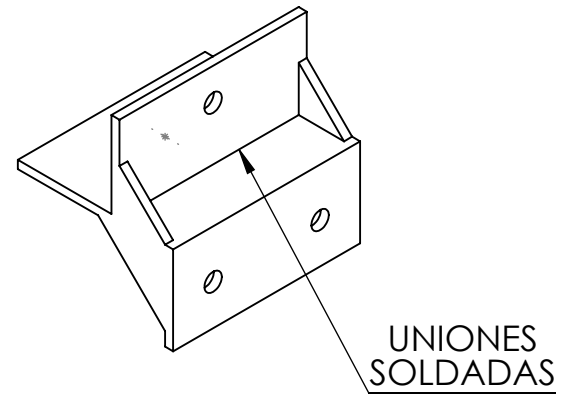
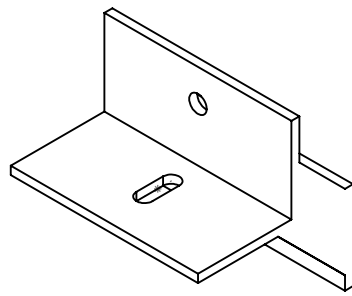
DETALLE A



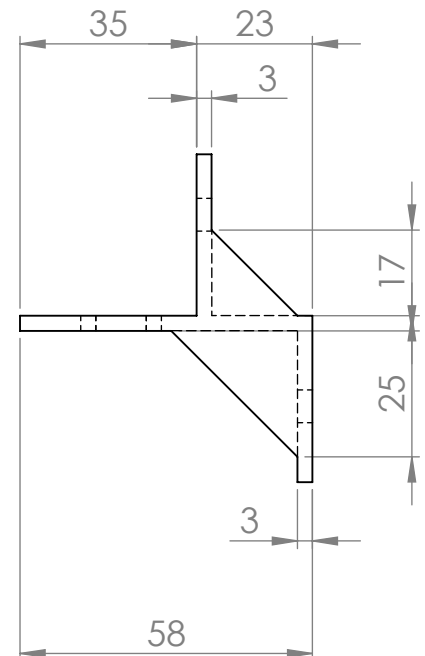
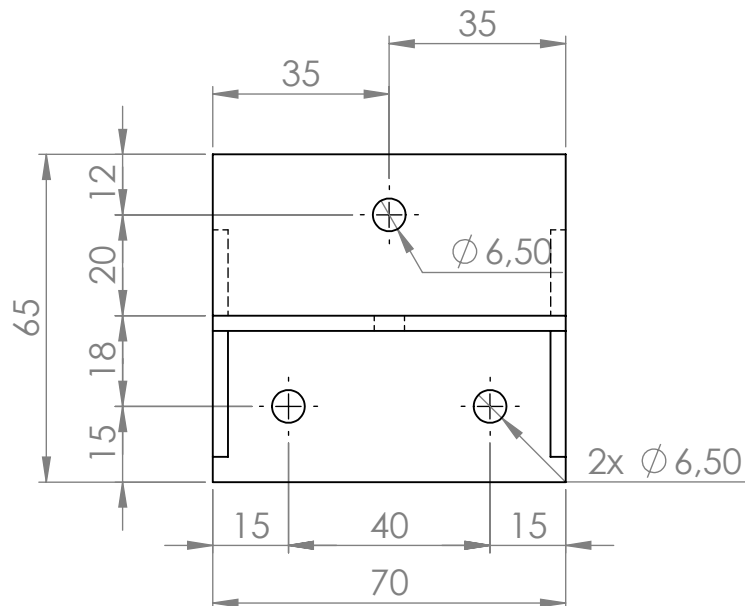
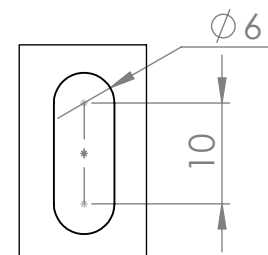
DETALLE B

TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: SOPORTE POLEA DE RETORNO HORIZONTAL	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P40	A4
AÑO: 2022	ESCALA: S/E	HOJA: 63/73	MATERIAL: SAE 1010	



DETALLE A



TODAS LAS DIMENSIONES EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
64/73

NOMBRE:

SOPORTE
CARRO INFERIOR

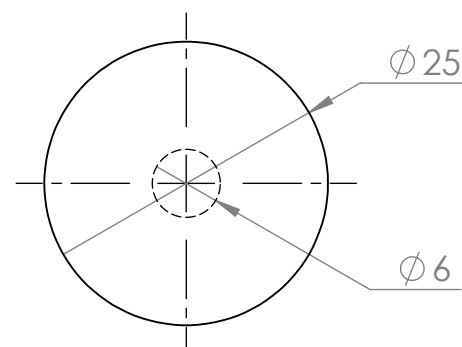
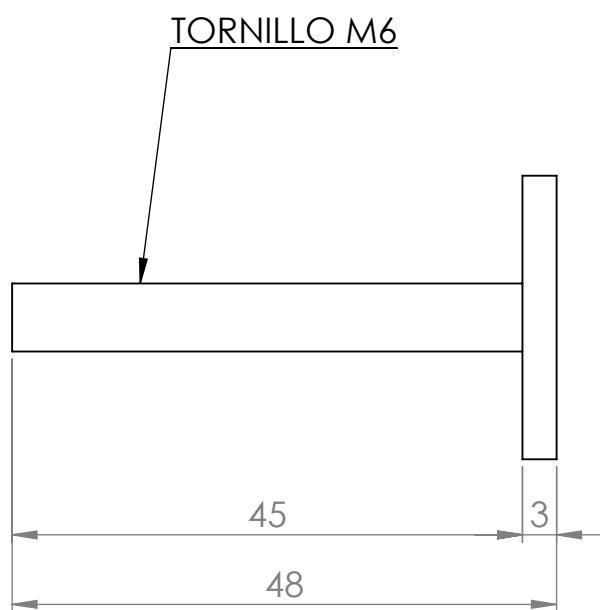
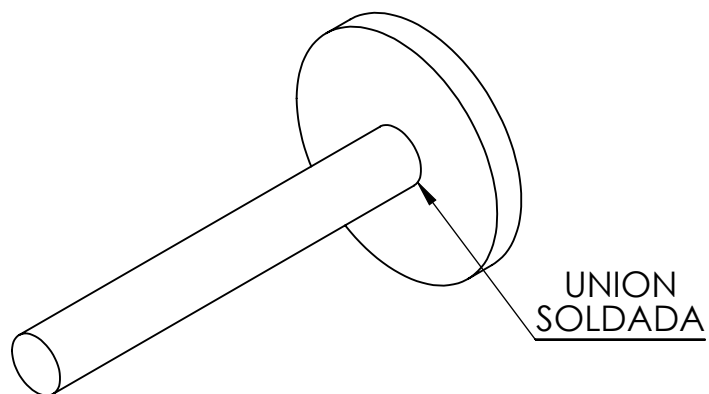
REFERENCIA:

P41

MATERIAL:

SAE 1010

A4



UNIVERSIDAD NACIONAL
DEL NORDESTE

FACULTAD DE INGENIERIA

DISEÑO:

OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

HOJA:
65/73

NOMBRE:

TRABA AXIAL
EJE HOR. INFERIOR

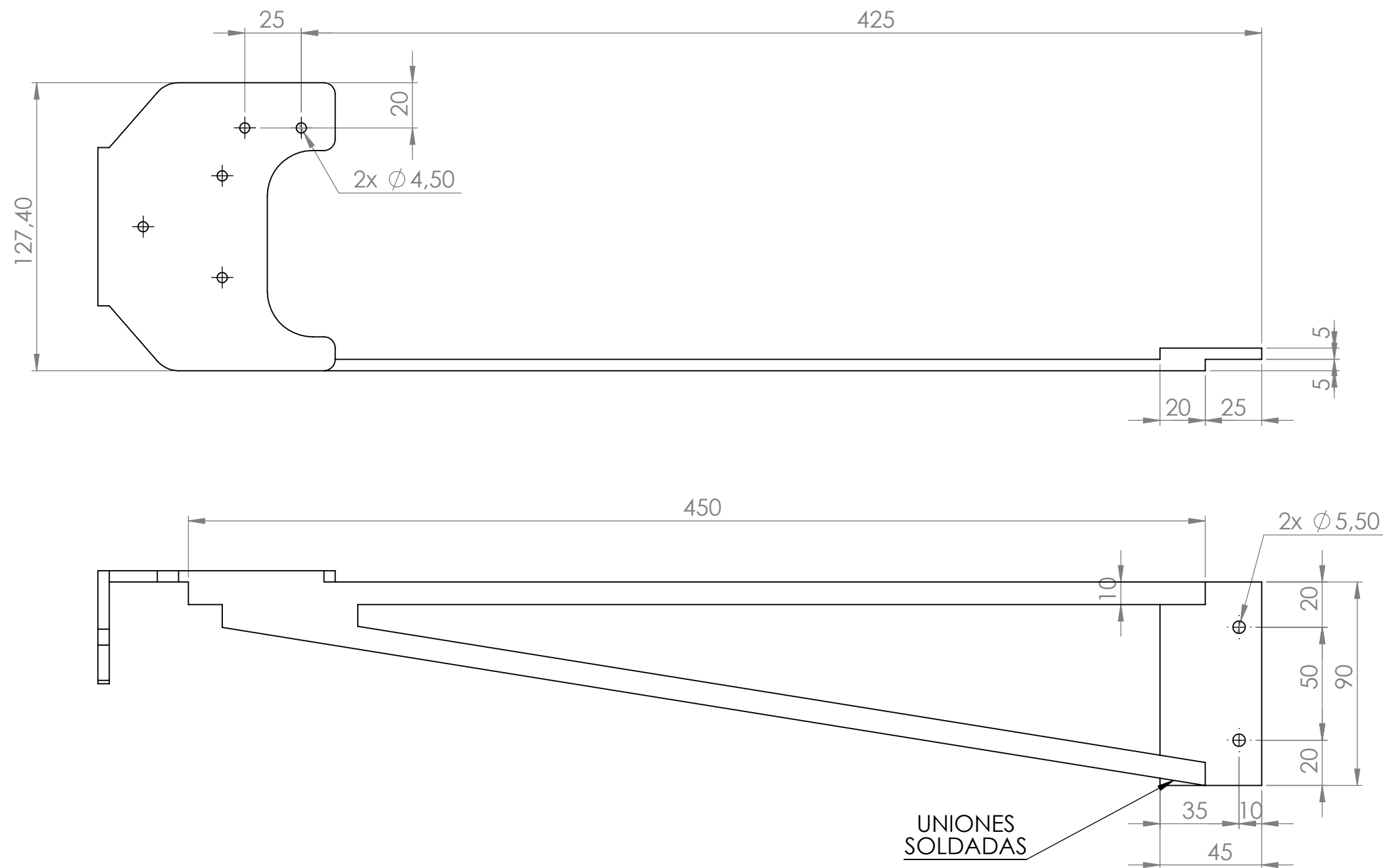
REFERENCIA:

P42

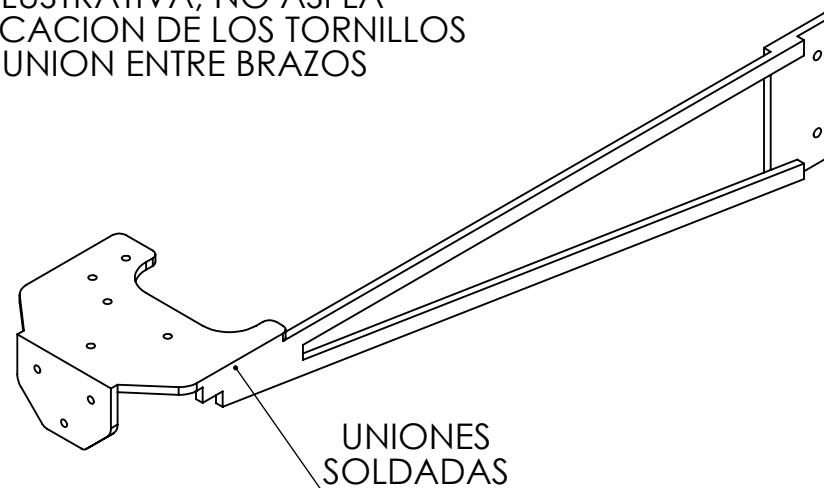
MATERIAL:

SAE 1010

A4

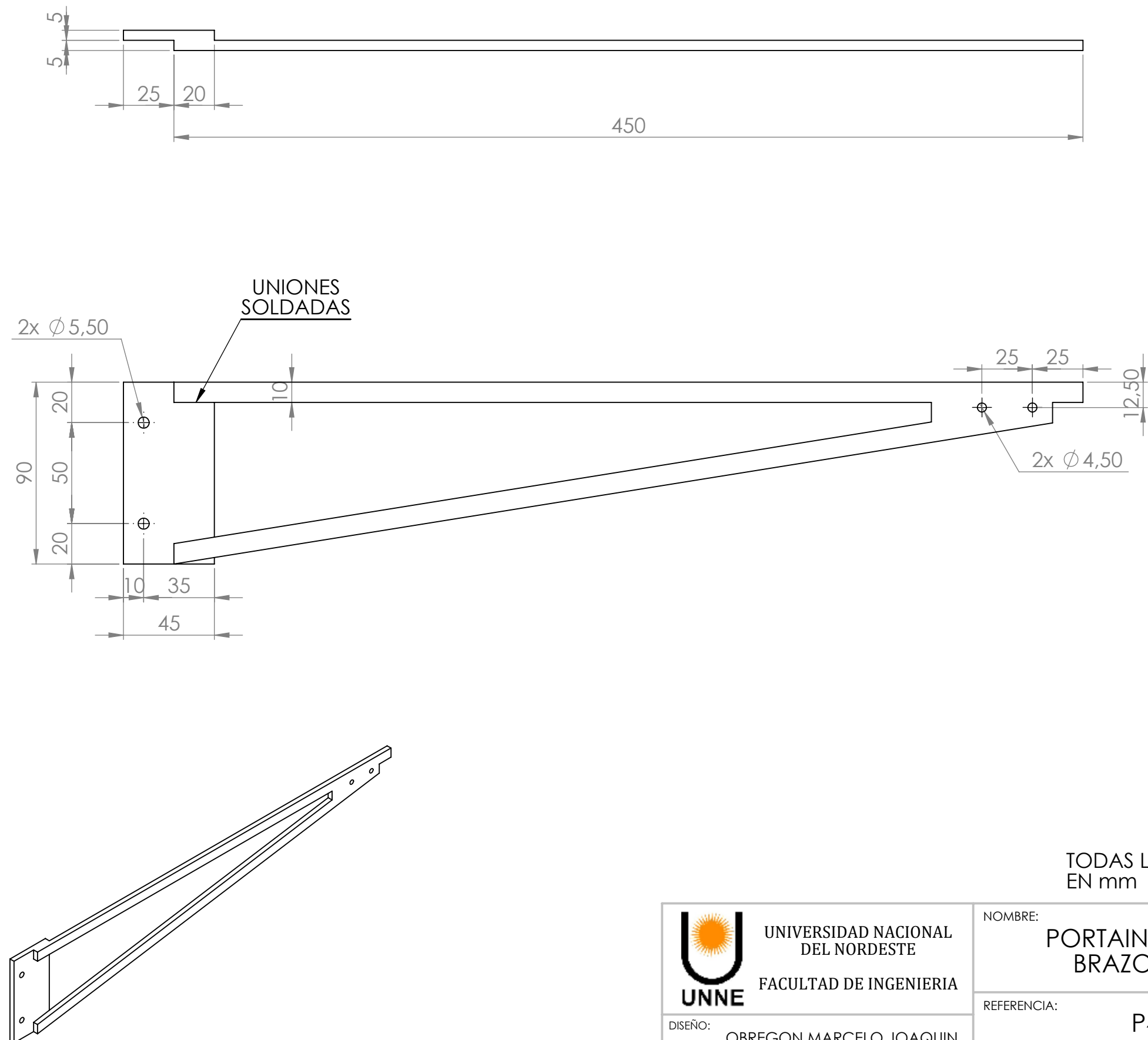


LA FORMA DEL EXTREMO
ES ILUSTRATIVA, NO ASI LA
UBICACION DE LOS TORNILLOS
DE UNION ENTRE BRAZOS



TODAS LAS DIMENSIONES
EN mm

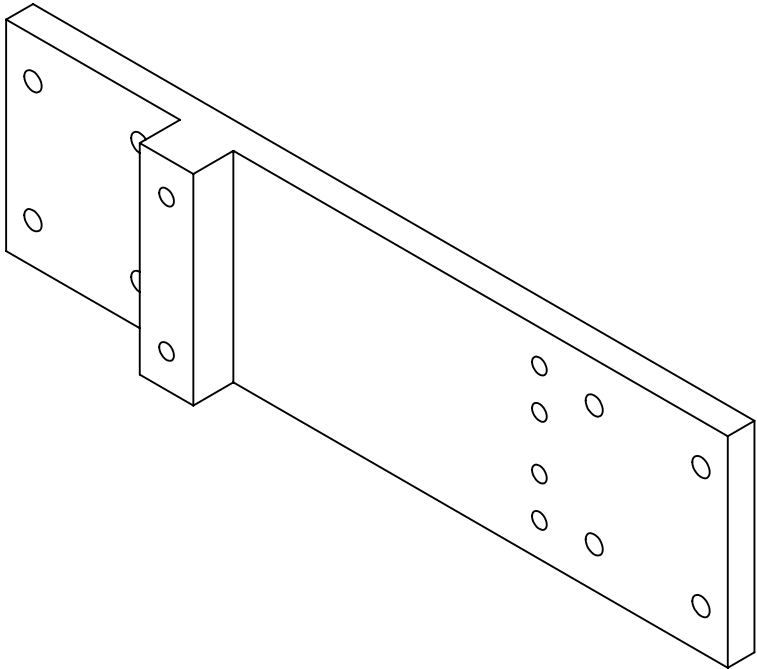
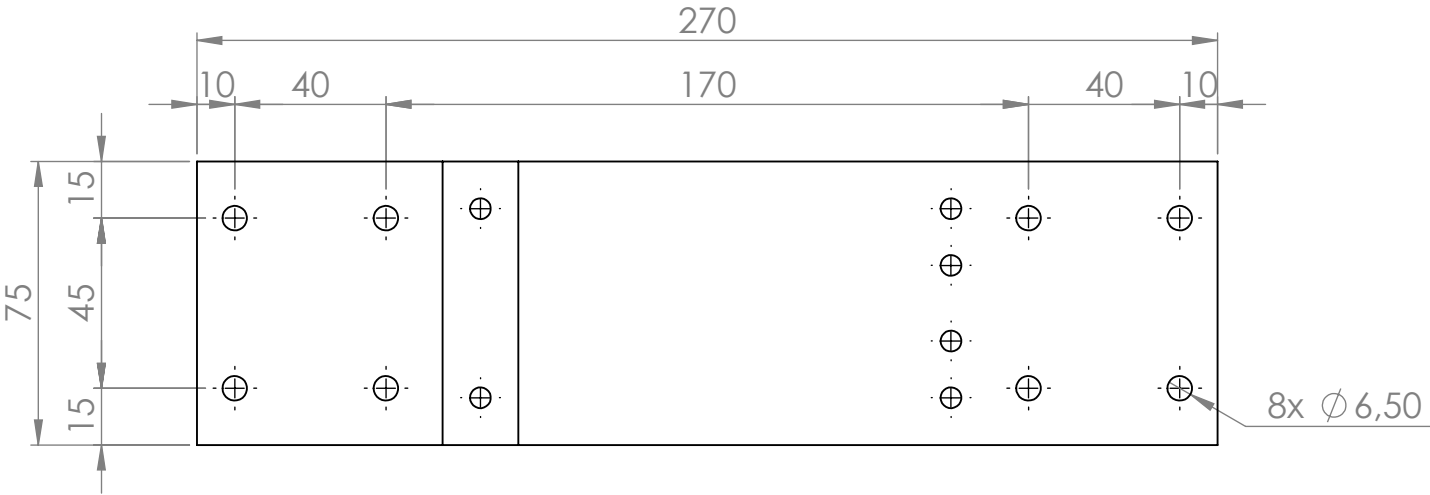
 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: PORTAINSTRUMENTOS - BRAZO DERECHO	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P43	A3
AÑO: 2022	ESCALA: S/E	HOJA: 66/73	MATERIAL: ALUMINIO	



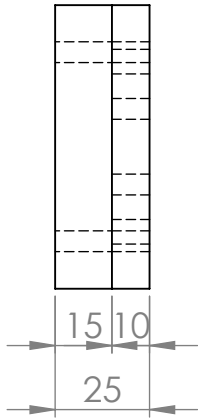
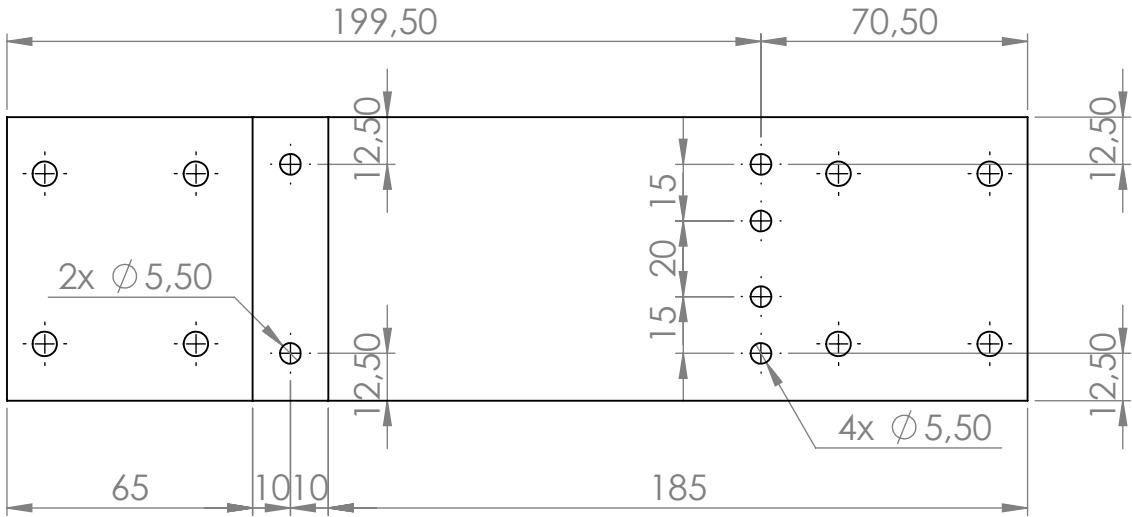
TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA	NOMBRE: PORTAINSTRUMENTOS - BRAZO IZQUIERDO	
DISEÑO: OBREGON MARCELO JOAQUIN		REFERENCIA: P44
AÑO: 2022	ESCALA: S/E	HOJA: 67/73
MATERIAL: ALUMINIO		A3

UNION A BUJES



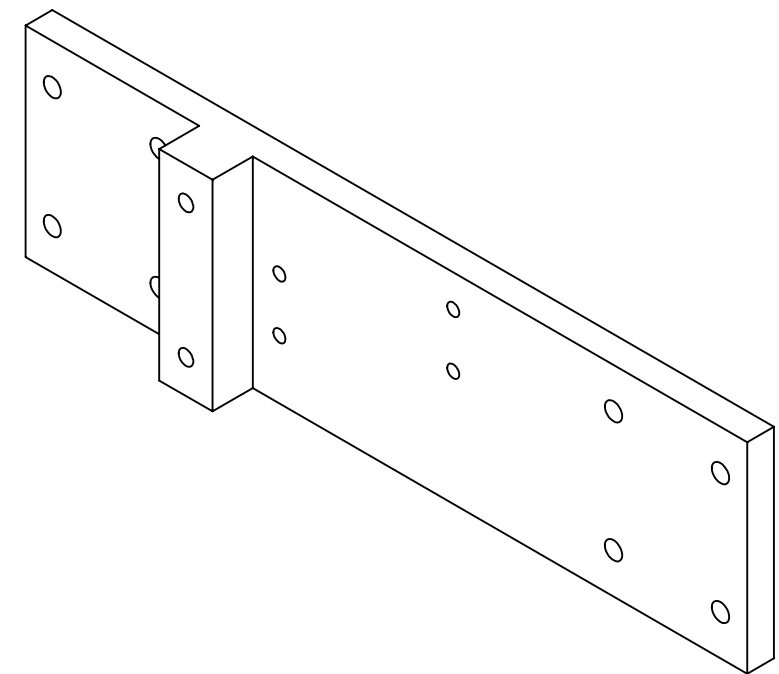
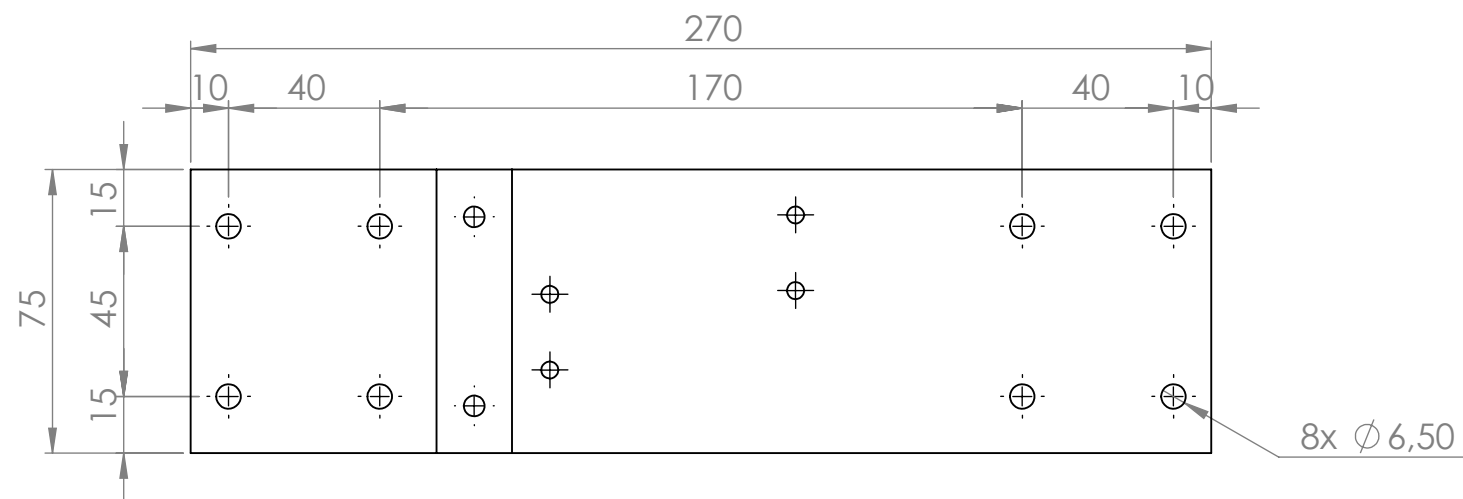
UNION A CORREA VERTICAL Y BISAGRAS



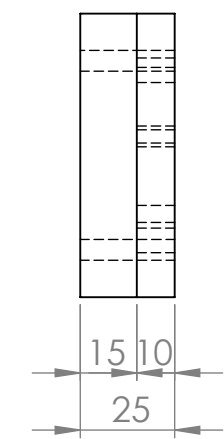
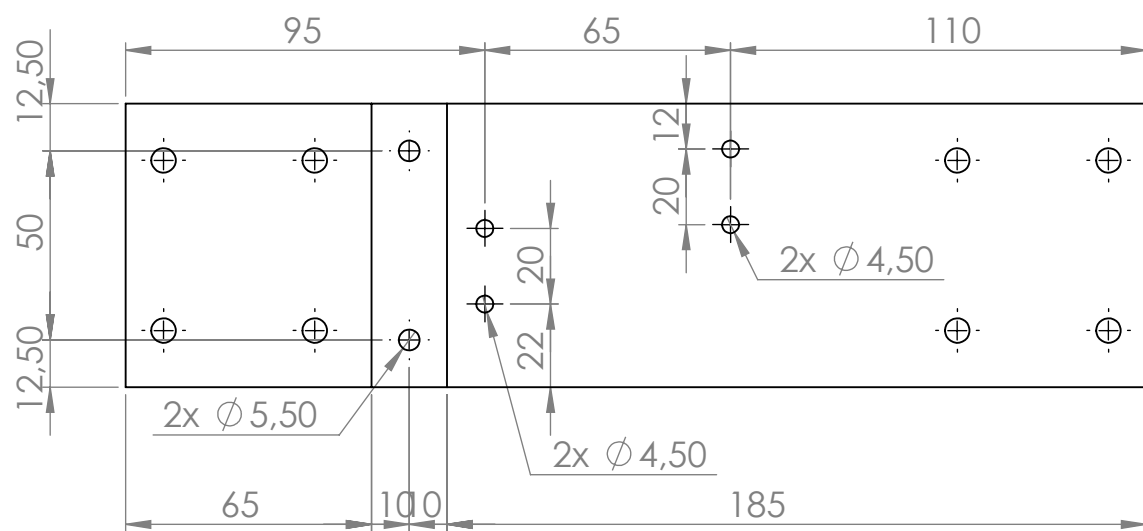
TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA	NOMBRE: PORTAINSTRUMENTO - PLACA SOPORTE DERECHA	
DISEÑO: OBREGON MARCELO JOAQUIN	REFERENCIA: P45	A3
AÑO: 2022	ESCALA: S/E	
HOJA: 68/73	MATERIAL: ALUMINIO	

UNION A BUJES

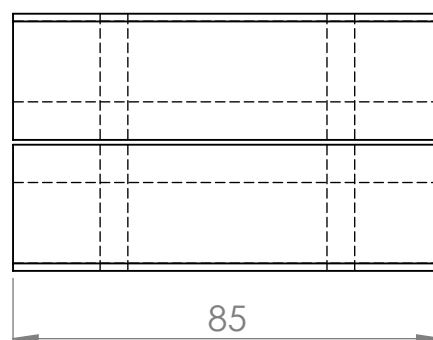
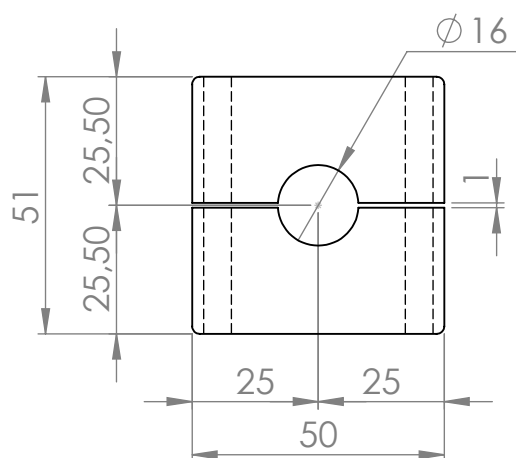
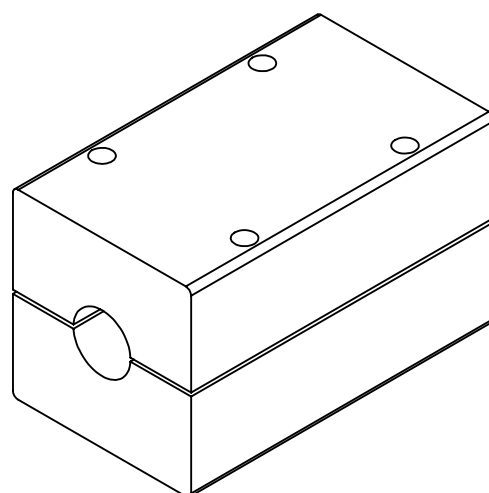
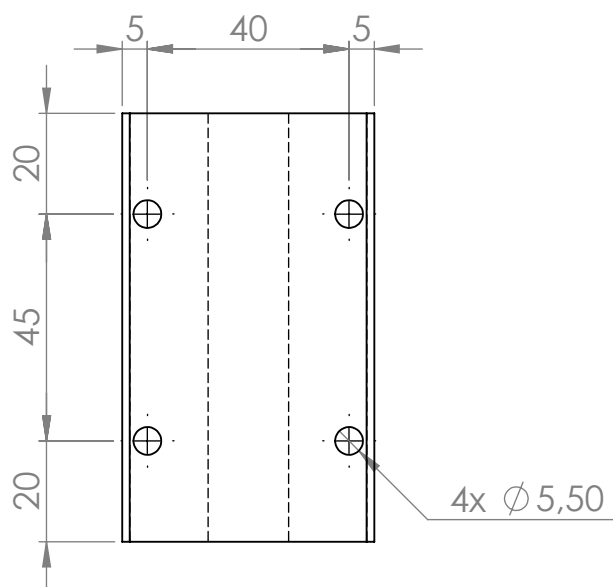


UNION A PLACAS DE ACCIONAMIENTO Y BISAGRAS



TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: PORTAINSTRUMENTO - PLACA SOPORTE IZQUIERDA	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P46	A3
AÑO: 2022	ESCALA: S/E	HOJA: 69/73	MATERIAL: ALUMINIO	



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

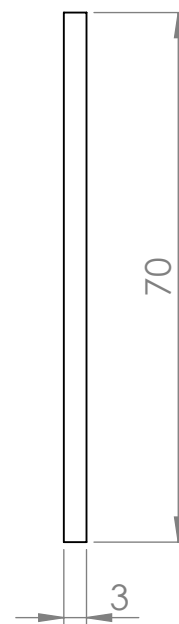
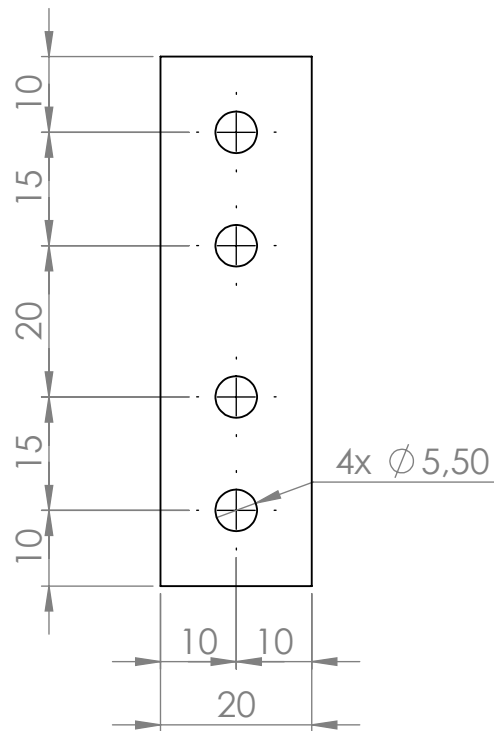
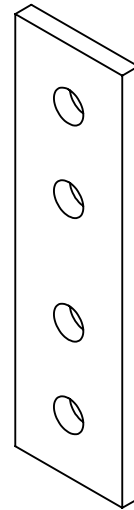
HOJA: 70/73

NOMBRE: BUJES PARA EJES
VERTICALES

REFERENCIA: P47

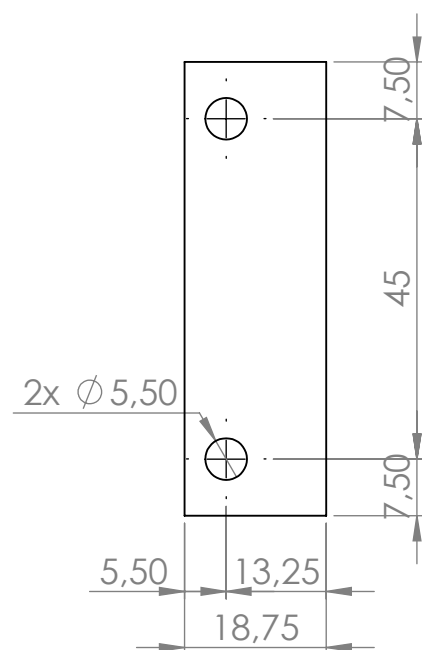
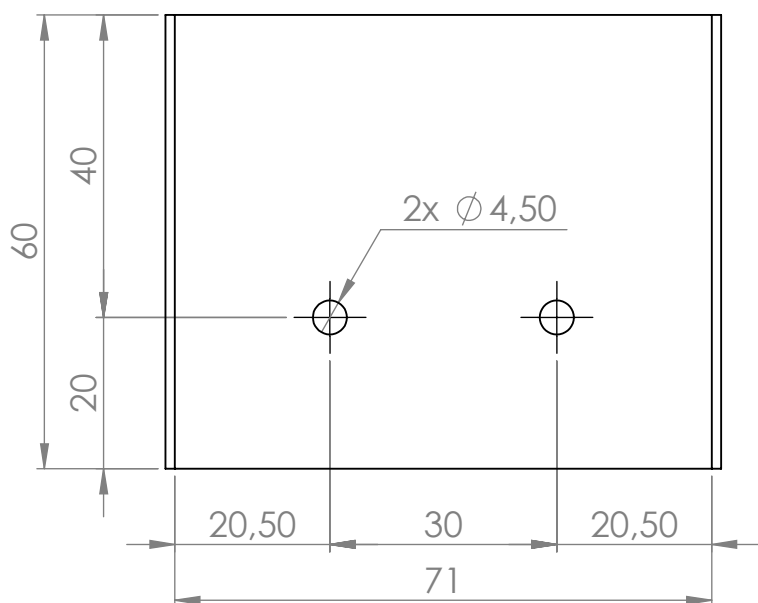
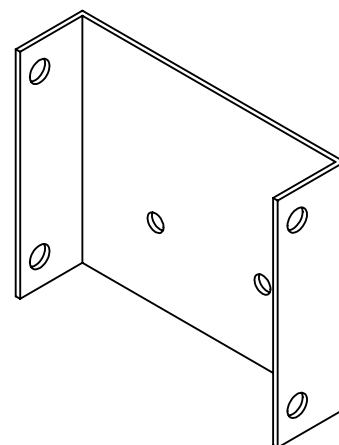
MATERIAL: GRILON

A4



TODAS LAS DIMENSIONES
EN mm

 UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERIA			NOMBRE: SUJECCIÓN DE CORREA VERTICAL	
DISEÑO: OBREGON MARCELO JOAQUIN			REFERENCIA: P48	A4
AÑO: 2022	ESCALA: S/E	HOJA: 71/73	MATERIAL: SAE 1010	



TODAS LAS DIMENSIONES
EN mm



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO:
OBREGON MARCELO JOAQUIN

AÑO:
2022

ESCALA:
S/E

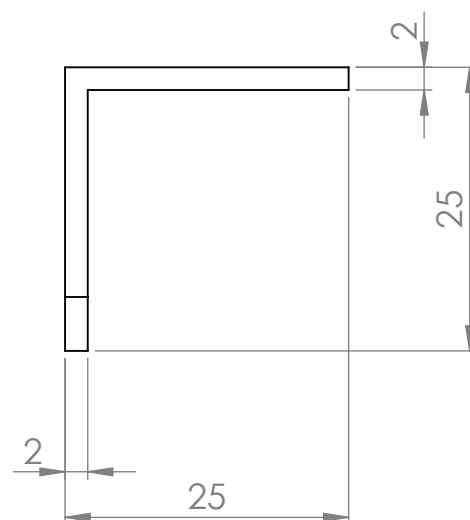
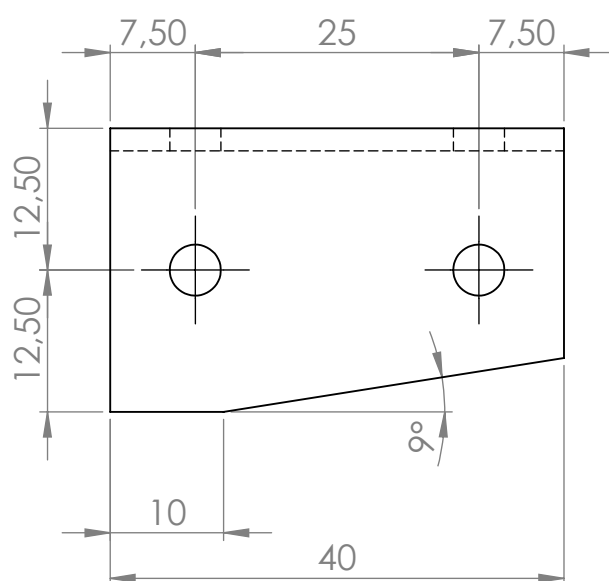
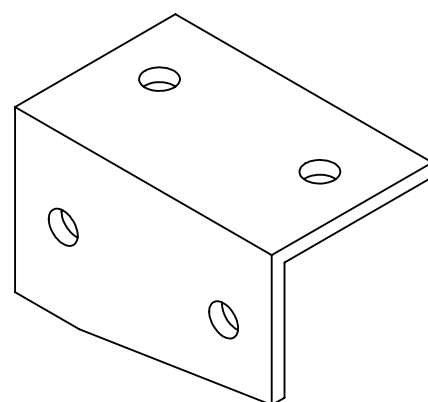
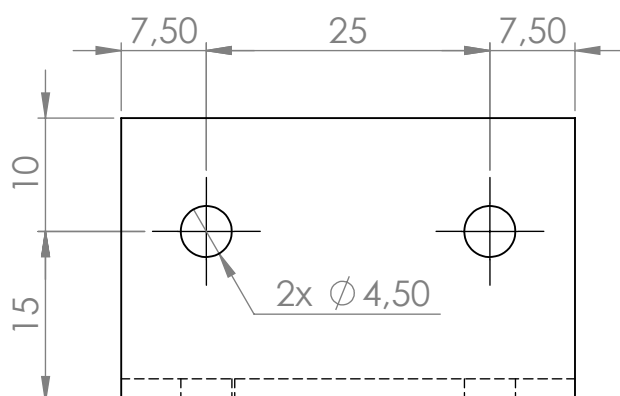
HOJA:
72/73

NOMBRE:
SUJECIÓN CADENA
PORTACABLES VERTICAL

REFERENCIA:
P49

MATERIAL:
CHAPA DE
ACERO GALVANIZADO

A4



UNIVERSIDAD NACIONAL
DEL NORDESTE
FACULTAD DE INGENIERIA

DISEÑO: OBREGON MARCELO JOAQUIN

AÑO: 2022

ESCALA: S/E

HOJA: 73/73

NOMBRE: PORTAINSTRUMENTO -
UNION ENTRE BRAZOS

REFERENCIA: P50

MATERIAL: SAE 1010

A4