Implementación Operativa de Prototipo Robótico

Paloma Barat Carnino

Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ingeniería Mecánica
Rosario, Argentina
Año 2023

Implementación Operativa de Prototipo Robótico

Paloma Barat Carnino

Proyecto Final de Ingeniería Mecánica presentado como requisito parcial para optar al título de:
Ingeniero/a Mecánico/a

Director:
Dr. Guillermo Rodríguez

Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ingeniería Mecánica
Rosario, Argentina Año 2023

Agradecimientos

Agradezco principalmente al profesor Pablo Demartini por instruirme y acompañarme durante todo el proyecto.

Al Dr. Guillermo Rodríguez por escuchar mi deseo de cerrar el ciclo de grado con un proyecto de aplicación dentro de la Escuela de Ingeniería Mecánica y abrirme las puertas al laboratorio FABLAB.

A Rubén Favre, representante de la marca SEW EURODRIVE en Argentina y profesor en la Facultad de Ingeniería Electrónica, por responder todas las dudas acerca del producto.

A la catedra de Sistemas Digitales Industriales por mostrarme el maravilloso mundo de la Automatización.

A la Educación Pública.

A mis siete amigos.

Resumen

En la actualidad, tanto en el ámbito industrial como académico, son pocos los sistemas mecánicos que existen por sí solos. En su mayoría estos se encuentran integrados con sistemas de control que permiten el sincronismo de tareas entre los dispositivos y su automatización. En este contexto el proyecto se propone recrear un sistema convencional de industria, robot de 5 barras con dos grados de libertad, conocido como *"scara parallel robot"* utilizando los dos servomotores y sus controladores que tiene en desuso la Escuela de Ingeniería Mecánica.

Esto permitirá poner en valor los servomotores para que los estudiantes dispongan de una maqueta de aprendizaje tecnológicamente actualizada para el entorno educativo.

El proyecto tiene como objetivo poner en marcha los servomotores que tiene en desuso la Escuela de Ingeniería Mecánica, llevar adelante su automatización y redactar un manual de uso del software necesario para el control de los motores: MOVITOOLS y sus herramientas IPOSplus y AppBuilder, de propiedad de la empresa fabricante de los servomotores.

Palabras clave: servomotor, automatización, control, robot Scara, SEW EURODRIVE, MOVITOOLS, IPOSplus.

Abstract

Currently, both in the industrial and academic fields, few mechanical systems exist on their own. Most of these are integrated with control systems that allow the synchronization of tasks between the devices and their automation. In this context, it is proposed to recreate a conventional industry system, a 5-bar robot with two degrees of freedom, known as "scara parallel robot" using the two servomotors and their controllers that are in disuse at" Escuela de Ingeniería Mecánica".

This will allow the servomotors to be valued and the students to have a technologically updated learning model for the educational environment.

The aim of this project is to start up the servomotors that are in disuse at" Escuela de Ingeniería Mecánica", carry out their automation and write a user manual for the software necessary to control the motors: MOVITOOLS and its IPOSplus and AppBuilder tools, proprietary from the company that manufactures the servomotors.

Keywords: Servomotor, automation, control, Scara robot, SEW EURODRIVE, MOVITOOLS, IPOSplus

Contenido

Introducción

En la actualidad no se puede pensar ningún sistema mecánico complejo sin la interacción con sistemas electrónicos y computacionales. El progreso científico alcanzado por estas tres especialidades –mecánica, electrónica e informática- y su integración sinérgica logran procesos de producción con precisión en el movimiento y en la posición, posibilitan a su vez el sincronismo entre componentes y la automatización de las tareas, y permiten, po último una economía en el número de dispositivos involucrados, entre otras innumerables ventajas.

La industria ya está completamente inmersa en este nuevo mundo, de este modo la formación de grado universitaria necesita incorporar herramientas que cuenten con las nuevas tecnologías para que los estudiantes lleguen al mundo laboral con las competencias necesarias. Es así que nace en la Escuela Ingeniería Mecánica de la UNR el laboratorio *FABLAB*, cuyo objetivo es acercarles a los estudiantes un ambiente de trabajo donde puedan tener contacto con dispositivos tecnológicos actualizados tales como CNCs, tornos automatizados, impresoras 3D y prototipos robóticos. Es en este laboratorio donde se lleva adelante el proyecto de aplicación.

Objetivos

El proyecto se propone alcanzar los siguientes objetivos:

1. Poner en marcha dos servomotores sincrónicos SEW-EURODRIVE con sus respectivos variadores de frecuencia MOVIDRIVE- B
2. Desarrollar y fabricar un prototipo robot 5 barras con dos grados de libertad
3. Redactar un manual de utilización del software MOVITOOLS.

Metodología

En primer lugar, instalar en un espacio de trabajo adecuado los servomotores que se encuentran en desuso.
Con posterioridad indagar en el funcionamiento de los motores con la asistencia de manuales y publicaciones que se encuentran en el sitio web del fabricante SEW EURODRIVE en relación a componentes de los servomotores, los controladores y el uso del software.
Definir, con la información recabada, el mecanismo de barra a utilizar, resolver su cinemática, desarrollar el programa para el movimiento controlado de los motores y por último diseñar las piezas para la fabricación del prototipo de mecanismo de barras elegido.

1. Puesta en marcha de los Servomotores

Una tarea habitual de los ingenieros hoy en día es la modernización de los dispositivos existentes para evitar su obsolescencia. Últimamente esta actividad se encuentra estrechamente relacionada con la incorporación de sistemas de control en procesos ya existentes con el fin de automatizar y sincronizar tareas reutilizando así dispositivos en buen estado que permiten una disminución de costos sin perjudicar la eficiencia.

Es en este marco que el proyecto se propone: a) poner en funcionamiento y llevar a cabo la automatización sincrónica de dos servomotores sincrónicos SEW-EURODRIVE pertenecientes a la Escuela de Ing. Mecánica que cuentan con variadores de frecuencia MOVIDRIVES que en la actualidad están de desuso; b) estudiar sus funciones y c) documentar el desarrollo para facilitar el uso de estos dispositivos de gran complejidad que cuenta con un abanico amplio de aplicaciones.

2. Robot planar 5 barras (2gdl)

La mayoría de los robots planares, que cuentan con dos grados de libertad (X-Y) son robots cartesianos, también conocidos como pórticos, poseen dos motores, cada motor controla el movimiento lineal de un vector y estos vectores están dispuestos perpendicularmente uno de otro. Estos robots se están dejando de utilizar debido a las ventajas que presentan los robots 5 barras también llamados “scara”.

Los robots 5 barras están conformadas por cuatro barras conectadas en cadena que se fijan en los extremos a dos motores; se denomina quinta barra a la distancia entre los motores. Estos robots alcanzan una versatilidad en el uso y simples de diseño; no tienen correas, poleas ni engranajes. que simplifican significativamente su fabricación y mantenimiento. [3]

Sin embargo, estos diseños presentan una dificultad posible de subsanar con un buen control: la introducción de los modos de trabajo y singularidades hace que la combinación de los ángulos de entrada no mantenga de manera constante el punto del efector y por ende varíe su posición poniendo en juego la confiabilidad del robot.

3. Manual de uso

Los proyectos aplicados generan un conocimiento sobre los procesos de estudio que generalmente no quedan registrados y por lo tanto no permiten una acumulación que proponga un producto sobre el cual continuar un proceso de desarrollo crítico, cada uno que se aproxime al problema tiene que iniciar el proceso.

La producción de conocimiento aplicado y colaborativo es esencial para el desarrollo la universidad democrática, libre y gratuita. Generalmente las empresas ponen en el mercado productos que requieren de manera constante y sistemática de su asistencia técnica, como es el caso de nuestro estudio.

La producción de un manual o instructivo de uso pretende poner a disposición de las y los estudiantes la experticia adquirida en el tiempo de desarrollo del proyecto para que pueda ser utilizado, criticado y revisado por aquellas y aquellos estudiantes que en un futuro próximo trabajen con los servomotores existentes en nuestro laboratorio.

1. Introducción a los servomotores

¿Qué es un Servomotor?

Un servomotor es un tipo especial de motor que permite controlar la posición del eje en un momento dado. Está diseñado para moverse un rango determinado de grados y luego mantenerse fijo en una posición. Al hablar de un servomotor se hace referencia a un sistema conformado por componentes electromecánicos y electrónicos que logran una alta eficiencia y precisión en su posición y velocidad. [5]

En síntesis, es una combinación de piezas que incluye un motor de corriente continua o alterna, de bucle cerrado (Figura 1-1: : Lazo cerrado de controlFigura 1-1) que utiliza la retroalimentación de posición para controlar su velocidad de rotación y posición. La señal de control es la entrada, ya sea analógica o digital, que representa el comando de posición final para el eje.

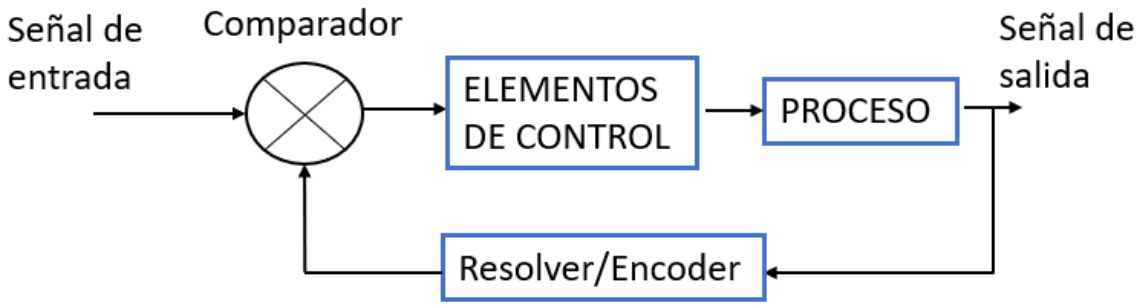


Figura 1-1: : Lazo cerrado de control

Los servomotores son utilizados en aplicaciones de automatización industrial y robótica por su bajo momento de inercia y su elevada capacidad de sobrecarga que les permite hacer movimientos muy rápidos, con grandes aceleraciones y frenadas. El driver del sistema, logra un exhaustivo control del par, del posicionamiento y de la velocidad para mejorar la calidad y la productividad.

Componentes de un Servomotor

Los servomotores son un sistema compuesto de partes eléctricas, mecánicas y electrónicas:

- **Motor eléctrico:** es el encargado de producir el movimiento mecánico a través de los giros del eje.
- **Sistema de control:** es la parte electrónica del mecanismo y es la encargada de controlar el movimiento y posición del motor mediante el envío de impulsos eléctricos. Debido a su importancia dentro del sistema amerita un apartado específico (ver
-
-
- Unidad de Control.)
- **Sistema de regulación:** está compuesto por engranajes, que permiten aumentar o disminuir el par y que el motor adopte una posición fija cuando así se requiera.
- **Potenciómetro:** este dispositivo electrónico permite monitorear en todo momento cuál es la

posición y ángulo en el que se encuentra el eje del motor.

Características de un servomotor

Las características de un servomotor son tres: alta velocidad, baja inercia y alta densidad de torque en bajo requerimiento de espacio.

Alta velocidad

$$\boldsymbol{n = \frac{2 * f * 60}{p}} \quad (1)$$

La alta velocidad esta expresada en la *Ecuación 1* en la que "$\boldsymbol{n}$" es velocidad nominal, "$\boldsymbol{f}$" es frecuencia suministrada y "$\boldsymbol{p}$" número de polos.
En un motor de inducción la frecuencia es fija y para aumentar la velocidad es necesario disminuir la cantidad de polos, por el contrario en nuestro caso de estudio, un servomotor, para modificar la velocidad es necesario variar la frecuencia por medio de un inversor o driver que permita alcanzar altas velocidades sin modificar el número de polos.

Baja inercia

La inercia, en física, es la propiedad que poseen los cuerpos de oponerse al cambio de estado en el que se encuentran, sea en reposo como en movimiento.
La inercia se expresa en la siguiente *Ecuación 2*, donde" $\boldsymbol{J}$" es la inercia**,** "$\boldsymbol{m}$**"** es la masa del objeto y "$\boldsymbol{r}$" es la distancia de la distribución de la masa con respecto a su centro.

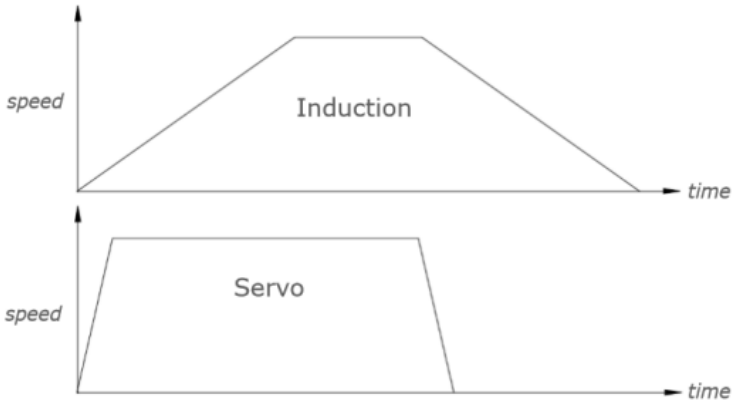
$$\boldsymbol{J = \frac{1}{2} * m * r^2} \quad (2)$$

Como se observa en la ecuación la inercia depende únicamente del radio y la masa, no de la longitud. Para disminuir la oposición al cambio los rotores del servomotor deben ser largos y con huecos en su interior (***¡Error! No se encuentra el origen de la referencia.***), lo que permite disminuir la inercia con facilidad.
La baja inercia del servomotor hace que éste se pueda posicionar de forma rápida y precisa de acuerdo a los comandos dados. Los servos motores soportan movimientos bruscos a grandes velocidades, es decir, son capaces de partir y frenar muy rápidamente.

[5]

Figura 1-2: Diferencia de tamaño entre un servomotor (izquierda) y un motor de inducción (derecha)

La baja inercia, le permite al servomotor tener rampas de aceleración y de frenado más cortas que los motores de inducción, como se muestra a continuación:



[5]

Figura 1-3: Comparación entre servomotores (grafica inferior) y motores de inducción (grafica superior) de graficas velocidad vs tiempo

Alta densidad de torque con bajo requerimiento de espacio

La densidad de potencia es la cantidad de energía que un motor determinado puede manejar por unidad de volumen. En términos realmente sencillos podríamos llamarlo la relación potencia/tamaño de un motor en particular.

Los servomotores requieren de menos volumen que los motores de inducción para generar la misma potencia ya que la potencia o torque depende de la inercia y de la aceleración angular.
La aceleración angular se expresa en la siguiente relación matemática (3):

$$a_\alpha = \frac{T_{orque}}{J} \tag{3}$$

Donde, "a_α" es Aceleración angular, "T"es torque y "J"es Inercia total.

Por lo tanto, la ventaja de un servomotor en comparación con un motor de inducción con la misma densidad de torque, es el espacio que abarca en el sistema que permite una mejor "ubicación espacial"

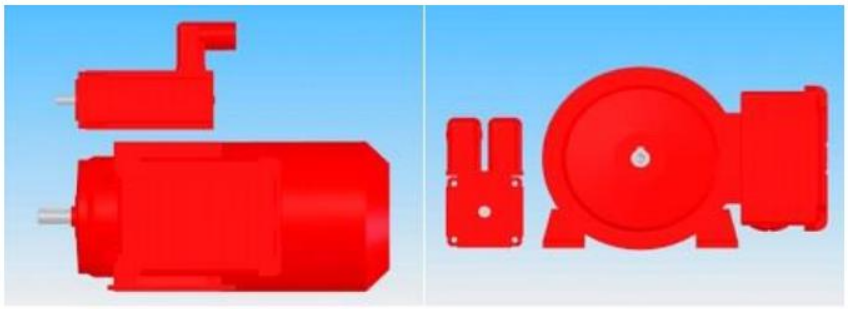


Figura 1-4: Diferencia de tamaño entre un servomotor y un motor de inducción con la misma densidad de torque.

Unidad de Control

El mecanismo de control de un servomotor es la retroalimentación. En ella cierta parte o proporción de la señal de salida de la unidad se redirige a la entrada del driver, y así adquiere datos necesarios para analizar y controlar el comportamiento del dispositivo. Esto se conoce como "lazo cerrado", permite controlar la velocidad y la posición porque puede comparar por medio de sensores la posición actual del eje del motor y modificarla para llegar a la posición deseada.

Servocontrolador o Driver

El servocontrolador (también conocido como controlador de movimiento) puede ser considerado como el cerebro del sistema del servomotor. En él reside el perfil de movimiento que incluye la aceleración, la velocidad y la desaceleración deseadas.
Su principal tarea es cerrar el bucle del sistema ya que le envía las señales al convertidor para que el motor ejecute el movimiento deseado y lea la información entregada por el encoder o resolver para corregir cualquier diferencia entre la velocidad, posición o par real frente a la deseada.

Potenciómetros

En la unidad de control se utilizan los potenciómetros denominados resolver y encoder. Estos pueden ir incorporados al servo (dentro de su estructura), o acopladas de forma externa (pieza física conectada al servo) y envían la información necesaria al driver (unidad de control) para que este, por su propia cuenta o en conjunto con un PLC analicen, descifren y controlen los datos de la realimentación (velocidad y posición).

Encoders

Los encoders son ***sistemas digitales*** de detección que proporcionan información y convierten el movimiento en una señal eléctrica que es leída por algún tipo de dispositivo de control de movimiento (driver) o PLC. Esta señal puede ser utilizada para determinar la posición, velocidad o dirección de un motor.

Los encoders pueden ser:

- Incrementales: no dispone de una referencia absoluta de la posición en la que se encuentra.

- Absolutos: la posición se da en valor absoluto mediante un bus paralelo. La cantidad de líneas que se leen no es muy elevada, por lo que la resolución es poca.

Resolvers

Los resolvers son ***sistemas analógicos*** que convierten el movimiento mecánico en una señal eléctrica, estos son generalmente utilizados para medir los grados de rotación, velocidad o torque de un motor. Esencialmente son un transformador rotativo con una salida de voltaje de CA que sigue la posición angular del eje. Los dos elementos del resolver son un rotor bobinado y un estator fijo, el primero gira dentro del segundo. El bobinado primario del resolver reside en el estator y el bobinado secundario reside en el rotor.

2. Componentes SEW- EURODRIVE

Los componentes disponible en la Escuela de Ingeniería Mecánica son: dos servomotores sincrónicos SEW-EURODRIVE, dos controladores MOVIDRIVE-B.

Servomotor sincrónico SEW-EURODRIVE

En este proyecto de aplicación usaremos un servomotor sincrónico marca SEW- EURODRIVE que tiene un resolver incorporado al motor. No cuenta con encoder externo.

El código de designación del servomotor RF17DS56L/TF/RH1M/KK y cada una de las partes brinda información sobre el mismo:

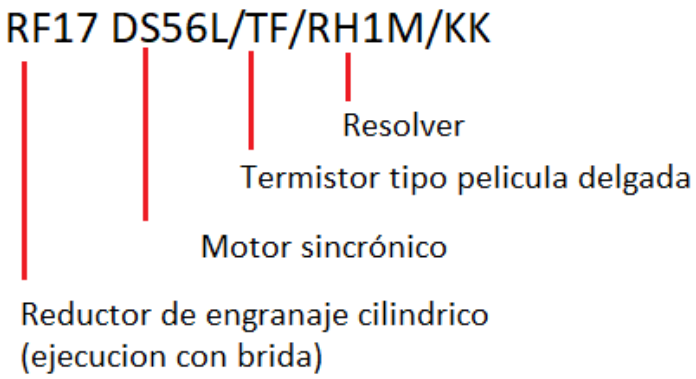


Figura 2-1: Código de designación del servomotor disponible en la Escuela de Ingeniería Mecánica

Esquema de Servomotor (sin reductor)

El gráfico muestra las partes del motor de importancia para el proyecto:

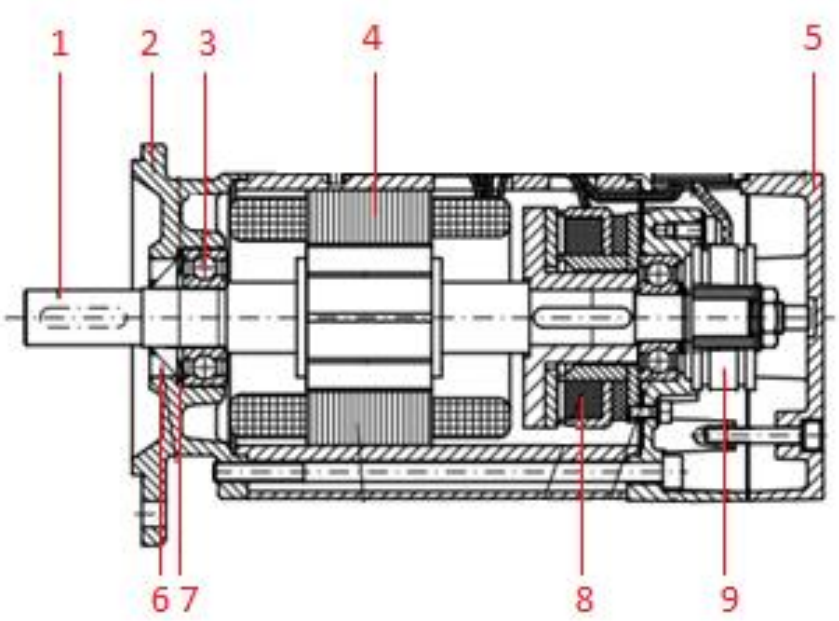


Figura 2-2: Partes del servomotor sin tener en cuenta el reductor

1. Rotor
2. Cascara con brida
3. Rodamiento de bolas acanalado
4. Estator
5. Tapa de cáscara
6. Retén sin muelle
7. Circlip
8. Freno completo
9. Resolver

Datos de producto

En la siguiente tabla se encuentra la información brindada por el fabricante según el código de designación del motor analizado en el apartado anterior:

Tabla 2-2-1: Datos de Producto

Velocidad [r/min]	3000 / 152
Índice reducción total [i]	19,71
Posición de montaje IM	M1
Velocidad entrada nepk [rpm]	4500
Velocidad salida napk [min-1]	228
Par de salida Mapk [Nm]	86
Código ISO	CLP 220
Tipo de lubricante	MINER.OIL
Cantidad de lubricante [l]	0,25
Velocidad nominal nN [rpm]	3000
Par a rotor parado M0 [Nm]	2
Límite máx. de par Mpk [Nm]	7,6
Corriente a rotor parado I0[A]	2,4
Corriente I máx.permitida[A]	9,6
Tipo de servicio S1-S10	S1
Tensión del motor [V]	400
Frecuencia máx. permitida [Hz]	150
Tipo aislamiento / IP	F / 65
Temperatura ambiente mín. [ºC]	-20
Temperatura ambiente máx. [ºC]	40
Peso	7,00 kg

Convertidor de frecuencia: MOVIDRIVE

Un MOVIDRIVE es un equipo convertidor de frecuencia, es decir, un aparato electrónico que permite controlar un motor o servomotor AC.
Si los motores eléctricos, también llamados motores AC, se operan directamente desde un sistema de suministro de voltaje AC, sólo pueden servirse de una velocidad fija basada en el número de polos y en la frecuencia de suministro del sistema de electricidad del lugar en el que se encuentre. Sin embargo, si una aplicación o proceso de producción requiere controlar distintas velocidades se requiere de los variadores de frecuencia.
Estos variadores de frecuencia pueden, como lo dice su nombre, generar distintas frecuencias según se requieran y consecuentemente controlar y variar la velocidad rotacional de un motor de corriente alterna (AC).

El MOVIDRIVE es llamado controlador universal ya que pueden utilizarse tanto en:

- Motores asincrónicos sin encoder, utilizados en transportadores, ventiladores, bombas y cortadores.
- Motores asincrónicos con encoder, utilizados en sistemas de almacenamiento, control de velocidad y posicionamiento.
- Motores sincrónicos, utilizados en manipulaciones dinámicas, sincronización de posición y máquinas de procesamiento.

Características del Equipo

Las características enunciadas a continuación son las ofrecidas por el fabricante en publicaciones revisadas:

1. Controlador universal: Posibilidad de manejo de Motores sincrónicos y asincrónicos.
2. Plug & Play: Posibilidad de adicionar tarjetas que el equipo reconoce automáticamente.
3. Programación de movimientos y secuencias a través de IPOS Plus incorporado.
4. Función “Safety Stop” integrada.
5. Tarjeta de Memoria integrada.
6. Módulo de Aplicaciones.
7. Programación rápida acorde a los datos transferidos directamente desde el encoder al equipo.
8. 8 entradas digitales / 6 Salidas digitales
9. 1 salida a relé / 1 entrada analógica.
10. Borneras removibles
11. Control automático de freno
12. Circuito chopper de frenado integrado.
13. Control PID integrado.
14. Memoria de fallos.
15. Puesta en marcha desde un panel o desde un PC.
16. Interfaces de comunicación: RS-485, Sbus y CanOpen integradas. Profibus, DeviceNet, Ethernet/IP

Designación de modelo MOVIDRIVE® MDX61B

El controlador SEW EURODRIVE que posee la Escuela es el MOVIDRIVE MDX61-B versión estándar que tiene integrado el control de secuencia y posición IPOSPlus. Sin embargo, es necesario resaltar que no se cuenta con la versión tecnológica por lo tanto no se pueden usar los módulos de aplicación de MOVITOOLS, es necesario desarrollar el programa de secuencia en el marco del proyecto.

El código de designación del MOVIDRIVE-B se analiza a continuación:

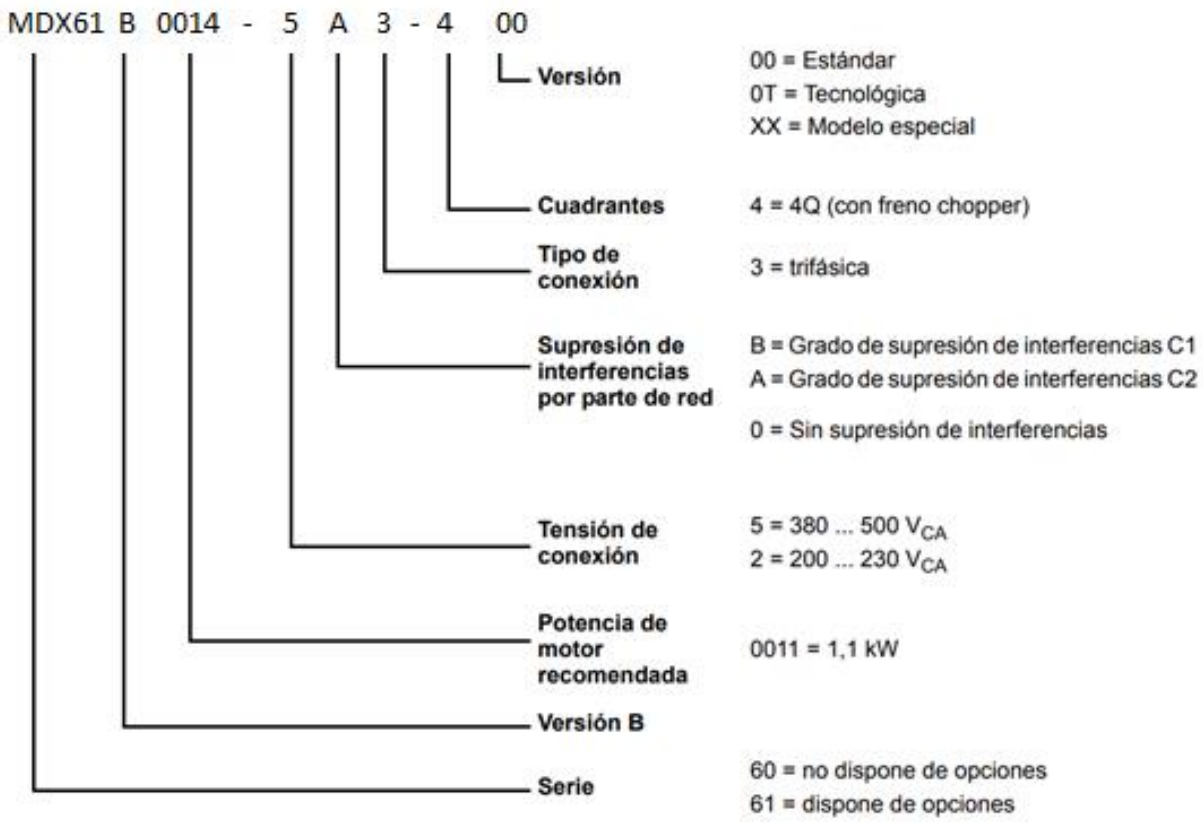


Figura 2-3: código de designación del MOVIDRIVE-B

Estructura del equipo

En los equipos usados para el proyecto se agregaron en el zócalo 8 un opcional para comunicación de enconder (DER11B) y en el zócalo 9 un opcional de comunicación (DIO11B).

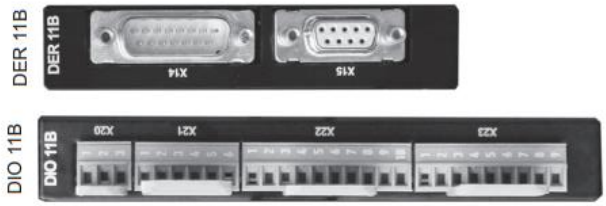


Figura 2-5: Opcionales agregados a los controladores disponibles en la Escuela de Ingeniería Mecánica

A continuación, se muestran las partes de la estructura del controlador de significación para el proyecto:

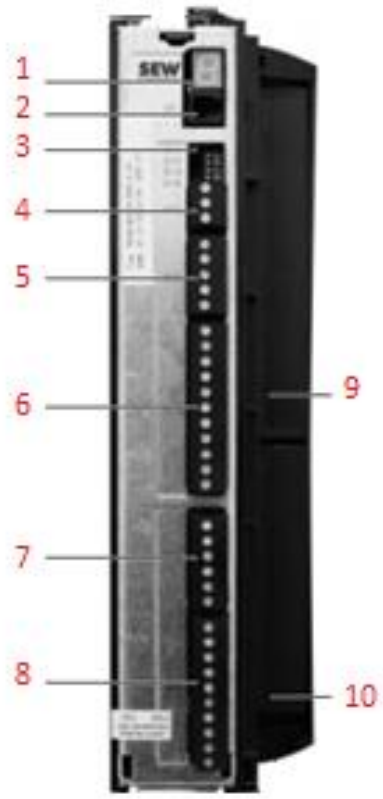


Figura 2-4: Estructura de controlador MOVIDRIVE-B

1. Display de 7 segmentos.
2. XT: Zócalo para la consola de programación DBG60B o para la interface en serie UWS21B
3. Interruptores DIP S11 ... S14 (usados para establecer comunicación vía SBUS).
4. X12: Regleta de bornas de señal para bus de sistema (SBus).
5. X11: Regleta de bornas de señal para entrada de consigna AI1 y tensión de referencia de 10V.
6. X13: Regleta de bornas de señal para entradas binarias e interface RS485.
7. X16: Regleta de bornas de señal para entradas binarias y salidas binarias.
8. X10: Regleta de bornas de señal para salidas binarias y entrada TF/TH.
9. Zócalo para bus de campo
10. Zócalo para encoder

Borneras de señalización

A continuación se muestra el gráfico proporcionado por el fabricante donde se define cada borne y los siete segmentos del display que muestra el estado de funcionamiento del servomotor:

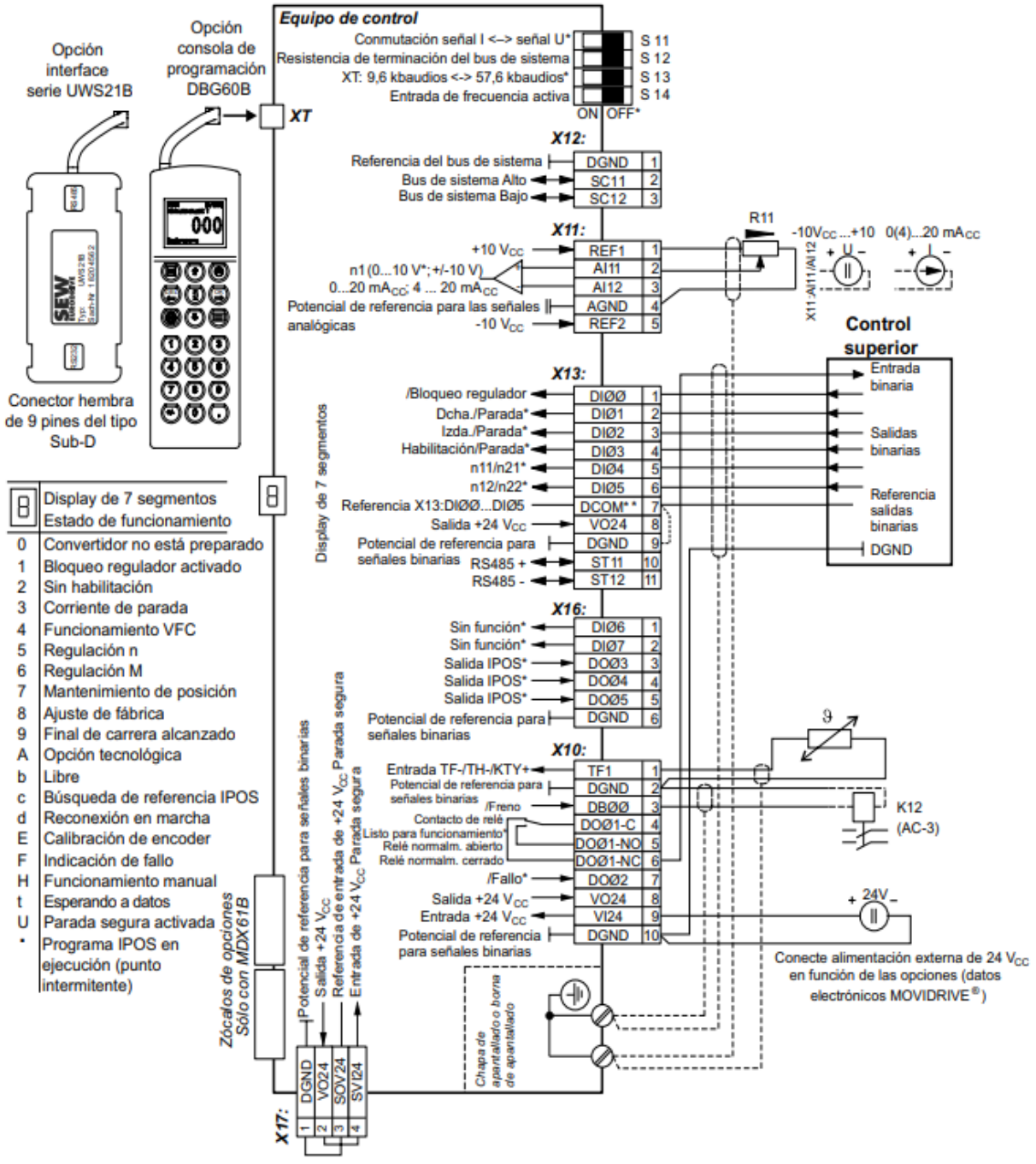


Figura 2-6: Borneras de señalización

3. Puesta en marcha del servomotor

Si bien ya se explicitó anteriormente, es necesario recalcar que la puesta en marcha de los servomotores se lleva a cabo después de mucho tiempo de haber sido adquiridos por la Escuela de Ingeniería Mecánica. Esta puesta en marcha se realiza sin contar con recurso humano especializado en la marca SEW EURODRIVE, por lo tanto se inicia un proceso de aprendizaje- investigación-trabajo con los conocimientos obtenidos durante la formación de grado en relación a los servomotores en general y con el importante acompañamiento de los docentes del Laboratorio FABLAB con amplia experiencia en electromecánica.

Puesta en marcha

La puesta en marcha de los servomotores se puede realizar de dos maneras, por medio de la consola de programación DBG60B o a través de la conexión de los controladores a la PC usando el software MOVITOOLS (ver Cap. 4). En este proyecto se opta por esta última dada su practicidad. El control de los servo se hace en su totalidad por medio de las borneras y se deja para una segunda etapa el aprendizaje del software IPOSPlus.

Comunicación PC- MOVIDRIVE B

Una vez encendido los motores para todo tipo de control es necesaria la comunicación con la PC ya que de ella se obtendrán los parámetros de MOVITOOLS y la elección del modo de trabajo.

La comunicación se logrará por medio del zócalo XT del MOVIDRIVE con un adaptador USB11A marca SEWDRIVE, usando un protocolo de comunicación Serial RS485.

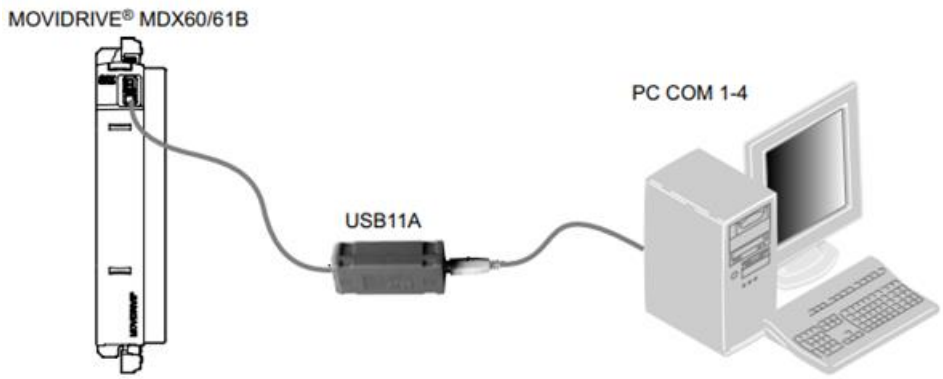


Figura 3-1: Conexión PC-CONTROLADOR

La comunicación del driver con la PC es necesaria para:

- la transferencia de parámetros definida en MOVITOOLS
- la descarga al controlador del programa de tareas desarrollado en IPOSPlus.
- el uso del tablero virtual de AppBuilder, como interface hombre-máquina

Arranque de motor

El arranque del motor requiere indefectiblemente la utilización de MOVITOOLS. Al abrir por primera vez MOVITOOLS se debe seleccionar el idioma, el tipo de dispositivo (en este caso MOVIDRIVE -B) y el grupo "Baudrate", donde se marca la velocidad de transmisión ajustada en la unidad básica con el interruptor DIP S13 (en este caso 9,6 kbaudios).

El arranque de motor puede ejecutarse a través de las borneras mediante dos opciones:

1. Solamente con la bornera X13 donde se podrá modificar el sentido de giro con tres velocidades prefijadas en MOVITOOOLS.
2. Para velocidades variables se usan las borneras X13 y X11 (esta última cuenta con un potenciómetro).

Para ambos casos es indispensable entregarle una señal “1” a la borna X13:1 (DIØØ) ya que esta desbloquea el regulador.

Opción 1: Especificaciones con consignas fijas

Para que el accionamiento funcione con consignas fijas se debe definir el parámetro Sepoint source de MOVITOOLS (P100) como "BIPOLAR/CONSIGNA FIJA" y aplicar en las borneras X13:1...X13:6 (DIØØ...DIØ5) las señales adecuadas según se define en MOVITOOLS.

En el proyecto, la puesta en marcha se hizo con los parámetros configurados como se observa en la Figura 3-2 y se energizaron las entradas DI00, DI02 y DI04 conectándolas con la salida a 24 que permiten eliminar el bloqueo del regulador, habilitar el motor y hacerlo girar en sentido contrario a las agujas del reloj (CCW).

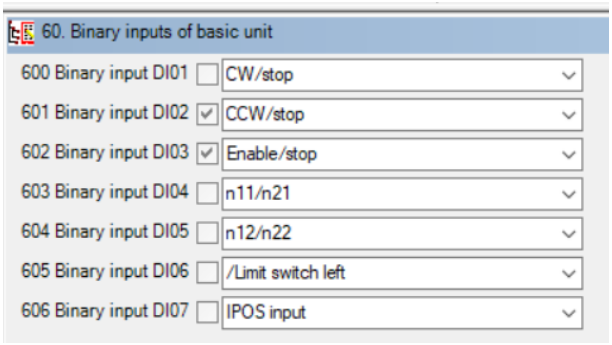


Figura 3-2: Definición de parametros DI00, DI02 , DI03 y DI04 en MOVITOOLS

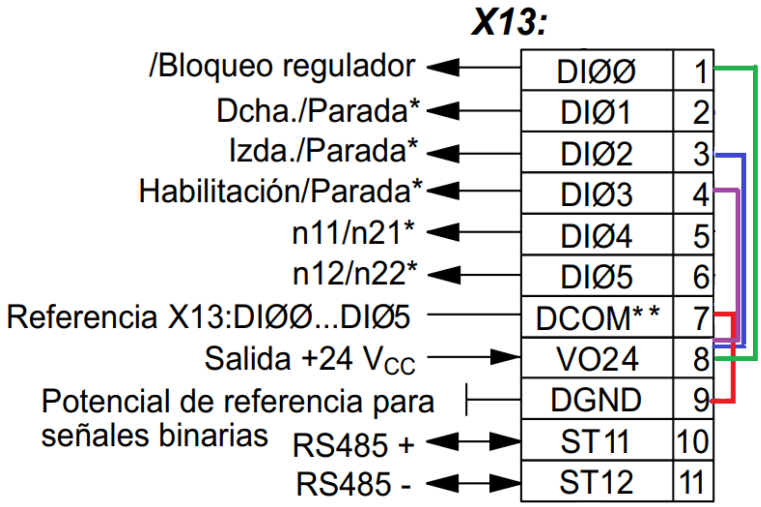


Figura 3-3- Bornera X13 con sus conexiones

Opción 2: Especificación de las consignas analógicas

En este caso se varia la velocidad con un potenciómetro conectado a la bornera X11 y la bornera X13 se mantiene como en la Opción 1.

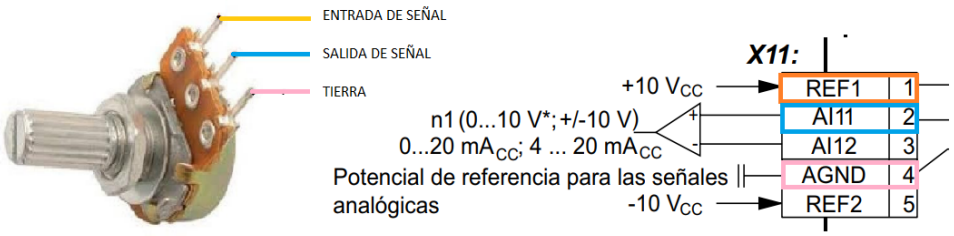


Figura 3-4- Conexión potenciómetro con bornera X11

Funcionamiento manual con Movitools

El funcionamiento manual mediante MOVITOOLS permite al usuario controlar el motor desde la computadora en tiempo real, se puede cambiar el sentido de giro, variar velocidades y parar el motor. Antes de comenzar es necesario habilitar el modo manual *("Activate manual mode")*.

Durante el funcionamiento, el display del equipo de 7 segmentos muestra la letra "H" y las entradas binarias, exceptuando X13:1 (DIØØ "/Bloqueo de regulador"), son ineficaces.

La entrada binaria X13:1 (DIØØ "/Bloqueo de regulador") debe recibir una señal "1" para que pueda arrancar el motor. Si X13:1 = "0" el funcionamiento manual no será posible. El sentido de giro no se determina por las entradas binarias "Dcha./Parada" o "Izda/Parada" sino mediante la selección en MOVITOOLS (CW y CCW).

El funcionamiento manual permanece activado también tras la desconexión y la conexión de red, aunque en estos casos el variador quedará bloqueado. Con la tecla "Run" se activa la habilitación y el arranque con $n_{mín}$ en el sentido de giro seleccionado.

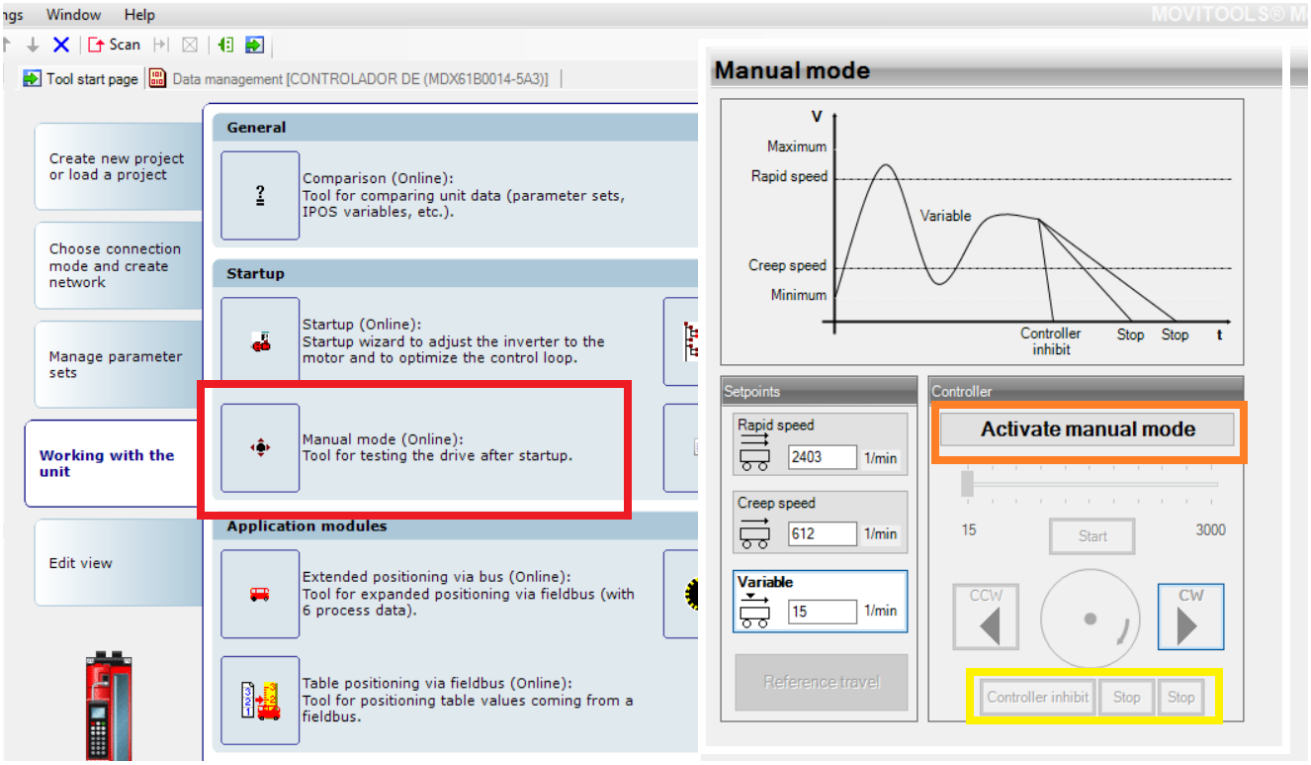


Figura 3-5: Acceso a MODO MANUAL en MOVITOOLS

4. Desarrollo de contenidos básicos del Software MOVITOOLS® MotionStudio

Para el uso eficiente de los servomotores es indispensable el control total del software MOVITOOLS donde se desarrolla el programa que controla la sincronización de las tareas del motor. Se tiene que indagar exhaustivamente sobre los parámetros del software, el lenguaje de programación de la herramienta IPOSPlus y el uso de la herramienta AppBuilder para construir la interfase humano-máquina, esta tecnología blanda constituye la asistencia que el fabricante brinda a sus clientes del sector privado. La Escuela no cuenta con esta asistencia, y es precisamente el desarrollo de los contenidos básicos del software uno de los aportes del proyecto.

MOVITOOLS

Movitools es el paquete de software libre de ingeniería desarrollado por SEW EURODRIVE compatible con todos los equipos electrónicos de la marca que permite:

- Parametrizar los variadores vectoriales
- Puesta en marcha del servomotor
- Visualización y diagnóstico del estado del servomotor
- Programación de la secuencia de tareas

Para ejecutar las funciones mencionadas anteriormente se requiere los siguientes componentes básicos incluidos en el paquete de software:

- MotionStudio
- MOVITOOLS

En estos componentes si bien existen numerosas herramientas en el proyecto se desarrollan y se usan las siguientes: IPOS Plus assembler and compiler para la programación de los variadores y AppBuilder para la interfaz gráfica. [11]

Estructura del Software

Estructura de la interface

La estructura de la interface está compuesta por tres áreas dinámicas que se modifican según las elecciones del usuario: área de visualización, área de trabajo e indicación y área de estado.
En la siguiente figura del software se resaltaron las áreas y las funciones más importantes para el proyecto con la explicación correspondiente:

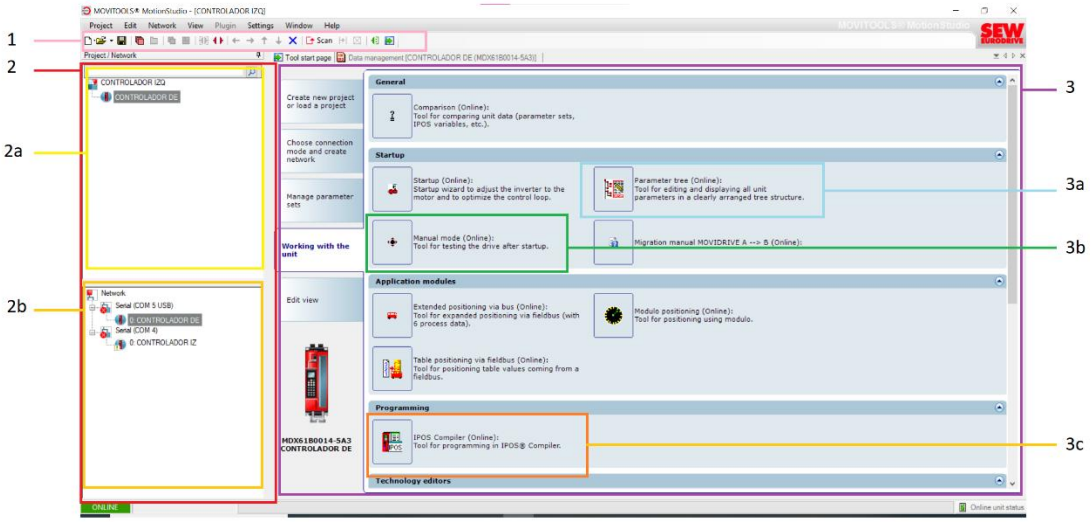


Figura 4-1: Interface de inicio MOVITOOLS

Tabla 4-4-1: Referencias Figura 4-1

1		Barra de herramientas	Contiene todos los comandos importantes para navegar por la estructura
2		área de visualización	Se observan las unidades del proyecto. Al hacer click izquierdo sobre ellas se habilita el menú contextual
	2a	Vista de unidades de proyecto	Se puede usar las herramientas y trabajar sobre los parámetros de dispositivos fuera de red
	2b	Vista de unidades en red	Se ven los dispositivos con un enlace de comunicación a la PC en el momento.
3		Área de trabajo e indicación	Se muestran dos tipos de complementos: • Las "herramientas" que se llaman desde el menú contextual. • La "página de inicio de herramientas" tan pronto como seleccione una unidad (se ve en la imagen)
	3a	Arbol de Parametros	Se accede a los parámetros de la unidad seleccionada en el área de visualización
	3b	Modo Manual	Se accede al modo manual de la unidad seleccionada
	3c	IPOSPlus Compiler	Se accede al software para programar las tareas del variador

Estructura del menú contextual

Para ingresar al menú contextual de cada dispositivo se debe hacer click secundario sobre el dispositivo que se define en el área de visualización. Es importante verificar que se está trabajando sobre la unidad que se desea.

El menú cuenta con tres áreas definidas:
En la primera (1) se observan las últimas 4 herramientas utilizadas. Ingresan dinámicamente.
En la segunda (2) se muestran la localización de las herramientas disponibles.
En la tercera (3) se encuentran los comandos para la gestión de proyectos y el comando para mostrar el estado de la unidad en línea.

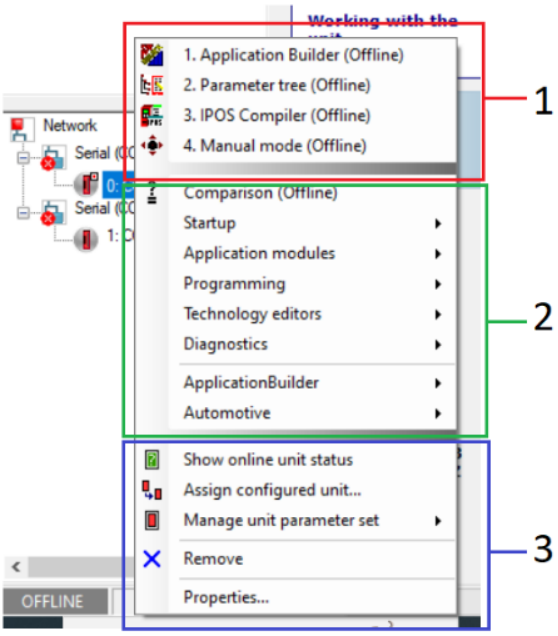


Figura 4-2: Menú contextual

Modo de conexión

En MOVITOOLS es posible trabajar con la unidad tanto en línea como fuera de ella.

- ONLINE: todos los cambios realizados en MOVITOOLS afectan únicamente a la unidad y se pueden ver instantáneamente en el funcionamiento del MOTOR. Es necesario guardar los cambios para que estos se almacenen en el disco duro de la PC y cargarlos *("Upload unit->PC)"* para que se transfieran a la memoria RAM.

- OFFLINE: todos los cambios realizados afectan únicamente a la memoria RAM. Se deben guardar para que estos se almacenen en el disco duro de la PC y descargar en la unidad para que los cambios se vean reflejados *("Download (PC->unit)").*

Para elegir el modo de conexión se pueden usar los iconos de la barra de herramientas:

1. Cambiar a modo online (en línea)
2. Cambiar a modo offline (fuera de línea)

Figura 4-3: Barra de herramientas MOVITOOLS

Cómo crear un nuevo proyecto

Un proyecto contiene todas las unidades que se encuentran involucradas y en él se definen los parámetros particulares de cada unidad.
Pasos de creación de un proyecto:

1. definir el nombre y la ruta de acceso usando la barra de herramienta de MOVITOOLS
2. Configurar el canal de comunicación.

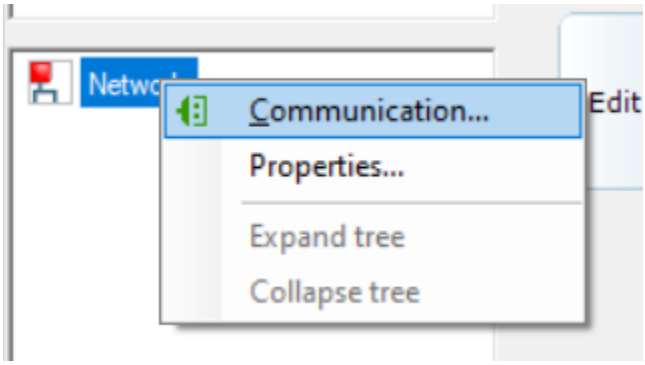


Figura 4-4: Ingreso a la configuración del canal de comunicación

Elegir la opción ***Serial*** y luego hacer click sobre **Edit** para corroborar la elección de puerto, se deberá elegir la que tiene el numero + (UBS)

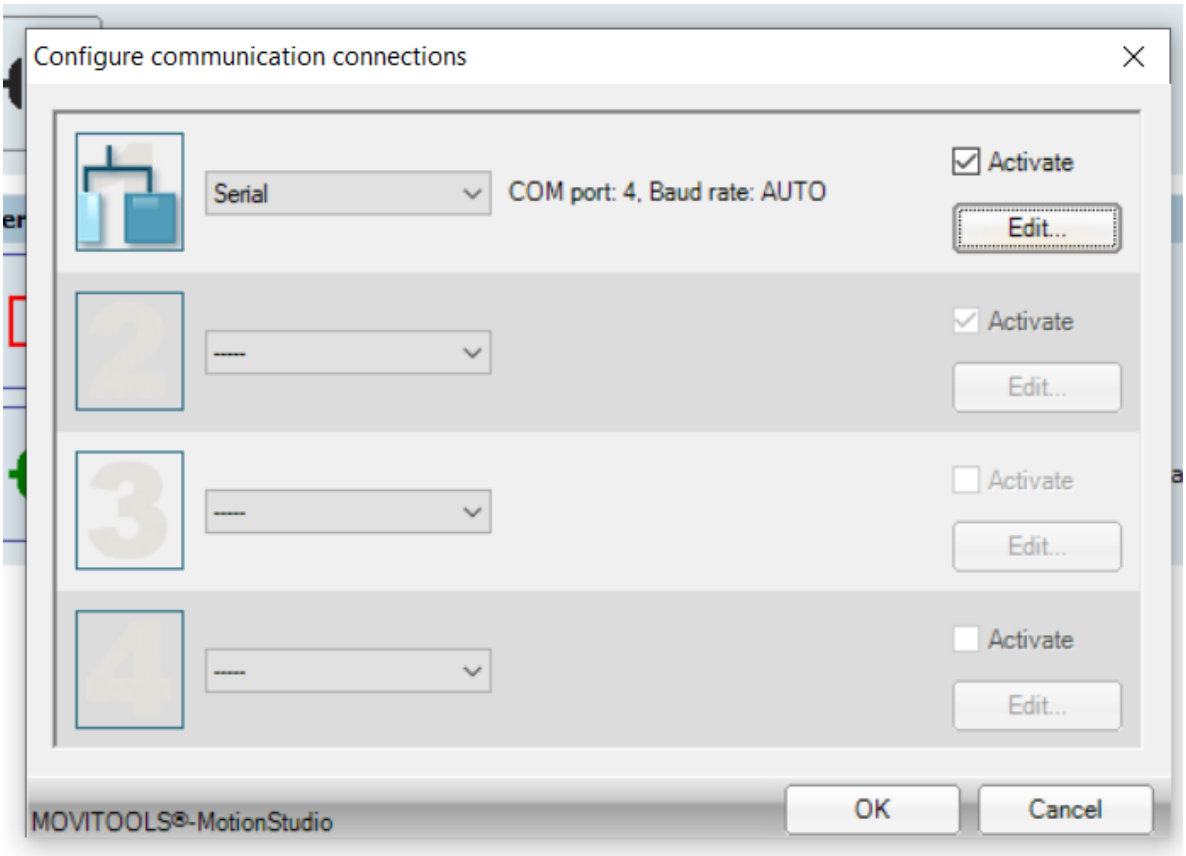


Figura 4-5: Configuración de los canales de comunicación

1. Scanear la red para encontrar los dispositivos haciendo esta vez click secundario sobre el modo de comunicación elegido Serial) y seleccionando ***Network scan (…)***

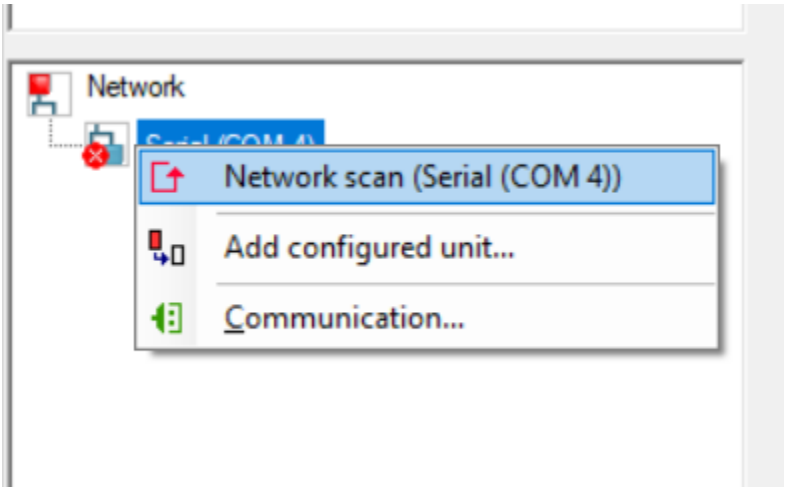


Figura 4-6: Scanear red

1. Conectar de forma online con los dispositivos para trabajar en tiempo real y configurar los parámetros

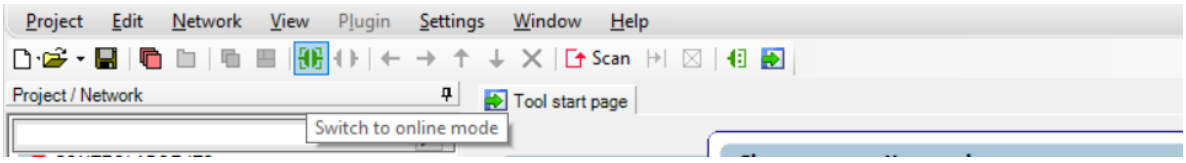


Figura 4-7: Conexión a modo online desde la barra de herramientas

2. Guardar los cambios realizados.

Árbol de parámetros

El árbol de parámetros es una vista que está dividida en dos secciones: parámetros de carácter informativo, no se pueden modificar (1) y parámetros de lectura y escritura, se pueden modificar haciendo click sobre ellos y apretando *Intro* una vez modificados (2).

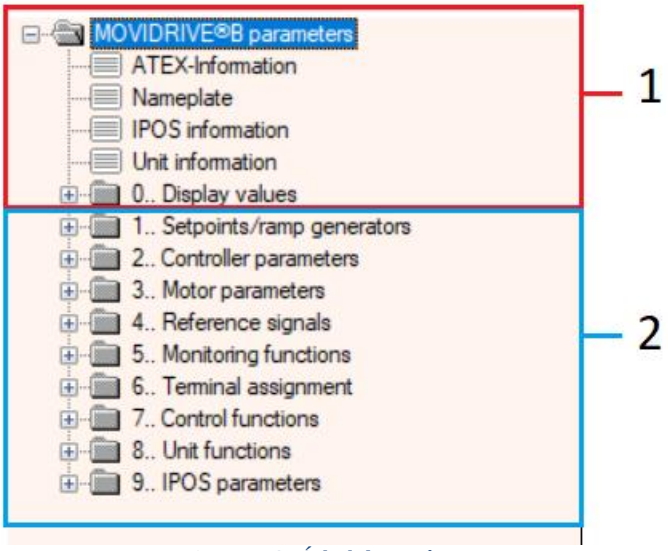


Figura 4-8: Árbol de parámetros

En este árbol los parámetros están divididos en grupos y en subgrupos según su naturaleza y a partir de esta subdivisión se le da un número de identificación:

$$Grupo\ principal + subgrupo + parametro\ individual = Código\ de\ Parámetro$$

Ej. El parámetro ***Setpoint source*** (100) es el primer parámetro del subgrupo ***Setpoint selection*** (10) del grupo principal ***Setpoints/ramp generators*** (1) .

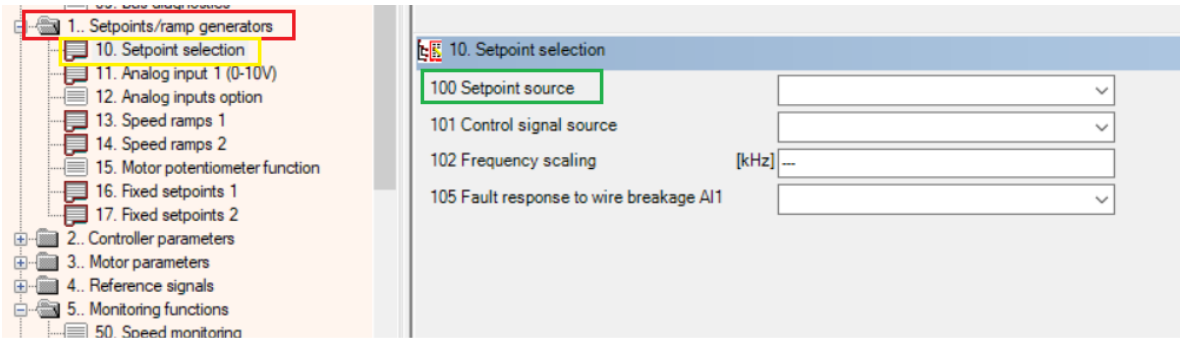


Figura 4-9: Método de búsqueda parámetro Setpoint Source

Cada unidad tiene un árbol de parámetro propio y se accede llamando al menú contextual en la sección *"Start Up"*

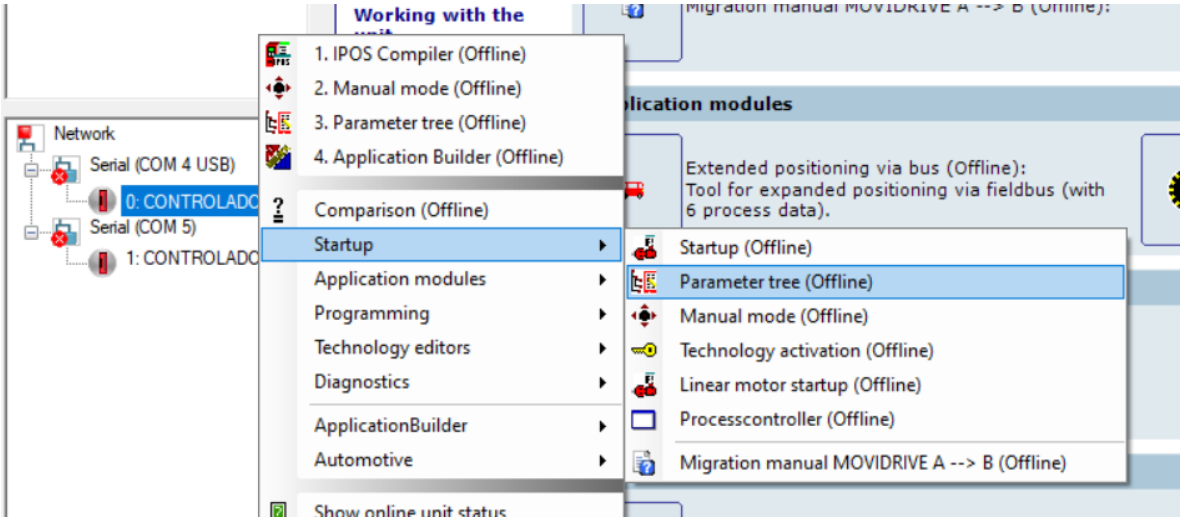


Figura 4-10: Acceso a árbol de parámetros de cada unidad

Parámetros usados en el proyecto

Si bien la gran cantidad de parámetros hace imposible para los usuarios acordarse de todos ellos, MOVITOOLS cuenta con una ayuda online de gran utilidad. Para acceder a ella solo se debe apretar **F1** una vez seleccionado el parámetro a buscar.

En esta sección se expondrán únicamente los parámetros que deberán ser definidos para el funcionamiento del proyecto

Setpoint selección

- Setpoint Source: define de donde recibe el controlador los puntos de ajuste
- Control signal source: define de donde se reciben las señales de control (bloqueo regulador, reseteo automático y modo de funcionamiento)

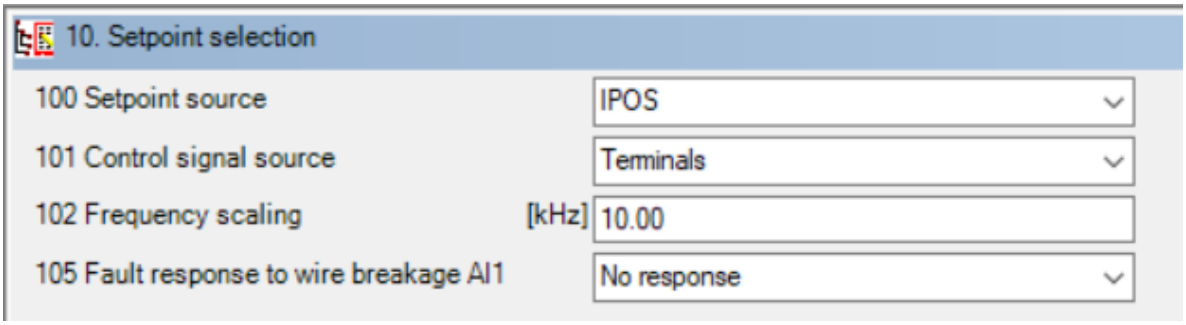


Figura 4-11: Parámetros del subgrupo Setpoint selection

Binary inputs of basic unit

En esta sección se pude definir que entradas binarias serán controladas por los bornes 1 a 8 (DI01 a DI07) de la bornera X11. El borne 0 (DI00) está reservado para la inhibición del controlador por lo que este no se puede modificar.
Estas entradas binarias pueden ser normal cerradas o normal abiertas. Las primeras tienen una barra inclinada "/" al inicio de su nombre para identificarlas y la acción programa se realiza cuando el borne no tiene voltaje. Se usan para eventos de fallas o paradas.

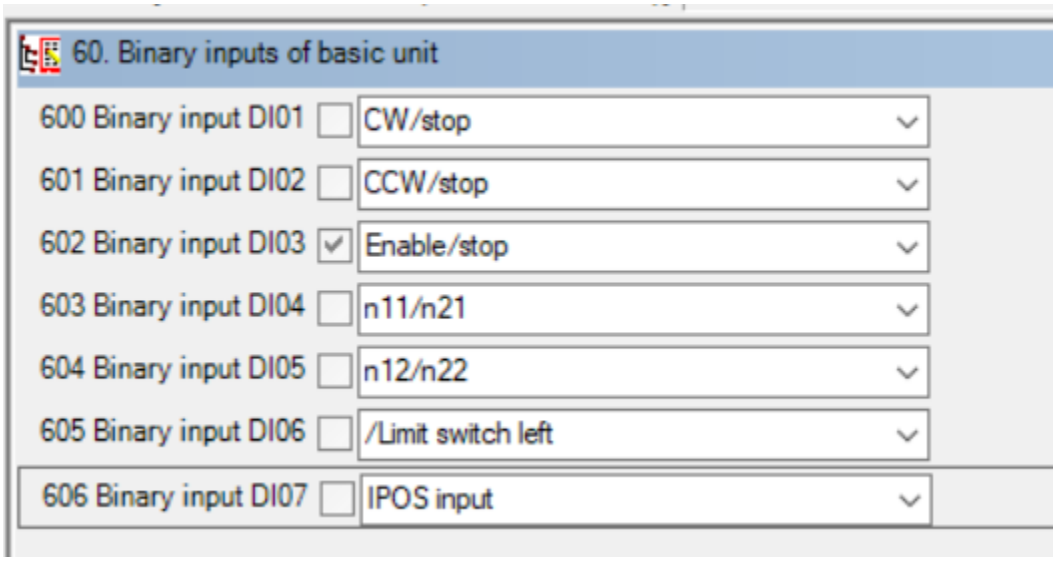


Figura 4-12: Parámetros del subgrupo Binary inputs

Serial communication SBus 1

- SBus 1 protocol: para este proyecto se usa el protocolo MoviLink
- Bus 1 group address: todas las unidades del sistema tienen que tener el mismo valor en este parámetro para permitir la comunicación.

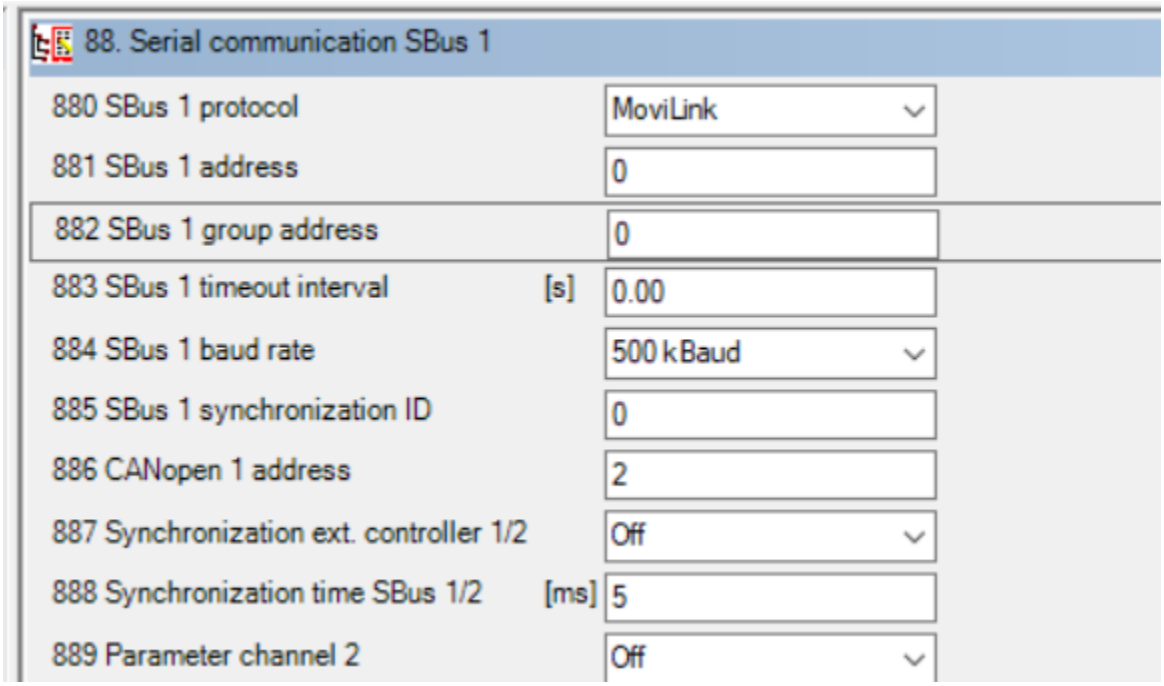


Figura 4-13: Parámetros para la comunicación SBus

IPOS Reference travel

- Reference travel type: especifica la estrategia de recorrido de referencia que se utiliza para establecer el cero de máquina.

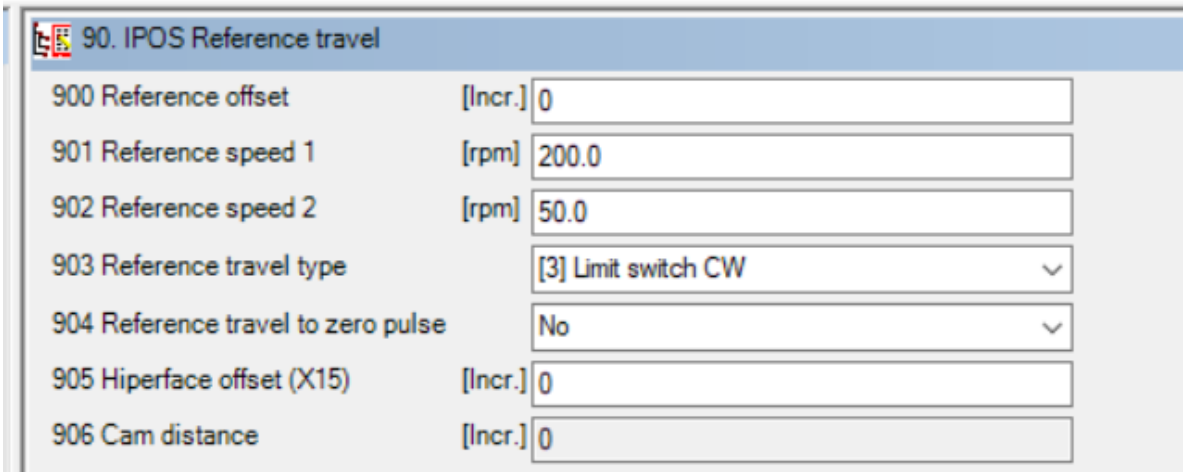


Figura 4-14: Parámetros para viaje a referencia usado en IPOS

IPOS Travel paramaters

- Travel Speed CW/CCW: Especifica la velocidad utilizada para el posicionamiento. La configuración debe ajustarse a la velocidad máxima del motor.

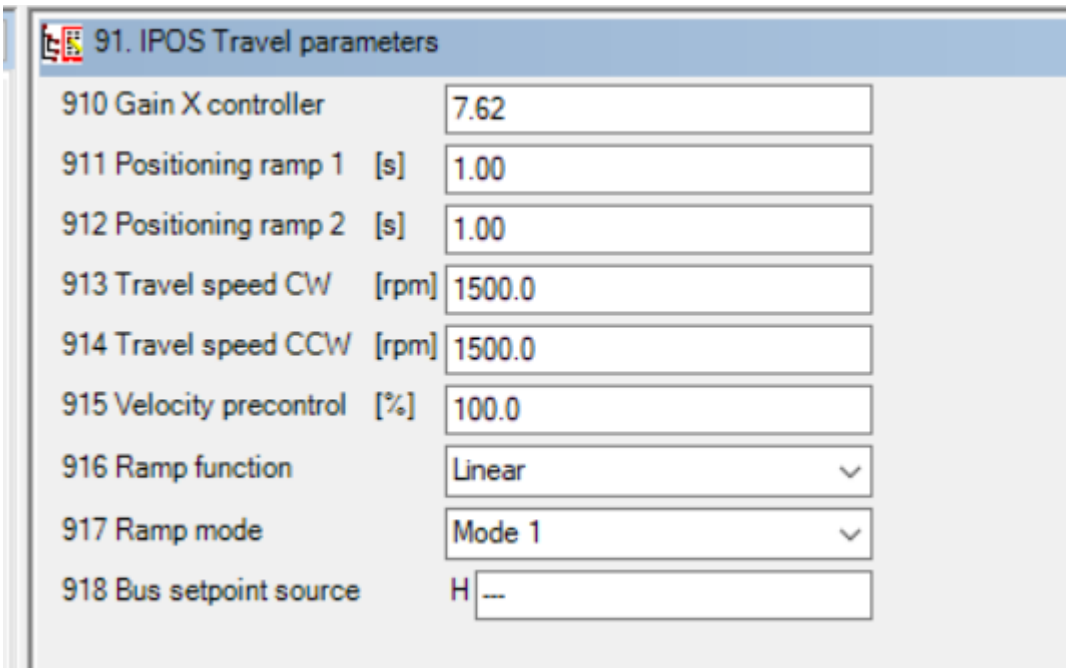


Figura 4-15: Parámetros de velocidad de viaje para IPOS

IPOSPlus

SEW EURODRIVE utiliza un entorno de programación llamado IPOS®. Este programa IPOS® está contenido dentro de la plataforma MOVITOOLS MOTION STUDIO. En él existen dos módulos que permiten a los usuarios programar los equipos: en lenguaje ensamblador (IPOSplus Assembler) usado para antiguos equipos, o en lenguaje C (IPOSplus Compiler) con el cual trabajaremos en el proyecto.

IPOS® (Sistema de Control de Secuencia y Posicionamiento integrado) es una potente herramienta de trabajo para expandir las capacidades de un variador de frecuencia pudiendo controlar una secuencia de movimientos con alta precisión en el posicionamiento sin llegar a la necesidad de usar un PLC. Con IPOS las funciones de control y las tareas de posicionamiento pueden ser ejecutadas simultánea o independientemente.

Ventajas IPOSplus Compiler

La programación en el Compilador también ofrece [9]:

- Creación de programas en un lenguaje de alto nivel
- Nombres de variables simbólicas
- Posibilidad de crear módulos de programa que se pueden volver a utilizar en otros proyectos
- Programación clara, modular y estructurada

- Diferentes técnicas de programación para loops
- Control del compilador usando comandos del preprocesador
- Estructuras estándar
- Estructuras definidas por el usuario
- Funciones estándar
- Depurador para resolución de problemas
- Amplias opciones para hacer comentarios

Primeros pasos

En esta sección se desarrolla un instructivo para aquellos usuarios que se están familiarizando con el software. [11]

Paso 1: iniciar el compilador IPOSplus® con MOVITOOLS® MotionStudio

Se debe definir para qué unidad se desea desarrollar el programa y seleccionarlo en el proyecto de MOVITOOLS en el que se está trabajando. Luego abrir el menú de contexto para seleccionar dentro de *"programming"* la opción *"IPOS Compiler"*

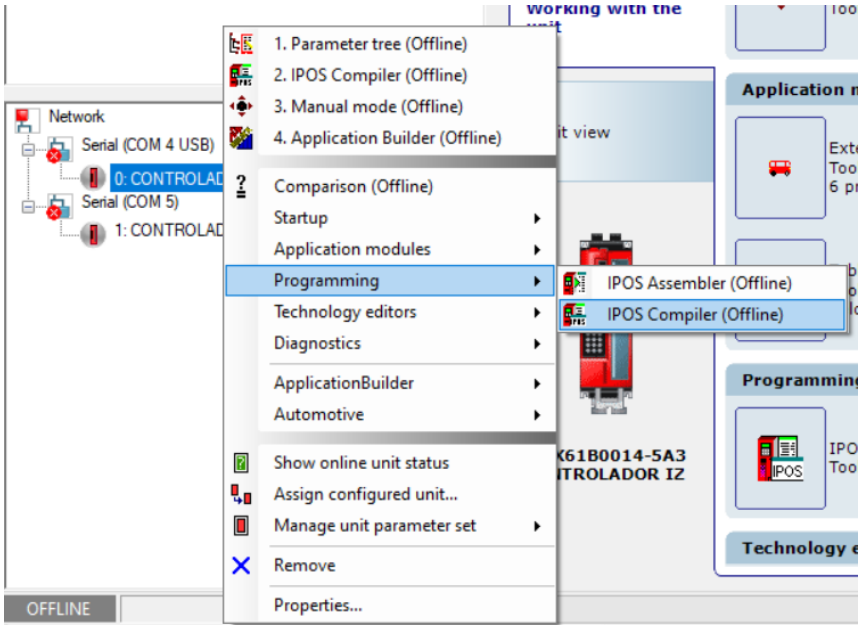


Figura 4-16: Acceso a IPOS Compiler

Se abrirá la interface de IPOSPlus

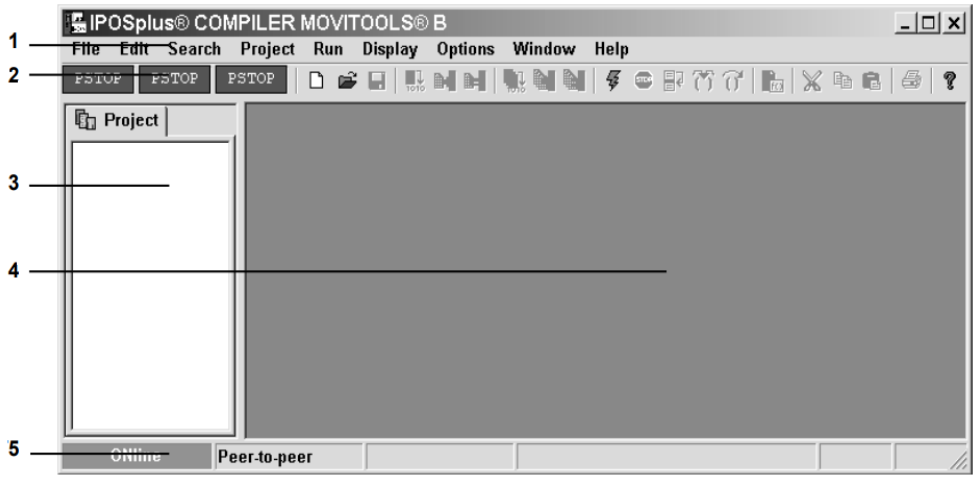


Figura 4-17: interface IPOSPlus

1	Menú
2	Barra de herramientas
3	Ventana de proyecto
4	Ventana de programación
5	Barra de estado

Paso 2: Creación de un nuevo proyecto

Al seleccionar la opción “New Proyect” en la opción “File” del Menú aparece una ventana donde se debe definir el nombre del archivo (primera línea) y ruta de acceso (segunda línea). Se recomienda guardar el programa dentro de la carpeta donde están todos los datos del proyecto MOVITOOLS. Las restantes líneas no deben modificarse ya que están definidas por defecto.

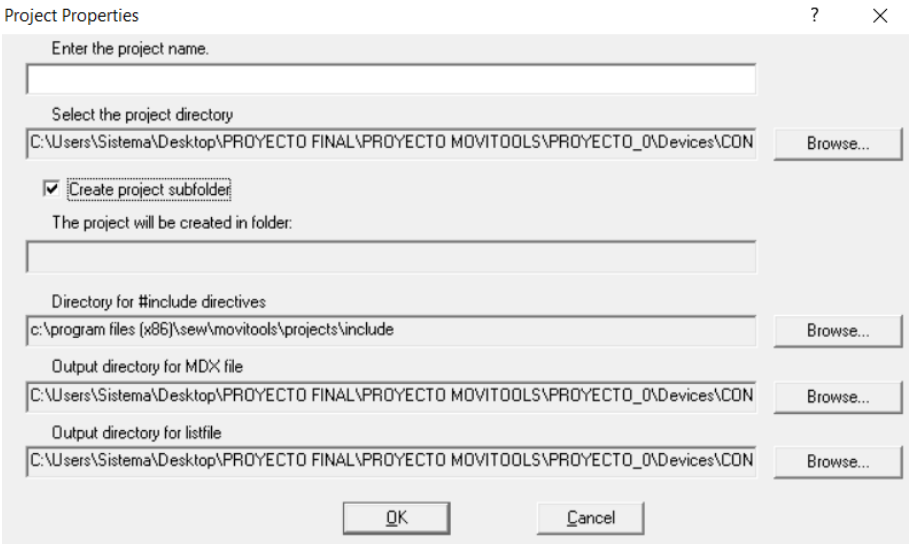


Figura 4-18: Definición de ruta de acceso al proyecto

Luego, se crea dentro del proyecto un archivo de texto fuente (.ipc) desde el menú eligiendo la opción "New source file". Una vez introducido el nombre, aparece una ventana para definir la estructura del programa.

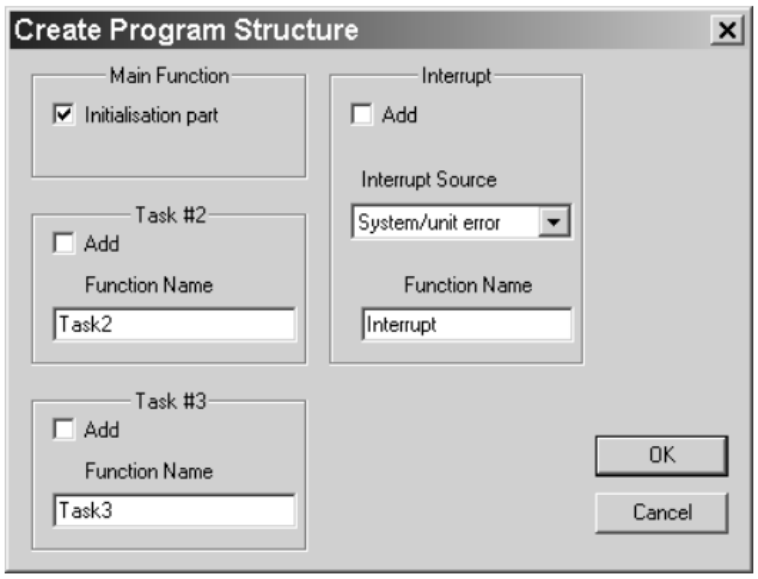


Figura 4-19: Ventana de creación de estructura de programa

Marcar en ella el cuadro *"Initialisation part"* y hacer click en *"Ok"*. la función "Principal" se genera automáticamente con la parte de inicialización.

```
/*=============================================
IPOS source file
=============================================*/
#include <constb.h>
#include <iob.h>

/*=============================================
Main function (IPOS initial function)
=============================================*/
main()
{

 /*-------------------------------------
 Initialization
 -------------------------------------*/

 /*-------------------------------------
 Main program loop
 -------------------------------------*/
 while(1)
 {

 }

}
```

La función "Principal" contiene una parte de inicialización y el bucle del programa principal. Este programa podría ejecutarse, pero como no contiene ninguna función no se verá ninguna modificación en el controlador.

Paso 4: Compilar e iniciar el programa IPOSPlus®

Para poder ver cambios en la unidad es necesario que el programa tenga contenido. Se puede usar el siguiente desarrollo a manera de ejemplo en el que se habilita el giro del motor en sentido de las agujas del reloj, en cada ciclo del programa el eje del motor girará aproximadamente media vuelta y la variable ***H1*** incrementará en 1.

```
    /*=============================================
IPOS source file
=============================================*/
#include <constb.h>
#include <iob.h>
/*=============================================
Main function (IPOS initial function)
=============================================*/
main()
{

_BitClear( ControlWord, 1 ); //habilitacion
_BitClear( ControlWord, 3 ); // sentido CCW
_BitSet(ControlWord,2) ;    // sentido CW

 Main program loop
 -------------------------------------*/
 while(1)
 {
 _GoRel( GO_WAIT,40000 );   //girará 40000 pulsos
 _Wait (500); // esperará 500 milisegundos
H1=H1+1;
 }
 }
```

Por último, se debe compilar y cagar en el driver usando los iconos  de la barra de herramientas e iniciar el programa  .

Visualización de variables

Durante la programación y las pruebas es útil observar el cambio en tiempo real de las variables.

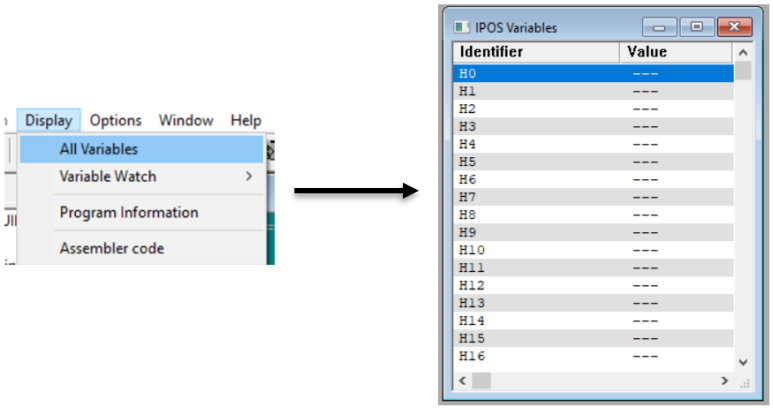


Figura 4-20: Visualización de variables IPOSplus

Cabeceras y distribución de memoria

La cabecera ‘const.h’ / ‘const.h’ define identificadores y funciones útiles para el comportamiento del variador de frecuencia tales como el control del bus de campo, velocidad de los motores, rampas de aceleración y desaceleración, timers, puesta a cero del encoder, etc. Mientras que la cabecera ‘io.h’ / ‘iob.h’ permite el control de las entradas digitales del variador. Estas dos cabeceras son obligatorias para el control de las funciones básicas del variador mediante funciones en el código fuente. [7]

```
/*=============================================
IPOS source file
=============================================*/
#include <const.h>
#include <io.h>
```

Las variables que se manejan son de 32 bits con signo, para referirse a ellas se les antepone la letra ‘H’ y se pueden almacenar hasta 1024, sin embargo a algunas variables de IPOS se les asignan funciones establecidas que se denominan variables del sistema (H453 - H560).
Mediante la directiva **“#define”** se pone una etiqueta a las variables.

```
#define setpoint H10
#define variable1 H12
#define minimum H11
```

Todas las variables son globales y se almacenan en el variador, existe una clasificación para éstas:

- **Variables iniciales (initials)**: son aquellas que se almacenan en una memoria no volátil de tal manera que quedan guardadas ante un corte de energía, sólo puede haber 128 de este tipo.
- **Variables auxiliares (var):** son aquellas que usa el programa para hacer cálculos auxiliares al realizar operaciones aritméticas, como mínimo se deben tener 10, según recomendación del fabricante, éstas se almacenan en una memoria volátil.
- **Variables globales (globals):** son aquellas que no son usadas por el variador para cálculos auxiliares, éstas se almacenan en una memoria volátil. [9]

Mediante la directiva **"#pragma"** se realiza la distribución de variables en la memoria del variador

Funciones

Funciones definidas por el usuario

El usuario puede programar funciones (subprogramas) a las cuales no se le transfiere argumentos, como sucede en la mayoría de los compiladores porque en IPOSPlus todas las variables son globales. La estructura de una función es la siguiente:

```
SampleFunction () {
 // Salir de la función si H1=5 si no definir H2=5
if ( H1 == 5 )
return;
H2=3;
}
```

Una vez realizada la función se continua con la línea siguiente de la tarea desde donde fue llamada la función.
Se puede salir de una función prematuramente usando la declaración de retorno ("*return;"*). En este caso, el programa continúa con la línea que sigue en la tarea principal.

Funciones para modificar bits

_BitClear(H, bit): pone un bit individual de una variable H a 0
_BitSet (H, bit): ajusta un bit individual de una variable H a 1

En el proyecto estos comandos se usan con la estructura ControlWord que como su nombre lo dice permite el control del sistema: su sentido de giro, habilitación de motor, velocidad fija, rampas de aceleración, etc. Puede cumplir la misma que función que las terminales físicas.

Tabla 4-4-2: definicion de funciones a ajustar con el comando ControlWord

ControlWord	Bit	Función al ajustar en "1"
	0	Sin función
	1	Sin habilitar
	2	Giro en sentido de las agujas del reloj
	3	Giro en contra del sentido de las agujas del reloj
	4	Velocidad fija n11
	5	Velocidad fija n12

Ejemplo:

```
_BitClear( ControlWord, 1 );  // ajusta en 0 la inhabilitación
_BitClear( ControlWord, 3 ); // ajusta en 0 el sentido de giro anti horario
_BitSet(ControlWord, 2) ;   // ajusta en 1 el sentido de giro horario
```

Funciones de posición

Estas funciones llevan al servomotor a una posición específica, esto es posible porque IPOS es un sistema de circuito cerrado con la retroalimentación del resolver.

_Go0

Este comando es la búsqueda del "home" así el servomotor hará un recorrido al punto de referencia. El argumento define el tipo de viaje a la referencia. La forma de búsqueda del cero, está relacionado también con la configuración del parámetro P903 de MOVITOOLS.

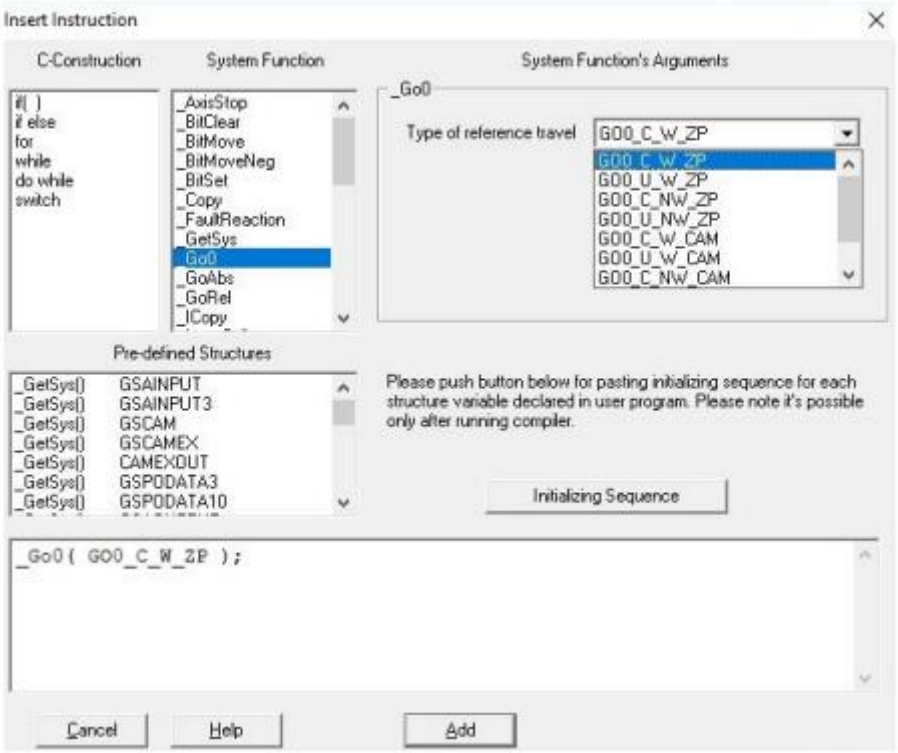


Figura 4-21: Estructura comando Go0

El significado de las letras son las siguientes [9]:

- C (conditional) = Viaje de referencia solo si el viaje de referencia aún no se ha realizado.
- U (Unconditional)= Siempre referenciado, independientemente de si el eje ya está referenciado o no.
- W (Wait)= Espera en esta línea de declaración hasta que se realice el viaje de referencia.
- NW (NoWait)= Procesa la siguiente línea de extracto durante el recorrido de referencia realizado.
- ZP (Zero Pulse) = Recorrido de referencia a pulso cero.
- CAM= Recorrido de referencia a cámara.

- RESET= El recorrido de referencia que se inició se interrumpe y la llamada se restablece.

– _GoAbs

Este comando permite el posicionamiento en función de un punto de referencia. La expresión para el tipo de comando de movimiento puede adoptar una de las siguientes opciones:

- GO_NOWAIT: Sin esperar se reanuda el procesamiento del programa en la siguiente línea de extracto inmediatamente después de enviar el comando de movimiento.
- GO_WAIT: Espera en esta línea de declaración hasta que se complete el viaje.

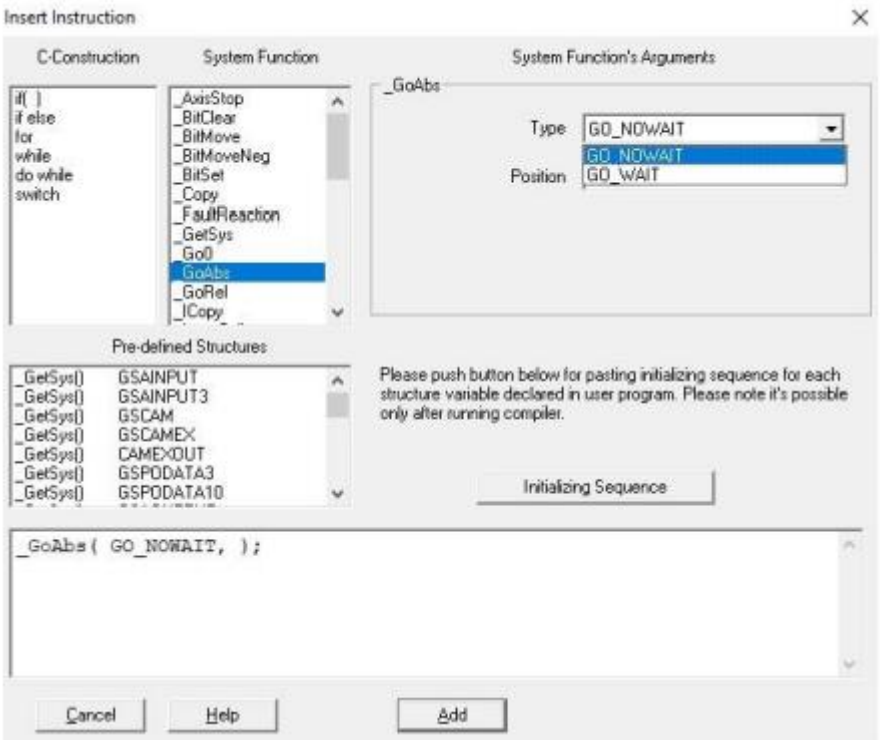


Figura 4-22: Estructura comando GoAbs

Estructuras estándares SEW (_SetSys/_MoviLink)

Son mascaras que SEW brinda al usuario para definir de forma sencilla la estructura de una variable. En este proyecto se usan las siguientes:

Tipo de instrucción	Estructura estándar
_MoviLink	MOVLNK
	MLDATA
_SetSys	SSPOSSPEED

Con el siguiente ejemplo se representa la forma de uso de este tipo de estructuras:

```
// Declaración de variables estructurales
SSPOSSPEED rapid speed, slow speed;

//Inicialización
 rapid_speed.cw = 14000; //velocidad rápida para el giro horario (cw)
rapid_speed.ccw = 12500; // velocidad rápida para el giro antihorario (ccw)
slow_speed.cw = 3000; // // velocidad baja para el giro horario (cw)
slow_speed.ccw = 4500; // velocidad baja para el giro antihorario (ccw)

 // se establece la velocidad alta para ambos sentidos de giro
_SetSys( SS_POSSPEED,rapid speed );

// // se establece la velocidad baja para ambos sentidos de giro
_SetSys( SS_POSSPEED,slow speed );
```

- **_GetSys(H, sys) :**escribe en una variable IPOS un valor interno del sistema.
- **_SetSys(H, sys)**: se le designa a un valor interno del sistema el valor de una variable IPOS.

Funciones para comunicación entre controladores:

_Movilink (H)

El comando MOVLNK [10] permite realizar cambios en los parámetros del variador y de cualquier otra unidad que pueda conectarse a través del bus del sistema o RS-485.

Cuando se llama al comando, MOVLNK lee y escribe datos de proceso, variables o parámetros de una unidad a otra una vez, dentro de una unidad una vez.

Los parámetros se leen/escriben mediante direccionamiento de índice. Los números de índice de las variables internas se pueden visualizar en MOVITOOLS® MotionStudio colocando el ratón en el cuadro de edición o campo de visualización del parámetro respectivo. Los números de índice de las variables de IPOS se componen por:

$$11000 + \text{número de variable de IPOS}$$

Por ejemplo, el índice de la variable H123=11123.

El comando Movilink (...) utiliza la variable H como argumento. Dicha variable se refiere a la estructura de un comando. En la estructura se debe introducir toda la información necesaria para la comunicación, contiene numerosas variables que seleccionan la interface, definiendo el tipo de transmisión y los datos.

En la siguiente tabla se explica la estructura del comando con ejemplos usados en el proyecto:

Tabla 4-3: definición de los valores MOVILINK usados para las variables H [9] [10]

Nº variable	Nombre de valores	Valores usados en el proyecto	Significado
H	Ml.BusType	ML_BT_SBUS1	Sirve para seleccionar la interface con la que se va a transmitir un comando MOVILINK®.
H+1	Ml.Address	1 (ver en P881 de Movitools)	La dirección SBus de la unidad. Cada unidad del sistema tiene que tener un valor distinto en P881 y el mismo valor en P882
H+2	Ml.Format	ML_FT_PAR ML_FT_1	Descripción de la estructura del mensaje. ML_FT_PAR: Los parámetros y 1 elemento con datos de proceso se transmiten usando un comando MOVLNK. ML_FT_1: datos de proceso se transmiten usando un comando MOVLNK.
H+3	Ml.Service	ML_S_RD ML_S_WR	Lectura de un parámetro mediante el mensaje de parámetros. Escritura, con memorización en la memoria no volátil.
H+4	Ml.Index	Número de variable	Número de índice del parámetro que se va a modificar o leer
H+5	Ml.DPointer	numof(MlD)	Nº de la variable H' en la que se almacenan los datos leídos o desde la que se obtienen los datos que se van a escribir. Siendo MlD la definición de la estructura MLDATA
H+6	Ml.Result	Código de fallo o 0	Contiene el cód. de fallo después de la ejecución del servicio, o contiene un cero si no se ha producido ningún fallo.

El DPointer apunta a la estructura de datos que puede inicializar numerosos valores, pero los que importan en este proyecto son:

Tabla 4-4: definición de los valores MLDATA usados para las variables H [9]

Nº variable	Nombre de valores	Significado
H'	Mld.Write Par	Contiene los datos para un servicio de escritura de parámetros
H' + 1	Mld.Read Par	Contiene los datos leídos por un servicio de parámetros.

Ejemplo en IPOSPlus escritura

```
/*=========================================
Se va a escribir la variable H200 del variador
Esclavo conectado vía SBUS
-Si DI17 = 0 del variador maestro, H200=-1000 en
el variador esclavo
-Si DI17 = 1 del variador maestro, H200=1000 en
el variador esclavo
=========================================*/
/*=========================================
IPOS Source file
=========================================*/
#include <constb.h>
#include <iob.h>
// Definición de estructuras MoviLink
MOVLNK tBus;
MLDATA tBusData;
/*=========================================
Main program
=========================================*/
main()
{
 // Inicializacion de MoviLink.
//Tiene que ser dentro del main ()
 tBus.BusType = ML_BT_SBUS1; // bus type SBus1
 tBus.Address = 10; // SBus address 10 del esclavo
 tBus.Service = ML_S_WRV; // escribir
 tBus.Index = 11200; // variable H200
 tBus.DPointer = numof(tBusData); // data buffer

 // Main program loop
 while(1)
 {
 if( DI17 )
 {
 tBusData.WritePar = 1000;
 _MoviLink( tBus );
 }
 else
 {
 tBusData.WritePar = -1000;
 _MoviLink( tBus );
 }
 }
}
```

5. Robot Planar 5 barras. Desarrollo matemático y estructural de los componentes reales.

La elección del mecanismo de 5 barras y su automatización para poner en funcionamiento los servomotores de la Escuela no fue al azar. Se optó por este mecanismo porque presenta ventajas de diferente índole para el proyecto: es pertinente para la automatización del movimiento del mecanismo de 5 barras para desarrollar un robots con los que se puede aprovechar la precisión por la que se destacan en la industria los servomotores SEW EURODRIVE. Es un mecanismo conocido por los estudiantes de ingeniería mecánica que permite utilizar la maqueta para realizar nuevos proyectos que profundicen el mejoramiento del control de movimiento sumamente importante en la disciplina. A través de su punto efector es fácil verificar la precisión del movimiento del mecanismo controlado. Y por último es un claro ejemplo de desarrollo sinérgico entre el control, la computación y la mecánica que es uno de los intereses del proyecto como aporte a nuevos contenidos disponibles para los estudiantes de la Escuela de Mecánica debido a que es muy utilizado en la industria, por ende es un problema real del proceso de trabajo con el que se encuentra el Ingeniero.

Robot Planar 5 barras

Los robots 5 barras, como se mencionó con anterioridad, están conformados por cuatro barras conectadas en cadena que se fijan en los extremos a dos motores; se denomina quinta barra a la distancia entre los motores.
Estos diseños presentan una dificultad posible de subsanar con un buen control: la introducción de los modos de trabajo y singularidades hace que la combinación de los ángulos de entrada no mantenga de manera constante el punto del efector y por ende varíe su posición poniendo en juego la confiabilidad del robot, tema que se abordará.

En este capítulo desarrollaremos los ejes elegidos para la construcción del robot, la cinemática inversa del mecanismo de 5 barras y el diseño de las piezas del prototipo y su fabricación.

Cinemática de posición

La cinemática como estudio del movimiento puede ser abordada desde dos perspectivas: a) determinando " *la posición y orientación del extremo final del robot, con respecto a un sistema de referencia, conocidos los valores de las articulaciones (ángulos) y los parámetros geométricos de los elementos del robot, cinemática directa*", y b) la cinemática inversa, que se utiliza en el proyecto, "*consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot ….para que su extremo se posicione y oriente según una determinada localización espacial*" (Fundamentos de Robótica Cinemática Inversa Ricardo-Franco Mendoza-García rmendozag@uta.cl Escuela Universitaria de Ingeniería Mecánica Universidad de Tarapacá Arica, Chile June 16, 2015)

Durante el estudio de la cinemática del mecanismo existe un problema real: los modos de trabajo y las singularidades, debido a su gran importancia se abordan y estarán presente constantemente en el proyecto .

Cinemática inversa de posición

[3]Para el análisis de la cinemática del mecanismo es necesario definir los ángulos principales, las barras con sus longitudes y un eje de coordenadas que se muestran en la siguiente figura:

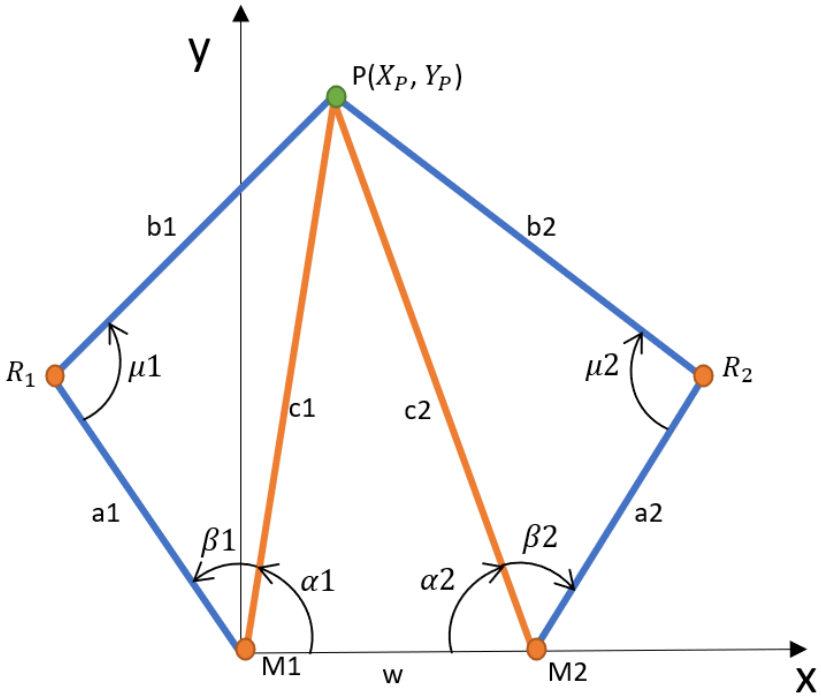


Figura 5-1- Notación usada en la configuración del sistema de 5 barras

La figura presenta la nomenclatura utilizada para el análisis cinemático del mecanismo. El sistema de referencia inercial se ubica en el punto M1, los puntos R1, R2, M2 y P se emplean para definir la ubicación de las juntas rotacionales. Además, el punto P define la ubicación deseada del mecanismo, es decir, la posición del efector final y en los puntos M1 y M2 se encuentran los motores o actuadores del sistema (ángulos de entrada).

Ángulos de actuadores

El mecanismo de cinco barras tiene 2 grados de libertad, por lo tanto es necesario controlar dos actuadores para definir la posición del efector. La cinemática inversa se encarga de calcular los valores de las juntas actuadoras, $(\alpha_i + \beta_i)$, en función de la posición deseada del punto P y de los demás parámetros del mecanismo. Esta se resuelve geométricamente de manera sencilla usando la fórmula de distancia y la ley de cosenos. [3]

- Usando la fórmula de la distancia, ci se puede expresar de la siguiente manera:

$$c_i = \sqrt{(x_p - x_{Mi})^2 + (y_p - y_{Mi})^2} \tag{4}$$

- Usando la ley de los cosenos, α y β se pueden expresar de la siguiente manera:

$$\alpha i = \arccos\left(\frac{c_j^2 - c_i^2 - w^2}{-2c_i w}\right) \tag{5}$$

$$\beta i = \arccos\left(\frac{b_i^2 - a_i^2 - c_i^2}{-2a_i c_i}\right) \tag{6}$$

Definiendo que M1 en el origen de coordenadas y M2 en (w,0) como se ve en la figura Figura 5-1

$$c_1^2 = \left(x_p - x_{M1}\right)^2 + \left(y_p - y_{M1}\right)^2 = x_p^2 + y_p^2 \tag{7}$$

$$\begin{aligned} c_2^2 &= \left(x_p - x_{M2}\right)^2 + \left(y_p - y_{M2}\right)^2 \\ &= \left(x_p - w\right)^2 + y_p^2 \\ &= x_p - 2x_p w + w^2 + y_p^2 \\ c_2^2 &= c_1^2 - 2x_p w + w^2 \end{aligned} \tag{8}$$

Reemplazar la Ecuación (*8*) en la Ecuación (*5*) por i = 1 nos permite simplificar nuestra expresión para α1:

$$\begin{aligned} \alpha 1 &= \arccos\left(\frac{c_2^2 - c_1^2 - w^2}{-2c_1 w}\right) \\ \alpha 1 &= \arccos\left(\frac{c_1^2 - 2x_p w + w^2 - c_1^2 - w^2}{-2c_1 w}\right) \\ \alpha 1 &= \arccos\left(\frac{x_p}{c_1}\right) \end{aligned} \tag{9}$$

La expresión para α2 se simplifica de manera similar

$$\begin{aligned} \alpha 2 &= \arccos\left(\frac{c_1^2 - c_2^2 - w^2}{-2c_2 w}\right) \\ \alpha 2 &= \arccos\left(\frac{c_1^2 - (c_1^2 - 2x_p w + w^2) - w^2}{-2c_2 w}\right) \\ \alpha 2 &= \arccos\left(\frac{-x_p + w}{c_2}\right) \end{aligned} \tag{10}$$

Para βi, no surgen simplificaciones de la manipulación algebraica, por lo que se mantiene (Ecuación.6).

Ángulos de transmisión de movimiento

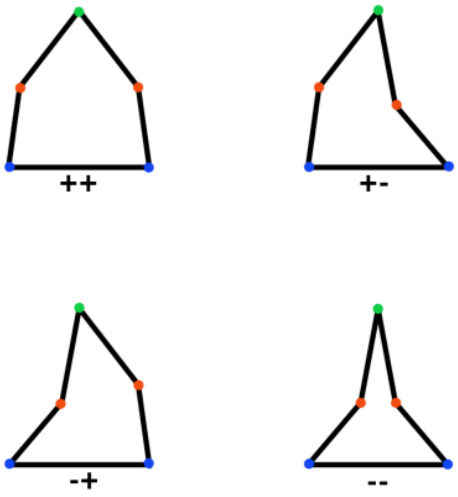
El ángulo de transmisión, μi, es el ángulo entre el enlace impulsor, a_i , y el enlace flotante al que conduce, b_i , como se ve en Figura 5-1. El mecanismo de 5 barras (5BL) tiene dos ángulos de transmisión, uno para cada brazo impulsor. El ángulo de transmisión es crucial en la transmisión de fuerza a través de los enlaces para mover el efector final. [3]

Los ángulos de transmisión se pueden encontrar de manera similar a βi usando la ley de los cosenos

$$\mu i = \arccos\left(\frac{c_i^2 - a_i^2 - b_i^2}{-2a_i b_i}\right) \quad (11)$$

Modos de trabajo

Toda posición final puede lograrse a partir de cuatro combinaciones de ángulos motores (θ_i) que corresponden a cuatro modos de trabajo distintos.
Se conoce como modos de trabajo a las áreas donde los brazos del sistema ($\overline{M_i R_i P}$) mantienen en el movimiento su forma respecto a la cuerda (ci) que une el efector final (P) con el motor del lado (Mi). Esta forma puede ser convexa (+) o cóncava (-). Como cada brazo puede lograr dos configuraciones distintas, el sistema logra los cuatro modos de trabajo. En la siguiente figura se observan los cuatro modos de trabajo para una posición del efector final y en sus referencias se observan las reglas para combinar los ángulos intermedios (α_i y β_i) que producen ángulos motores (θ_i) de acuerdo con un Modo de Trabajo específico. [4] [3] [2]



[3]

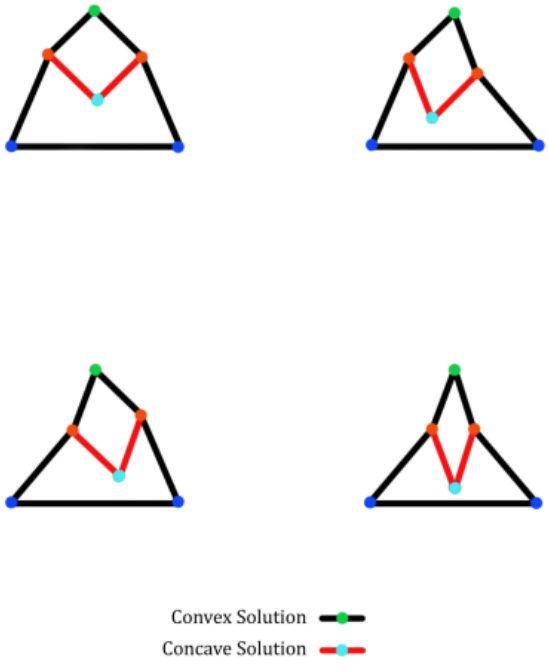
MODO DE TRABAJO	θ_1	θ_2
++	$\alpha_1 + \beta_1$	$\alpha_2 + \beta_2$
+-	$\alpha_1 + \beta_1$	$\alpha_2 - \beta_2$
-+	$\alpha_1 - \beta_1$	$\alpha_2 + \beta_2$
--	$\alpha_1 - \beta_1$	$\alpha_2 - \beta_2$

Figura 5-2- Combinación de los ángulos intermedios (α y β) para producir

ángulos motores (θ) de acuerdo con un Modo de Trabajo específico.

Debido a que, para pasar de un modo de trabajo a otro se deben atravesar singularidades paralelas (desarrollado en el apartado siguiente), es necesario elegir alguno de estos modos expuestos a la hora de resolver el problema.

De manera similar, el mecanismo de 5 barras tiene dos modos de ensamble, es decir, hay dos posibles soluciones del efector final (P) para cada par de ángulos actuadores (θ_1, θ_2) . Al pasar de un modo de ensamble a otro ocurre una singularidad en serie (explicada en el apartado anterior) la cual genera desconfianza en el control del mecanismo, por esta razón siempre se quiere evitar. Es por esto, que se trata de mantener el modo de ensamble durante el funcionamiento del mecanismo



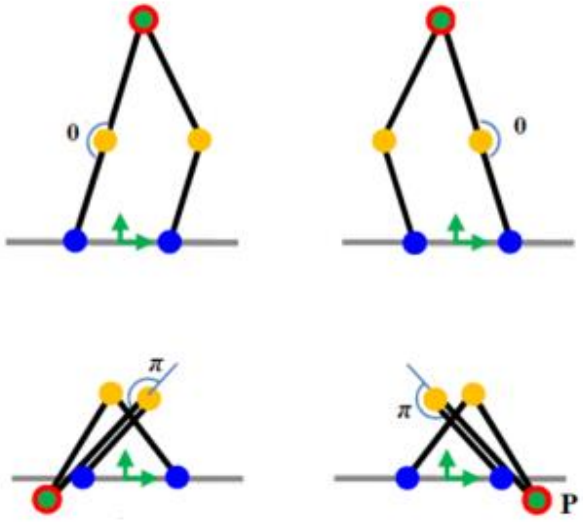
[3]

Figura 5-3- Los dos modos de ensambles para cada uno de los cuatro modos de trabajo.

Singularidades

Las configuraciones singulares son posiciones en las cuales el extremo del robot pierde o gana al menos un grado de libertad.

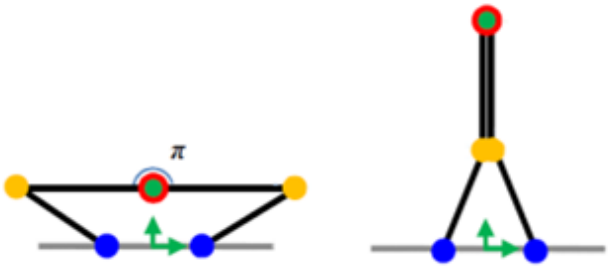
Si el robot pierde un grado de libertad (singularidad en serie), el extremo se ve bloqueado ante ciertos pares de los actuadores. Esto dibuja una trayectoria que es básicamente el límite del espacio de trabajo de un robot paralelo y ocurre cuando un brazo está completamente estirado (es decir, los puntos Ri, Mi y P son colineales pero los puntos Mi y P no coinciden, siendo i = 1, 2) o completamente doblado (ai coincide con bi). [4] [2]



[3]

Figura 5-4: Tipos de singularidades en serie

En el caso de que gane un grado de libertad (singularidad en paralelo) el extremo del robot puede moverse aunque los actuadores estén bloqueados, esto implica que no puede resistir ciertas fuerzas o momentos sobre su extremo. Este tipo de singularidades se encuentran dentro del espacio de trabajo y ocurre cuando los enlaces distales son colineales (es decir, los puntos R1, P y R2 son colineales).



[3]

Figura 5-5: Tipos de singularidades en paralelo

Es sabido que la industria está en constante búsqueda de aumentar el área de trabajo de los robots paralelos. Habitualmente se trata optimizar el diseño del robot eliminando las singularidades paralelas ya que en este caso no se tiene un control confiable de la posición final del efector, pero también existe otro enfoque que deja las singularidades y hace navegar el mecanismo dentro de la región libre de ellas. [4] [2] [8]

Si se requiere un movimiento continuo preciso, como en el mecanizado, se deben evitar todos los tipos de singularidades utilizando uno de los enfoques anteriores. Sin embargo, si solo las posiciones discretas del efector final deben ser precisas, como en el manejo de materiales, la ruta entre dos posiciones no es de interés y, por lo tanto, puede involucrar singularidades. En este caso el problema es cómo permitir que un robot paralelo navegue a través de singularidades para hacer un uso óptimo de su espacio de trabajo, esto se logra usando fuerzas externas como la inercia.

Aplicación de Ecuaciones Cinemáticas Inversas en MATLAB

Definidas las constantes de las ecuaciones desarrolladas que son las longitudes de las 5 barras y el conjunto de posiciones del efector final, las ecuaciones se usan en MATLAB para definir los ángulos de los motores que logran las posiciones adecuadas resolviendo los siguientes pasos para cada movimiento:

1. Resuelva las Ecuaciones para c1 y c2.
2. Resuelva las Ecuaciones para $\alpha 1$ y $\alpha 2$.
3. Resuelva la Ecuación para $\beta 1$ y $\beta 2$.
4. Para obtener el conjunto de ángulos del motor, $\theta 1$ y $\theta 2$, seleccione un Modo de trabajo y combine αi y βi de acuerdo con la Tabla de la Figura 5-2. [3]

Ver Anexo A: Programa de Matlab

Diseño del modelo físico

Para las instancias anteriores fue suficiente con definir el mecanismo de barra: 5 barras con dos grados de libertad, pero para poder hacer real el control del programa es necesario el diseño del prototipo.

El prototipo es de creación propia y las piezas se fabrican por impresión 3D por su bajo costo y accesibilidad, este prototipo tiene que cumplir con los siguientes requerimientos:

- Lograr que los brazos giren libremente uno sobre el otro
- Permitir el giro entre los nodos y reducir el juego entre ellos lo máximo posible teniendo en cuanta que la fabricación es de baja precisión con impresora 3D
- Evitar interferencias mecánicas entre los brazos del robot.
- Definir posición del cero maquina con el uso de fines de carrera.

Una condición que se debe tener en cuenta al momento de diseñar el robot es que los servomotores, al ser de uso industrial, son extremadamente pesados y por esta razón se opta por posicionarlo perpendicular al piso.

Longitudes de barras

Para el desarrollo del proyecto se definió que las dos barras ensambladas al motor tendrán la misma longitud *(200mm)* como así también las barras flotantes *(250mm).* La quinta barra que une los dos motores tiene una longitud de *450mm* para evitar que las barras unidas al motor choquen ente sí al estar posicionadas en forma horizontal.

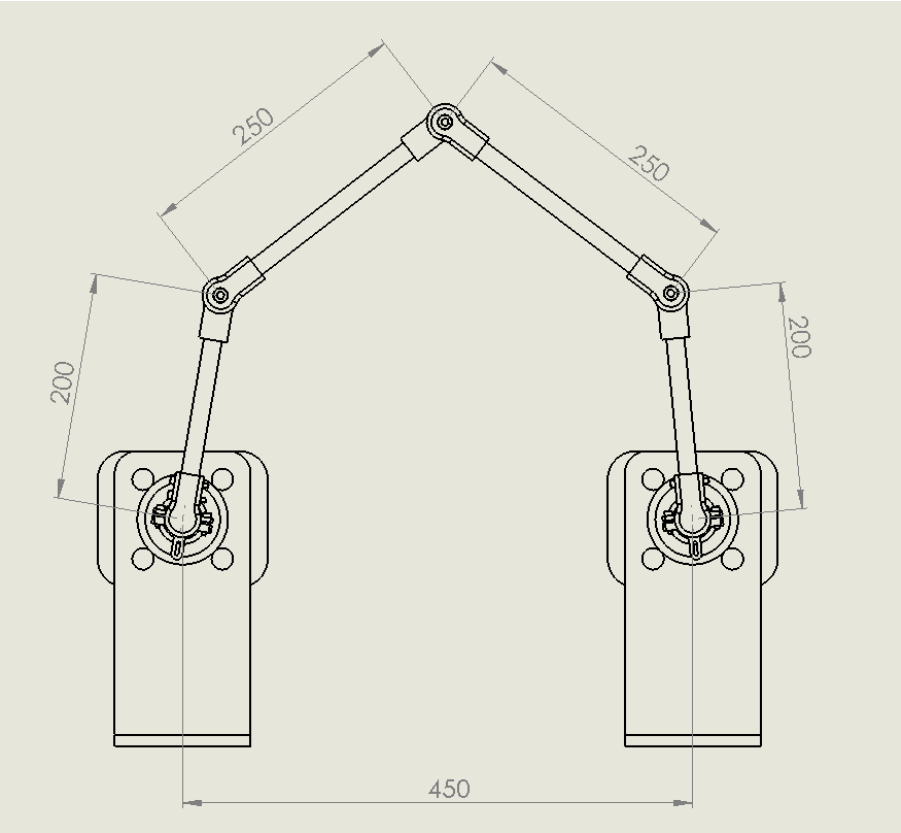


Figura 5-6: Distancias entre centros de las piezas del Robot 5 barras (2GDL)

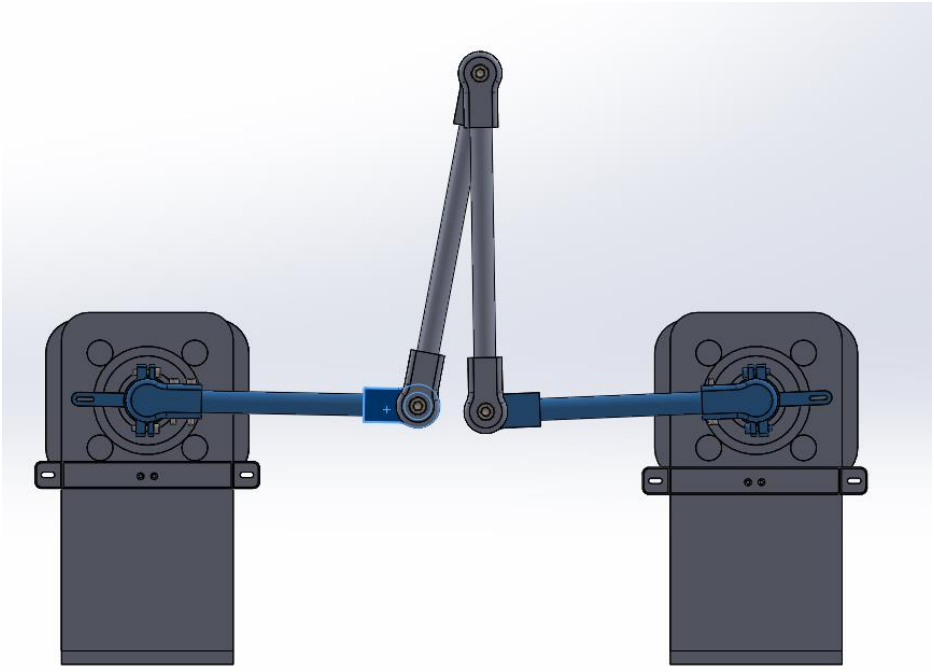


Figura 5-7: Distancia entre centros de motores que evita colisiones mecánicas entre barras

Diseño de articulaciones del robot

Hay dos tipos de articulaciones: la que trasmite el movimiento rotacional del motor a los brazos (hay dos de este tipo) y otra que trasmiten el movimiento entre dos barras, teniendo en cuenta que cada brazo está formado por dos barras y que el robot tiene dos brazos unidos (tres de este tipo).

La primera articulación permite la perpendicularidad del robot al piso y está constituida por dos piezas tipo hembra-macho bloqueadas entre si a través de un apriete con tornillos en la pieza hembra, a su vez la pieza macho está encastrada al eje del motor para imitar el movimiento del mismo. Como hay dos motores existen dos articulaciones de este tipo como se dijo anteriormente.

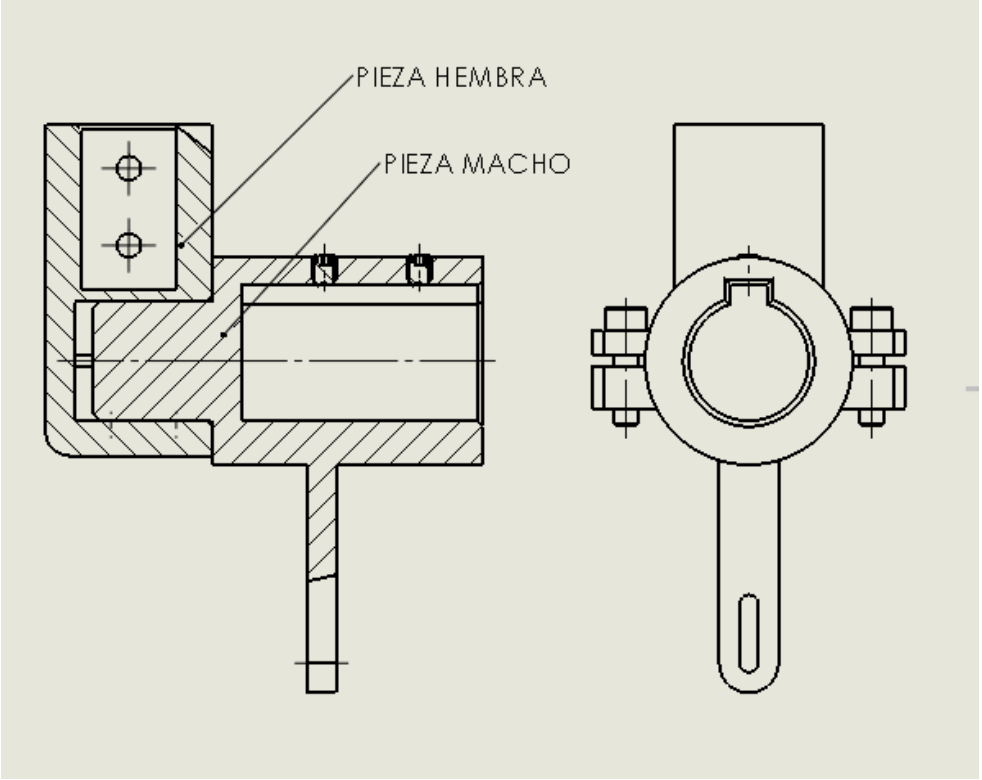


Figura 5-8: Subconjunto acople barra-motor. La pieza hembra encastra en la pieza macho y tiene un cubo para la barra. La pieza macho es apretada por la pieza hembra con el uso de tornillos y tiene un cubo para el eje del motor.

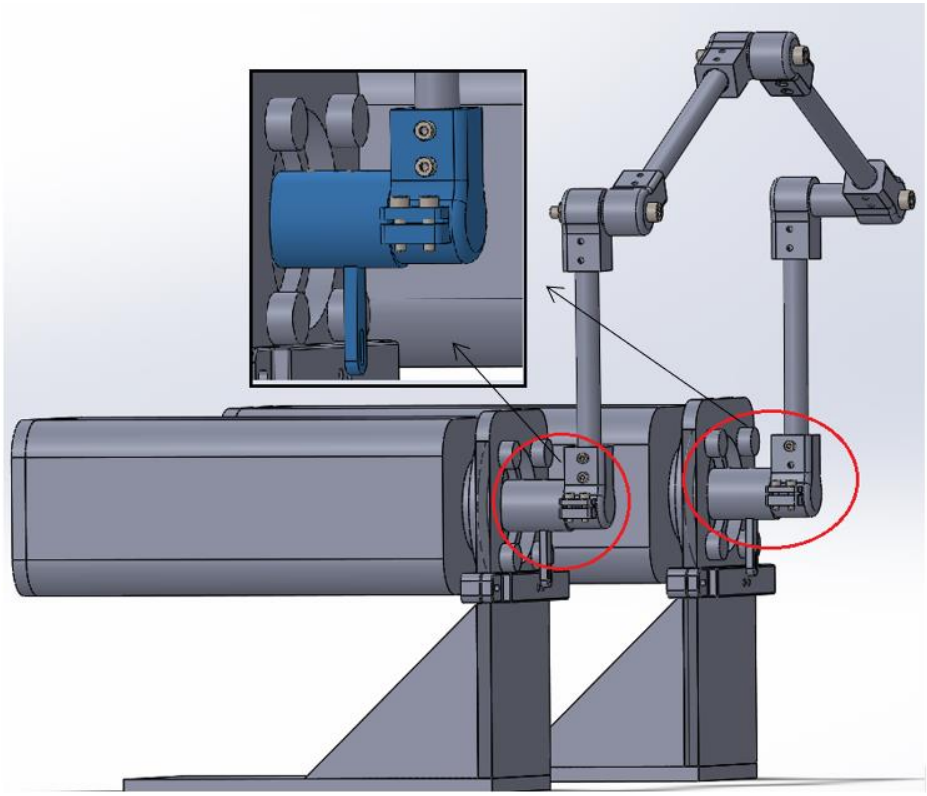


Figura 5-9: Subconjunto acople barra-motor

La pieza que se encastra al motor tiene una pestaña que se usa para el posicionamiento del robot en su cero al tocar la solapa con el fin de carrera.

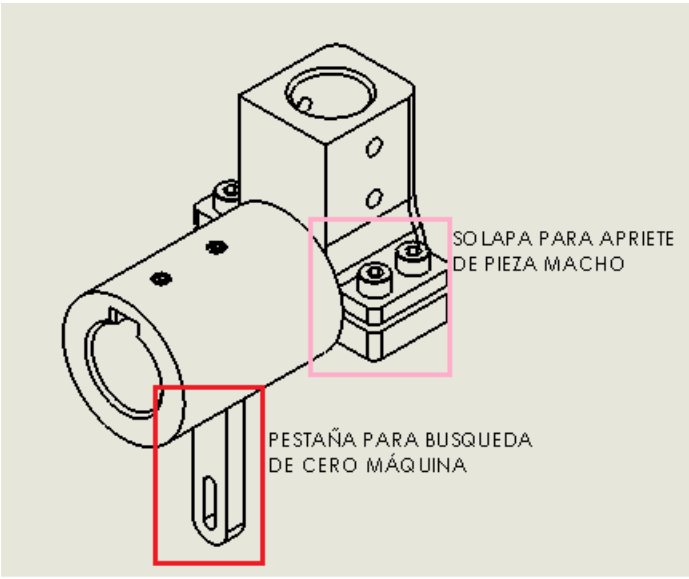


Figura 5-10: pestaña de acople barra-motor para posicionar las barras en el cero máquina

La segunda articulación está conformada por tres piezas, dos de las cuales son los nodos de las barras y la tercera es un buje separador que posibilita el giro libre entre las barras. Estas piezas se mantienen unidas por un tornillo eje que las atraviesa y una tuerca que les hace de tope.
Los bujes separadores tienen longitudes distintas para lograr que los brazos se encuentren en distintos planos y no colisionen. Esta piezas cumplen la función de “codos” de los brazos y de unión entre los dos brazos en el punto efector.

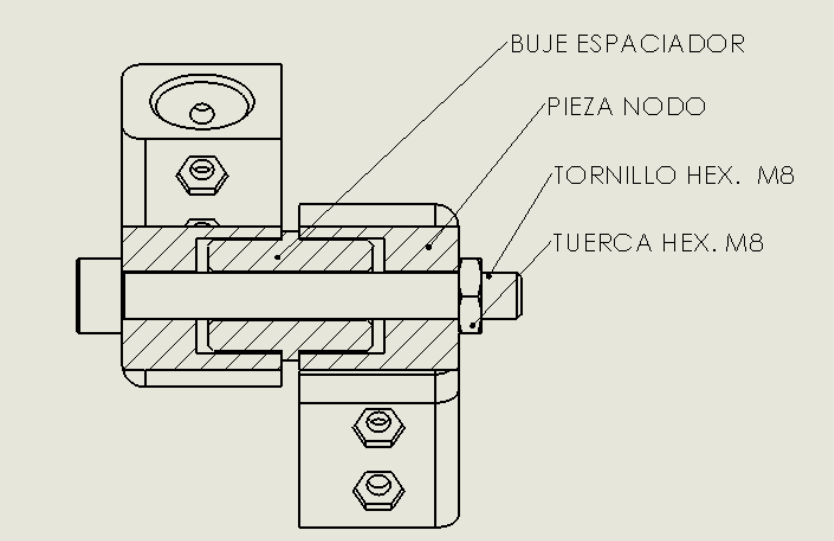


Figura 5-11: Articulacion barra-barra

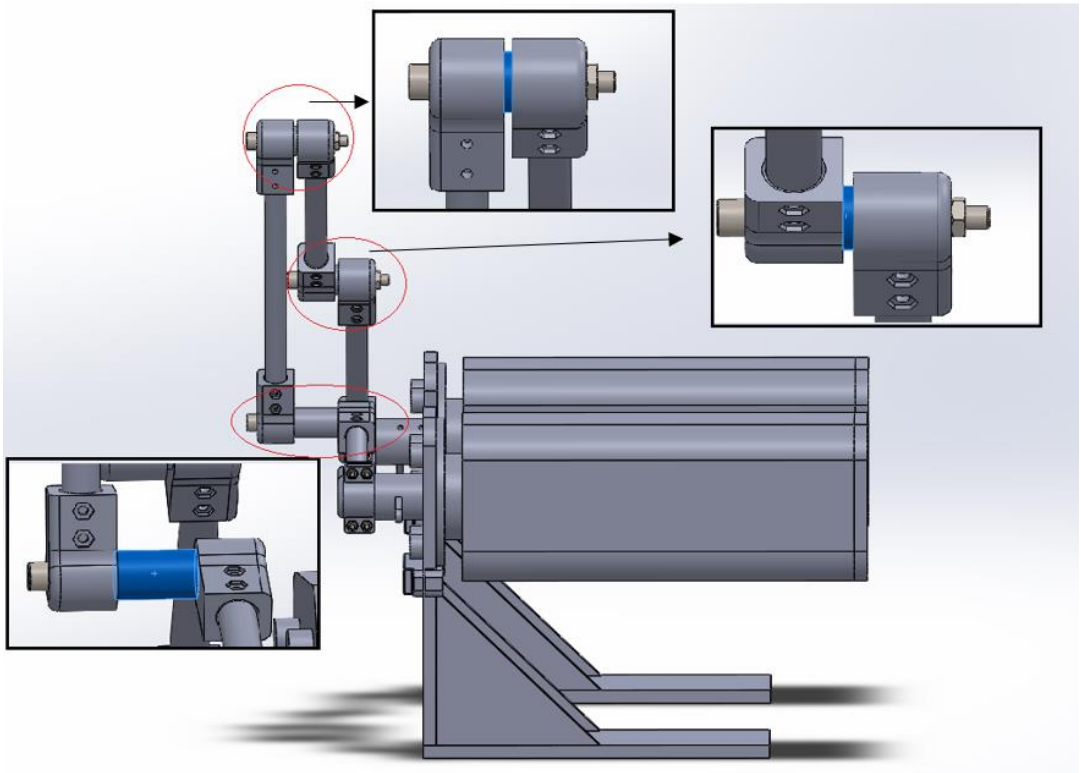


Figura 5-12: Localización de los 3 bujes separados del sistema

Cero maquina

El cero máquina se logra con dos límites de carrera, uno para cada motor, el perteneciente al motor izquierdo está dispuesto de forma tal que para su encuentro dicho giro se realiza en el sentido de las agujas del reloj, mientras que el perteneciente al motor derecho esta presentado para que el giro de la búsqueda de la referencia sea en el sentido contrario a la aguja del reloj. Esto permite que en el recorrido del mecanismo para llegar a su cero los brazos se acerquen entre si evitando en todo momento que haya una separación entre ellos.

En la posición cero del prototipo las barras unidas al motor están posicionadas horizontalmente en el espacio que queda entre los motores y las barras flotantes están formando un ángulo hacia arriba determinando así el modo de ensamble en el que se desea trabajar, imitando el sentido de giro de los actuadores definidos en la resolución de la cinemática inversa.

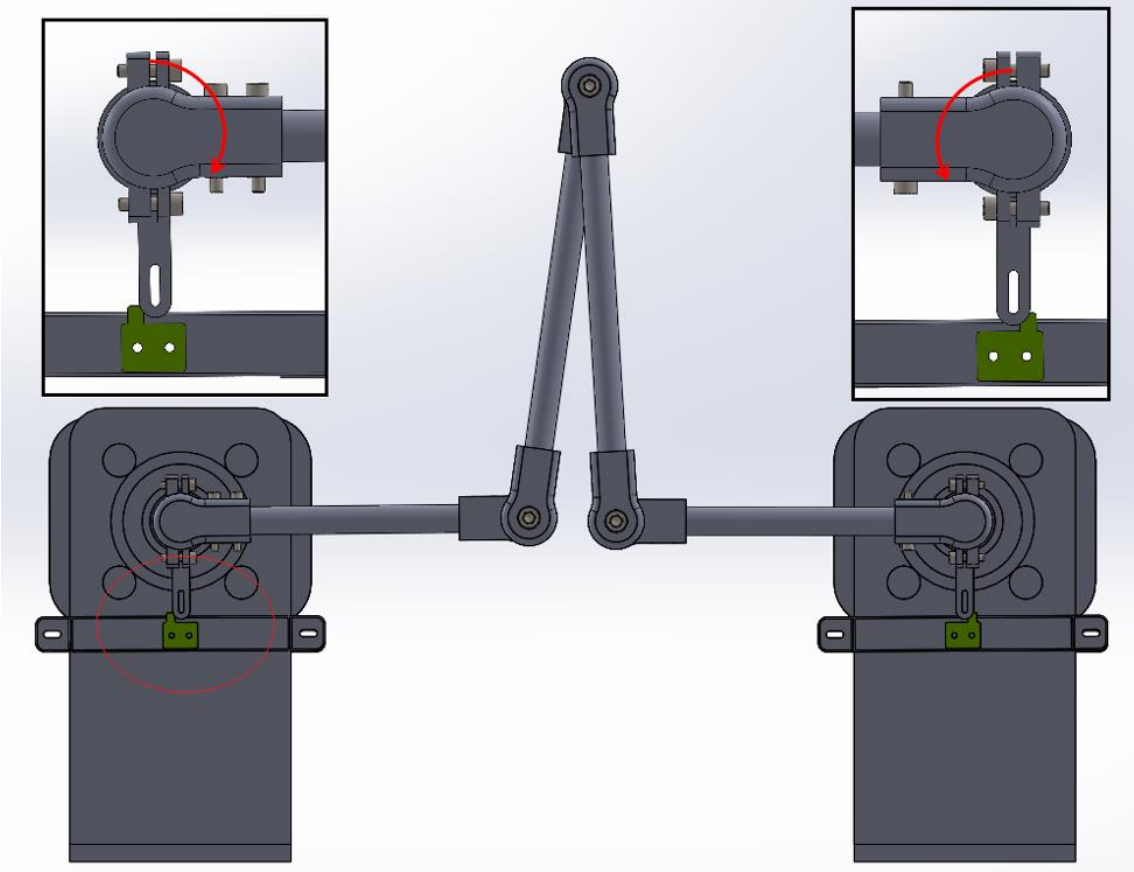


Figura 5-13: Cero maquina

6. Sistema de Control de robot 5 barras

Control del sistema

El proyecto requiere la automatización completa de las tareas por lo tanto es necesaria una comunicación constante entre los dispositivos involucrados, un programa que determine las tareas a realizar por cada motor y una interfaz por medio de la cual los usuarios puedan indicarle que hacer al sistema.

Comunicación

Las conexiones industriales más modernas tipo “ethernet” para el control y la comunicación de los dispositivos usados (PC, MOVIDRIVE B) no son posibles en este proyecto ya que los controladores disponibles no cuentan con los opcionales físicos necesarios .

Por esta razón, el control en tiempo real se logra por medio de una comunicación serial: tipo bus de sistema (SBUS) para la comunicación entre DRIVERS, o como se explica con anterioridad por interfaz RS485 para la comunicación PC-DRIVERS.

Comunicación entre controladores

Cada servomotor SEW está conectado a su MOVIDRIVE B quien le entrega las ordenes de movimiento y donde está compilado el programa IPOSPlus correspondiente.
Sin embargo, el proyecto requiere un sincronismo en el movimiento de los motores que se logra por la comunicación entre sus drivers en tiempo real por medio de la interfaz CAN (SBus). Esta se conecta por cables a través de la bornera X12 como se ve en la imagen y necesita conectar la resistencia de terminación del bus de sistema (S12 = ON) de ambos drivers ya que serán el comienzo y el final de la conexión del bus de sistema.

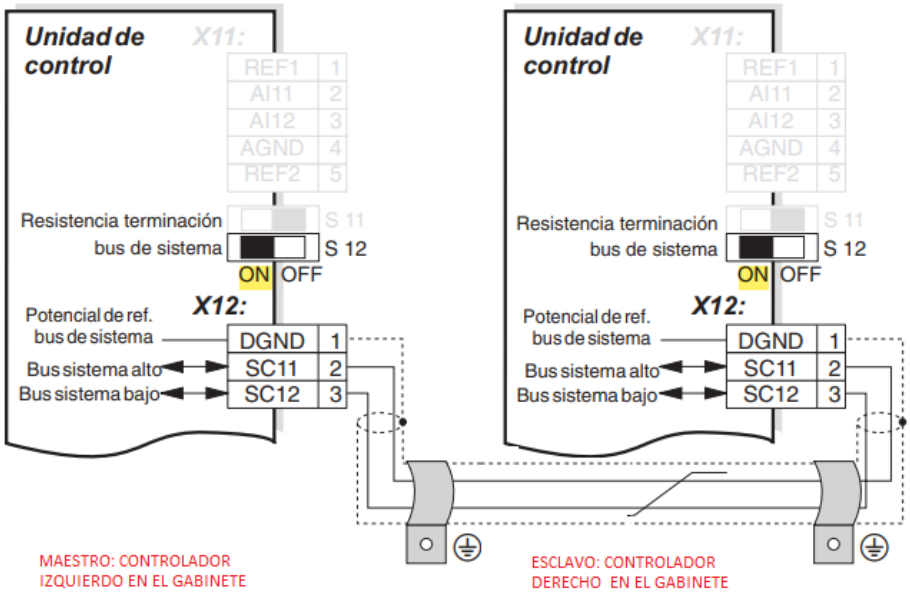


Figura 6-1: Conexión por cable para la comunicación SBus

Para el intercambio de datos en serie se usa el protocolo MOVILINK con el que se puede intercambiar mensajes que contienen datos relativos a los parámetros y al proceso. La unidad MOVIDRIVE puede llevar a cabo la comunicación actuando bien como maestro o bien como esclavo.
Como maestro, la unidad puede iniciar de forma activa un intercambio de mensajes con datos relativos a los parámetros y al proceso utilizando un programa IPOSPlus con el comando MOVLNK. Como esclavo, la unidad puede recibir mensajes con datos relativos a los parámetros y al proceso a través del SBus y darles respuesta.

Implementación del sistema informático IPOSplus

En este proyecto se desarrollan dos programas distintos para cada controlador teniendo en cuenta el modo de trabajo MAESTRO-ESCLAVO. De forma tal que el controlador maestro (el posicionado a la izquierda) defina el momento de inicio de tareas, el fin de ellas, el modo de trabajo y también es el que está conectado a la PC al momento de la ejecución de tareas, ya que la PC trabaja como interfaz humano-maquina. Por el contrario, el esclavo (controlador posicionado a la derecha) está a la espera de instrucciones para ejecutar el movimiento indicado.

Especificaciones técnicas

Los programas que fueron creados juntos cumplen una secuencia de movimientos coordinados entre los motores para lograr las posiciones finales del robot. Asimismo, son capaces de llevar al sistema a su posición inicial (cero maquina), variar los modos de trabajo entre automático y paso a paso (hacia delante y hacia atrás) y parar el movimiento de los motores cuando sea indicado.

La secuencia de estas tareas se desarrolla mediante el software IPOSPlus, presentado en el

Desarrollo de contenidos básicos del Software MOVITOOLS® MotionStudio.

Programa IPOSPlus Controlador MAESTRO

Definición de estructuras y variables

Este programa tiene estructuras MOVLINK y MLDATA ya que es el encargado de leer y escribir indicaciones en el controlador esclavo.

WR_ST2: se usa para escribir datos en la variable ST2 (H133) que define el estado en que se encuentra el variador esclavo.
0: en movimiento
1: listo para la próxima acción
2: a la espera de que el variador maestro le confirme que está listo
3: perdido, no conoce la posición inicial

WR_DITA2: su usa para escribir datos en la variable DITA2 (H131) que define en pulsos el ángulo de giro que tiene que realizar el motor derecho.

RD_ST2: lee la variable ST2 para saber si el motor derecho está listo para el siguiente movimiento o indicación.

```
//================================CONTROLADOR IZQUIERDO================

/*=============================================
            IPOS Source File
=============================================*/
#include <constb.h>
#include <iob.h>
//#pragma initials 0 127
#pragma globals 700 800
//#pragma var 700 800


MOVLNK WR_ST2;
MLDATA WR_ST2D;
MOVLNK WR_DITA2;
MLDATA WR_DITA2D;
MOVLNK RD_ST2;
MLDATA RD_ST2D;
```

Figura 6-2: definición de estructuras

Se puede cargar 4 posiciones de cada motor y estas quedan grabadas en las variables que se ven en la Figura 6-3. Siendo servo 1, el servomotor del variador maestro y servo 2, el del variador esclavo.

```
//DEFINICION DE ANGULOS


#define dital_1 H17 // angulo de giro de servo 1 de posicion 1
#define dita2_1 H18 // angulo de giro de servo 2 de posicion 1
#define dital_2 H19 // angulo de giro de servo 1 de posicion 2
#define dita2_2 H20 // angulo de giro de servo 2 de posicion 2
#define dital_3 H21 // angulo de giro de servo 1 de posicion 3
#define dita2_3 H22 // angulo de giro de servo 2 de posicion 3
#define dital_4 H23 // angulo de giro de servo 1 de posicion 4
#define dita2_4 H24 // angulo de giro de servo 2 de posicion 4
```

Figura 6-3: Definición de variables H para ángulos dita

Por último, se definen las variables que guían el programa en sus distintos modos y estaciones.

```
// ESTADOS, MODOS y POSICIONES

#define home H129// perdido( estaod inicial) lost=1, after homming lost=0  (SERVO 1)
#define dital H130
#define dita2 H131
#define st1 H132 // estado servo 1 ( moving=0; ready=1; finish=2; lost=3)
#define st2 H133 // estado servo 2 ( moving=0; ready=1; finish=2; lost=3)
#define stop H134 //parada
#define n H135 //numero de posicion (SERVO 2)
#define AUTO H136 // automatico;
#define PAP H137 //paso a paso
#define BB_N H138 //boton para posicion siguiente en PAP
#define BB_P H139 // boton para posicion anterior en PAP
#define pos_sig H140 // memoria para posicion siguiente
#define pos_ant H141 // memoria para posicion anterior
```

Figura 6-4: Definición de variables H para modos y estados

Subtareas

En el programa existen dos taras GiroAng1() y Homming. En ellas se escribe y leen variables del drive esclavo.

- GiroAng1 () es llamado por el programa cuando se quiere una nueva posición de los servomotores.

```
AngGirol()
  {
// SE ESCRIBE  A SERVO 2 QUE SE MUEVA (st2=0) EL ANGULO DITA2
WR_DITA2D.WritePar=dita2;
_MoviLink(WR_DITA2);
st2=0;
WR_ST2D.WritePar=st2;
_MoviLink(WR_ST2);
//MOVIMIENTO SERVO 1 A DITA1
st1=0; // ==> moving
_GoAbs( GO_WAIT,dital );
st1=2;
// SE LEE EL ESTADO DE SERVO 2 HASTA QUE SE DEJE DE MOVER (sta2=2)
while( st2 !=2 ) {
 _MoviLink( RD_ST2 ); // actual command call
st2=RD_ST2D.ReadPar;
 } //end while
// AMBOS SERVOS LISTOS PARA LA PROX ACCION.SE ESCRIBE A SERVO2 QUE ESTA LISTO
st1=1;
st2=1;
WR_ST2D.WritePar=st2;
_MoviLink(WR_ST2);
} //end AngGirol
```

Figura 6-5: Subtarea Ángulo de Giro

- Homming () se llama cuando se requiere encontrar el 0 del sistema (posición inicial). En el proyecto las posiciones iniciales son detectadas por fines de carrera.

```
Homming()
{
 // ENTREGA A SERVO2 ESTADO INCIAL PERDIDO
WR_ST2D.WritePar=st2;
_MoviLink(WR_ST2);
//MOVIMIENTO SERVO 1 A HOME
st1=0; // ==> moving
_Go0( GO0_U_W_ZP );
st1=2;
// SE LEE EL ESTADO DE SERVO 2 HASTA QUE SE DEJE DE MOVER (sta2=2)
while( st2 !=2 ) {
 _MoviLink( RD_ST2 ); // actual command call
st2=RD_ST2D.ReadPar;
 } //end while
// AMBOS SERVOS LISTOS PARA LA PROX ACCION.SE ESCRIBE A SERVO2 QUE ESTA LISTO
st1=1;
st2=1;
WR_ST2D.WritePar=st2;
_MoviLink(WR_ST2);
_Wait(500);
//HOME VUELVE A 0
home=0;
} //end homming
```

Figura 6-6: Subtarea Homming

Programa principal

Al energizar los variadores se deben reiniciar todas las variables y habilitar el movimiento del servomotor.

```
main() {

//CONCIONES INCIALES
st1=3; //estado inicial perdido
st2=3; //estado incial perdido
home=0;
n=0;
stop=0;
AUTO=0;
PAP=0;
BB_P=0;
BB_N=0;
pos_sig=0;
pos_ant=0;
dital=0;
dita2=0;

//HABILITACION DE SERVOMOTOR Y DEFINICION SENTIDO DE GIRO
_BitClear( ControlWord, 0 ); //inhibicoion
_BitClear( ControlWord, 1 ); //inhibicoion
_BitClear( ControlWord, 3 ); //sentido CCW deshabilitado
_BitSet(ControlWord,2) ;    //sentido CW  habilitado
```

Figura 6-7: Condiciones iniciales del programa principal. Habilitación del controlador

Se crean las estructuras MOVLNK que son usadas dentro de las subtareas. Es necesario que estas estén creadas dentro del programa principal “main ()”.

```
// SE CREAN ESTRUCTURAS MOVLNK PARA LA COMUNICACION

WR_ST2.BusType = ML_BT_SBUS1; // SBUS
WR_ST2.Address = 1;    // direccion sbus de variador esclavo
RD_ST2.Format = ML_FT_PAR; // 2 PD with parameter
WR_ST2.Service = ML_S_WRV; // WRITE
WR_ST2.Index = 11133; // Indice de variable H133 de SERVO2
WR_ST2.DPointer =numof(WR_ST2D); // puntero hacia la estructura MLDATA

WR_DITA2.BusType = ML_BT_SBUS1; //SBUS
WR_DITA2.Address = 1; // direccion sbus de variador esclavo
WR_DITA2.Format = ML_FT_PAR; // 2 PD with parameter
WR_DITA2.Service = ML_S_WRV; // WRITE
WR_DITA2.Index = 11131; // Indice de variable H131 de SERVO2
WR_DITA2.DPointer =numof(WR_DITA2D); //  puntero hacia la estructura MLDATA

RD_ST2.BusType = ML_BT_SBUS1; //SBUS
RD_ST2.Address = 1; // direccion sbus de variador esclavo
RD_ST2.Format = ML_FT_PAR; // 2 PD with parameter
RD_ST2.Service = ML_S_RD; // Read
RD_ST2.Index =11133; // Indice de variable H133 de SERVO2
RD_ST2.DPointer =numof(RD_ST2D);//  puntero hacia la estructura MLDATA
```

Figura 6-8: Definición estructuras MOVILINK

Luego comienza el programa cíclico que está dividido en cuatro secciones:

- Homming: si el servomotor se encuentra en movimiento no se ejecuta la subtarea. Si se logra ejecutar, todas las variables van a 0 y se espera hasta que se indique el nuevo modo de trabajo (automático o paso a paso) o se pida una parada.

```
//PROGRAMA PRINCIPAL CICLICO
while (1) {

//================ HOMMING ==============================
if (st1==0 && (home==1)) home=0;
if ((st1!=0)&& (st2!=0) &&(home==1)) {
st1=3; //estado inicial perdido
st2=3; //estado incial perdido
Homming ();
n=0;
AUTO=0;
PAP=0;
BB_P=0;
BB_N=0;
pos_sig=0;
pos_ant=0;
dital=0;
dita2=0;
} //end if   homming
while(PAP==0 && AUTO==0 && home!=1 && stop!=1) {};
 // a la espera señal de modo elegido
```

Figura 6-9: Llamada al subprograma Homming desde el programa principal

- Stop: re reinician todos los modos.

```
//========= STOP ====================
if(stop==1){
AUTO=0;
PAP=0;
BB_P=0;
BB_N=0;
_Wait (500);
stop=0;
} //end if stop   */
```

Figura 6-10: Parada luego de finalizar el movimiento

- Definición de modo: se cuenta con las variables “pos_sig” y “pos_ant” que sirven como memorias para definir el próximo posible “n” según el modo. Si el modo es automático, se ejecuta el programa cíclicamente 4 veces sin necesidad de ninguna otra variable. Si el modo es paso a paso, se requiere definir en cada ciclo la próxima posición deseada, la siguiente o la anterior.

```
//========= DEFINICION DE MODO ====================
// AUTOMATICO
if (AUTO==1 && n<=3 ){
pos_sig=n+1;
n=pos_sig;
}
//PASO A PASO
if (PAP==1 && n<=4){
while(BB_N==0 && BB_P==0 && home!=1) {};
pos_sig=n+1;
pos_ant=n-1;
// PASO A POSICION SIGUIENTE
  if((PAP==1)&&(BB_N==1)&& n<=3 ){
    BB_P=0;
    n=pos_sig;
    } //end if bb_n
// PASO A POSICION ANTERIOR
  if((PAP==1)&& BB_P==1 && n>=2) {
     BB_N=0;
    n=pos_ant;
    } //end if bb_p
} //end if pap
```

Figura 6-11: Definición de modo de funcionamiento (automático o paso a paso=¿)

- Definición de ángulo de giro según “n”: por medio de la interfaz humano-maquina se

entregan las 4 posiciones deseadas de cada servomotor. Y según el “n” se define a cuál de estas posiciones va cada servomotor.

```
//========== POSICIONES SEGUN "n" ============
switch (n)
{
 case 1:
dital=dital_1;
dita2=dita2_1;
 break;

 case 2:
dital=dital_2;
dita2=dita2_2;
 break;

 case 3:
dital=dital_3;
dita2=dita2_3;
 break;

 case 4:
dital=dital_4;
dita2=dita2_4;
 break;

 }  // end switch
```

Figura 6-12: Definición de ángulo dita1 y dita 2 según “n”:

- Ejecución del subprograma AngGiro (): una vez definido el ángulo de giro, se ejecuta la subtarea AngGiro() y termina el ciclo. Esto sucede continuamente hasta que alguna acción interrumpa el programa.

```
//==================== MOVIMIENTO SEGUN MODO ==========
if(AUTO==1){
AngGirol();
} // end GIRO AUTO
if((PAP==1) && (BB_N==1))  {
AngGirol();
BB_N=0;
} // end GIRO A PROXIMA POSICION
if((PAP==1) && (BB_P==1)){
BB_P=0;
AngGirol();
}  // end GIRO A POSICION ANTERIOR
} //end while
} //end main
```

Figura 6-13: LLamada a subtarea AngGiro () según el modo de funcionamiento

Programa IPOSPlus Controlador ESCLAVO

Este programa es mucho más sencillo porque lo único que hace es recibir valores de dita2 y st2 y a partir de ellos se ejecutan acciones.

Definición de variables

Solo son necesarias las variables st2 y dita2 del ciclo. La variable “dita2” se actualiza ciclo a ciclo recibiendo el dato del driver maestro y “st2” se actualiza según el estado, puede recibir el valor requerido desde el driver maestro o se puede definir desde el programa según la situación.

```
//============================CONTROLADOR DERECHO=========================

#include <constb.h>
#include <iob.h>

//ANGULOS DITA  Y ESTADOS
#define dita2 H131
#define st2 H133 // estado servo 2 ( moving=0; ready=1; finish=2)
```

Figura 6-14: Definición de variables H

Subtareas

Las subtareas del driver esclavo solo le indican al servomotor el movimiento a realizar, todo el resto lo define el driver maestro.

```
 /*=========================================*/
AngGiro2() {
// MOVIMIENTO SERVO 2 A DITA 2
_GoAbs( GO_WAIT,dita2 );
st2=2;
//ESPERA A RECIBIR DE DRIVER 1 QUE ESTA LISTO PARA PROXIMA ACCION (ST2=1)
while( st2!=1 ){}
//_Wait(1000);
//Hl36=4;
 } //end AngGiro2

 /*=========================================*/
Homming() {
//MOVIMIENTO SERVO 2 A HOME
st2=0;
_Go0( GO0_U_W_ZP );
st2=2;
//ESPERA A RECIBIR DE DRIVER 1 QUE ESTA LISTO PARA PROXIMA ACCION (ST2=1)
while( st2!=1 ){}
} // end homming
```

Figura 6-15: Subtarea AngGiro2 () y Homming ()

Programa principal

Se inicializa el driver en estado de espera para evitar que comience alguna subtarea sin la indicación del driver maestro y se habilita el servomotor con el sentido de giro en sentido de las agujas del reloj. Luego se ingresa en el bucle cíclico que se encuentra a la espera constante del estado que le define el driver maestro por medio de la variables “st2”.

```
main() {

//INICIALIZACION
st2=2; //ESTADO DE ESPERA A RECIBIR INDICACION DEL DRIVER MAESTRO

//HABILITACION DE SERVOMOTOR Y DEFINICION SENTIDO DE GIRO
_BitClear( ControlWord, 0 );
_BitClear( ControlWord, 1 );
_BitClear( ControlWord, 3 );
_BitSet(ControlWord,2) ;

// PROGRAMA CICLICO

while (1) {
if(st2==3) Homming();
if ((st2==0)) AngGiro2();
} //end while
} //end main
```

Figura 6-16: Programa principal para servomotor esclavo

Definición del Cero maquina (Pulso Cero)

En el proyecto se define el cero maquina con fines de carrera físicos, uno en cada servomotor. Estos se conectan por su salida NORMAL CERRADA a la bornera X16 (DI06) de cada MOVIDRIVE

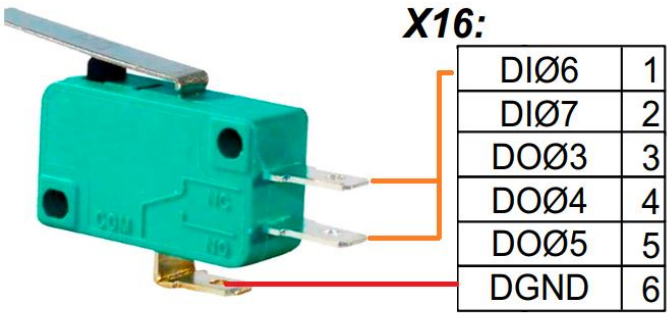


Figura 6-17: Conexión fin de carrera en bornera X16

Para su correcto funcionamiento es necesario que los parámetros de MOVITOOLS se definan en relación a la posición en la que se instalan los fines de carrera por lo que los parámetros *Binary input DI06* (P605) y *Reference travel type* (P903) serán distintos para cada controlador.

- Controlador izquierdo

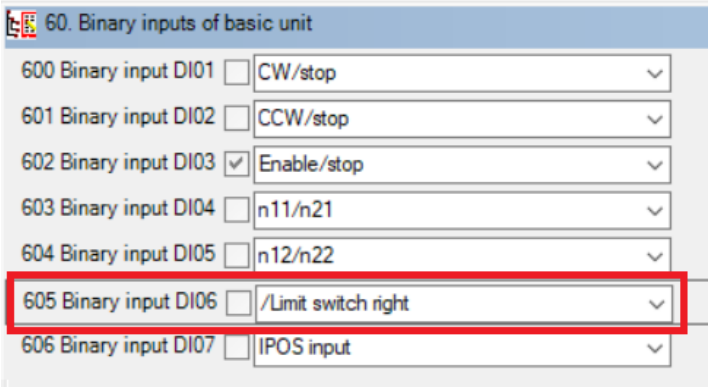


Figura 6-18: Parámetro P605 controlador izquierdo

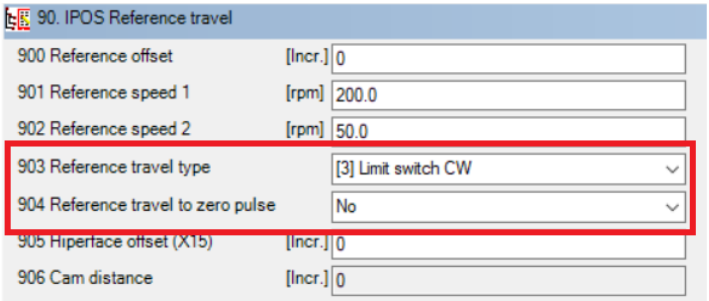


Figura 6-19: Parámetro P903 y P904 controlador izquierdo

- Controlador derecho

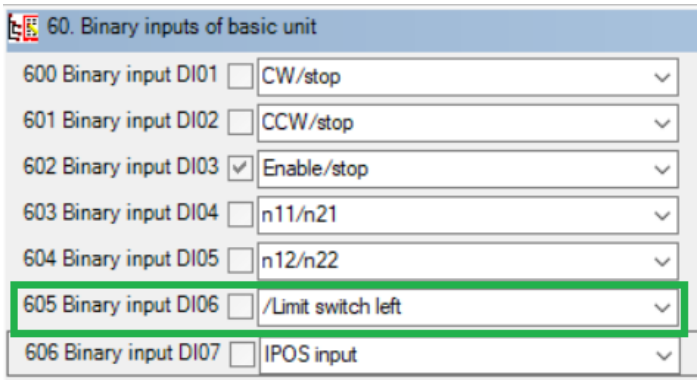


Figura 6-20: Parámetro P605 controlador derecho

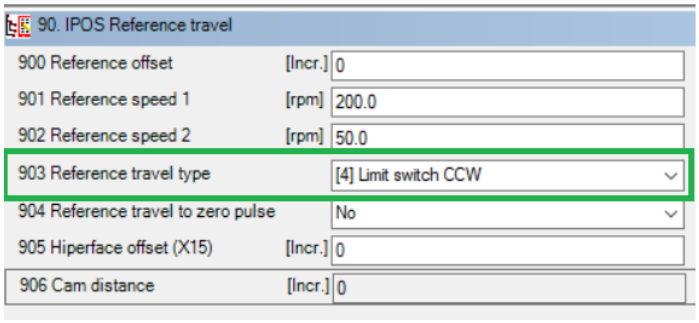


Figura 6-21: Parámetro P903 controlador derecho

Interfaz hombre-maquina

La interfaz se construye con el programa Application Builder, versión 6.20, perteneciente a la suite de programas de Movitools permitiéndole al usuario el control del sistema por un panel virtual. Para ingresar a esta herramienta desde MOVITOOLS se siguen los mismos pasos utilizados para entrar a IPOSplus (desde el menú de contexto de la unidad indicada).

En el caso del proyecto, es el controlador maestro quien está conectado al tablero virtual, por lo que se debe entrar a AppBuilder desde dicha unidad.

En el programa desarrollado para lograr la interface hombre-máquina se pueden introducir las posiciones deseadas de ambos servomotores, definir el modo de trabajo (automático o manual), obligar a los servomotores dirigirse a su cero y parar el sistema.
También se puede observar las condiciones en tiempo real del sistema.

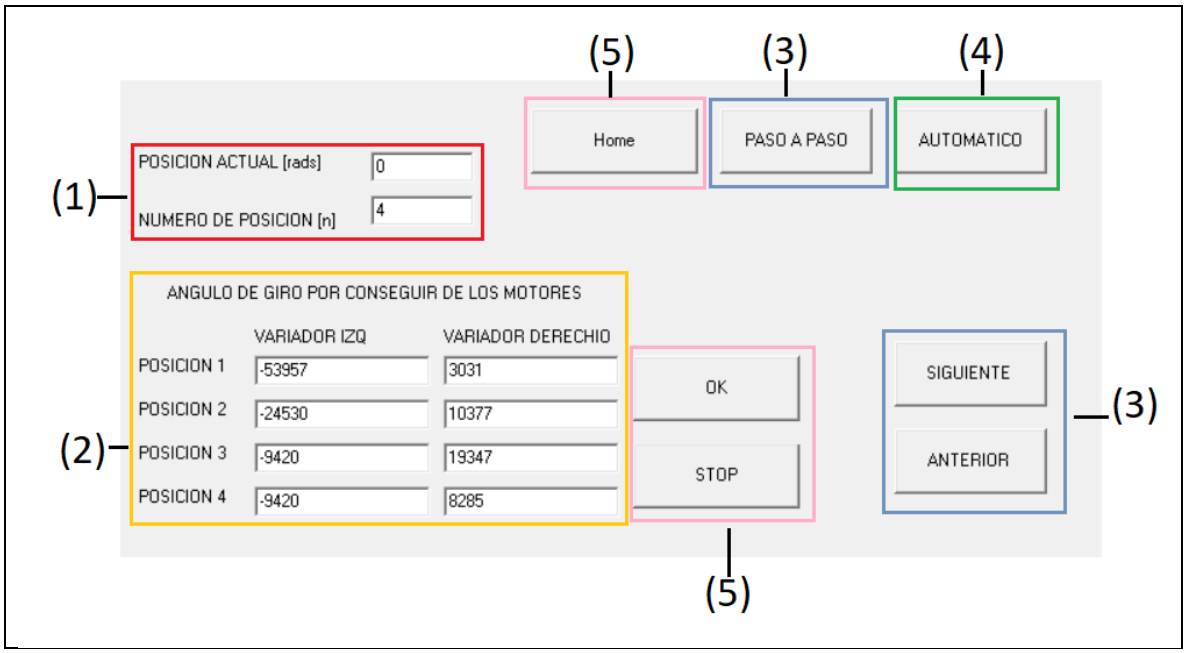


Figura 6-22: table virtual del proyecto desarrollado en AapBuilder

(1) Cajas de texto de lectura que muestran el estado del sistema en tiempo real
(2) Ángulos de giro introducidos por el usuario para lograr las posiciones del efector deseadas
(3) Botones para definir y dirigir el modo paso a paso de funcionamiento
(4) Botón para definir como automático al modo de funcionamiento
(5) Botones para usar independientemente del modo de funcionamiento. "OK" se usa para entregarle al programa IPOSplus los ángulos de giro definidos por el usuario.

Definición de botones y cajas de texto

Cajas de Texto

En el proyecto se usan las cajas de texto para leer información proveniente del programa (posición actual, numero de posición actual) o para escribir entradas en el programa (posiciones deseadas).

Cajas de texto de lectura

Para comunicar los boxes con la variable de IPOSPlus correcta se usa el índice de la variable (*Index* en pestaña *Comunication*). Los boxes son de solo lectura por lo que, como se ve en la Figura 6-23, en la pestaña *Properties* se define como *true* a la opción *ReadOnly* y en la pestaña *Comunication* se define como *true* las opciones de *Download* y *Upload* en la sección *Monitor*, lo que permite una visualización en tiempo de real del cambio en los valores.

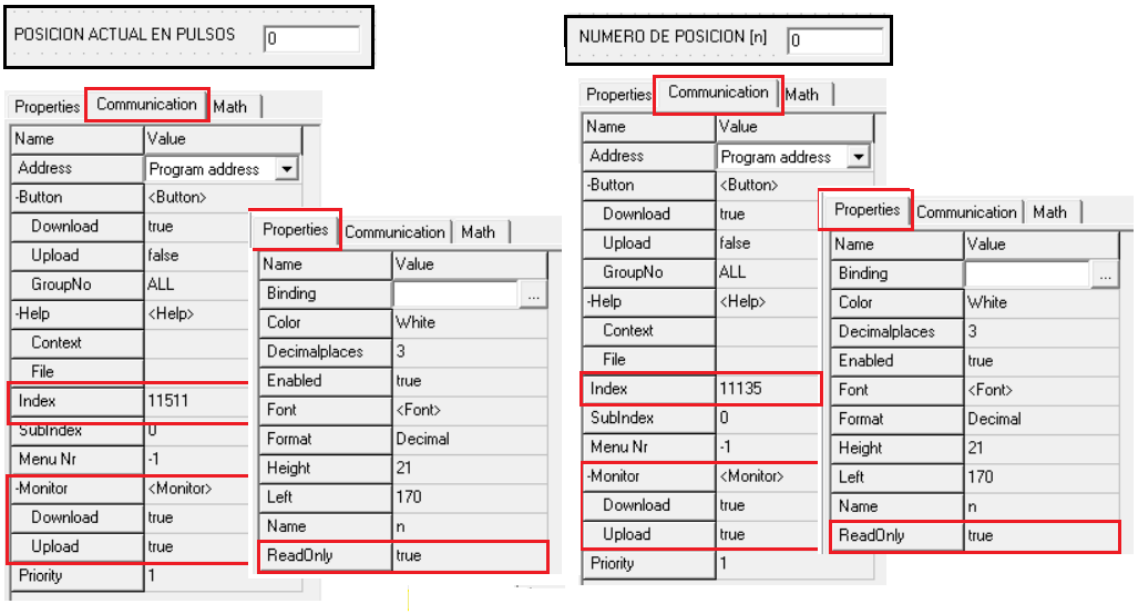


Figura 6-23: Pestañas Properties y Comunication para las cajas de lectura del tablero

Caja de texto de escritura

Las posiciones deseadas las ingresa el usuario una vez calculadas en MATLAB y al pulsar el botón OK estas se cargan en las variables H de IPOSPlus correspondientes según el *Index* definido en la pestaña *Comunication*. Como la entrega de estos valores al programa se lleva a cabo por el botón OK, se define como *false* en todas las opciones de *Button* y *Monitor*.

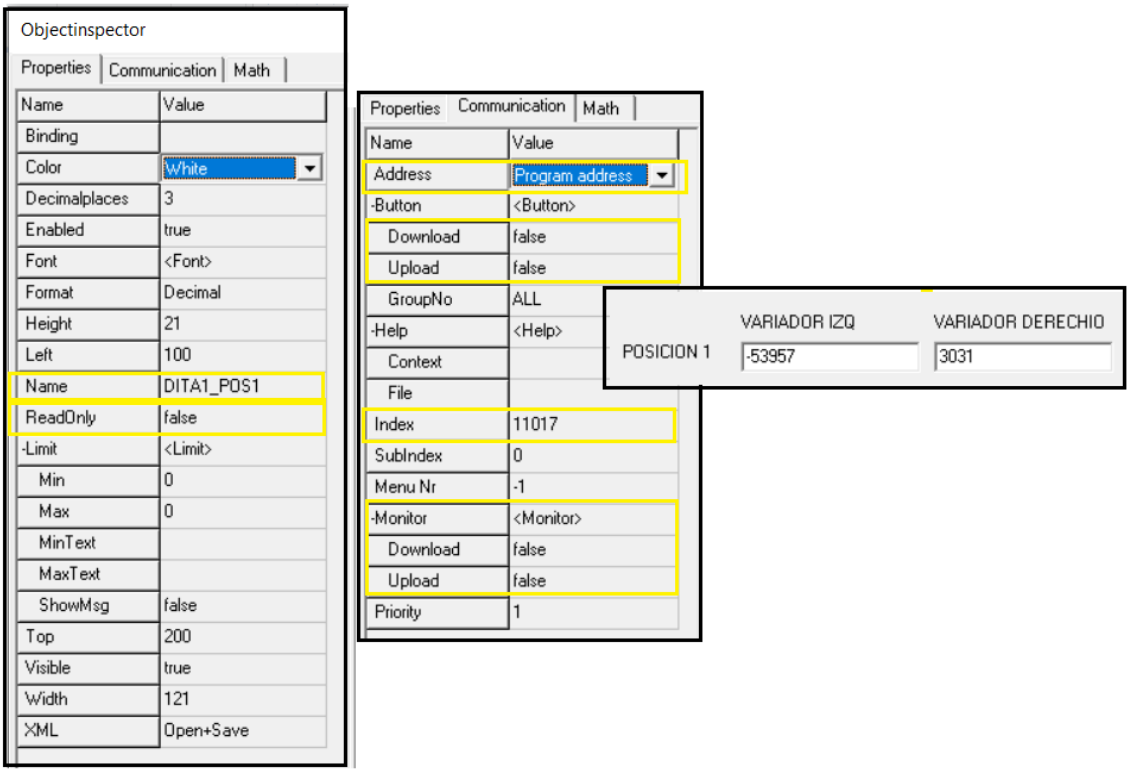


Figura 6-24: Pestañas Properties y Comunication para las cajas de escritura del tablero

Botones

Los botones del proyecto sirven como interpretadores de fórmulas(ver Figura 6-25). Las funciones escriben en el programa IPOSPlus los valores definidos de dita (“BOTON OK”), el modo de funcionamiento (BOTON AUTOMATICO, PASO A PASO, POSICION ANTERIO, POSICION SIGUIENTE) o una tarea específica (BOTON STOP o HOME). Todos los botones logran su cometido entregándole un valor a una variable H de IPOSPlus y las fórmulas usadas tienen el siguiente formato:

WriteIndex(address del controlador, índex ,valor);

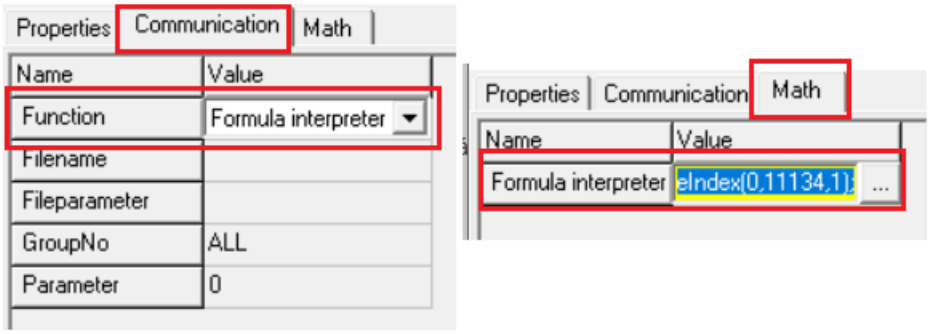


Figura 6-25: elección de función de botones y definición de formula

Tabla 6-1: Descripción de las fórmulas de los botones

BOTON	Formula escrita en pestaña Math	Descripción
Botón Automático	WriteIndex(0,11137,0); WriteIndex(0,11136,1);	Entrega un “0” a la variable H137 de IPOSplus y un “1” a la variable H136. Inhibe el modo paso a paso y lo pisa con el modo automático.
Botón Paso a Paso	WriteIndex(0,11136,0); WriteIndex(0,11137,1);	Similar a la anterior
Botón Siguiente	WriteIndex(0,11138,1);	Entrega un “1” a la variable H138, permite sumar 1 a la posición actual del servo
Botón Anterior	WriteIndex(0,11139,1);	Entrega un “1” a la variable H139, permite restar 1 a la posición actual del servo
Botón Home	WriteIndex(0,11129,1);	Entrega un “1” a la variable H129 que lleva al servomotor a

		su posición cero
Botón Stop	WriteIndex(0,11134,1); WriteIndex(0,11136,0); WriteIndex(0,11137,0); WriteIndex(0,11138,0); WriteIndex(0,11139,0);	Para al robot una vez concluido el movimiento que estaba haciendo y quita el modo de funcionamiento definido
Botón Okey	WriteIndex(0,11017,Comp.DITA1_POS1); WriteIndex(0,11018,Comp.DITA2_POS1); WriteIndex(0,11019,Comp.DITA1_POS2); WriteIndex(0,11020,Comp.DITA2_POS2); WriteIndex(0,11021,Comp.DITA1_POS3); WriteIndex(0,11022,Comp.DITA2_POS3); WriteIndex(0,11023,Comp.DITA1_POS4); WriteIndex(0,11024,Comp.DITA2_POS4); WriteIndex(0,11135,0);	Le entrega a cada variable H definida como "*DITA"* el valor definido por el usuario. Resetea el contador de posición *"n"*

Puesta a prueba del sistema

Para poner a prueba el sistema se elegirán dos sucesiones de cuatro posiciones para el efector final usando un croquis 2D (ver **¡Error! No se encuentra el origen de la referencia.**) hecho en SolidWorks con las longitudes reales del brazo medidas en el prototipo. Esto nos permite ver la configuración que obtiene el robot completo al llegar a la posición deseada.

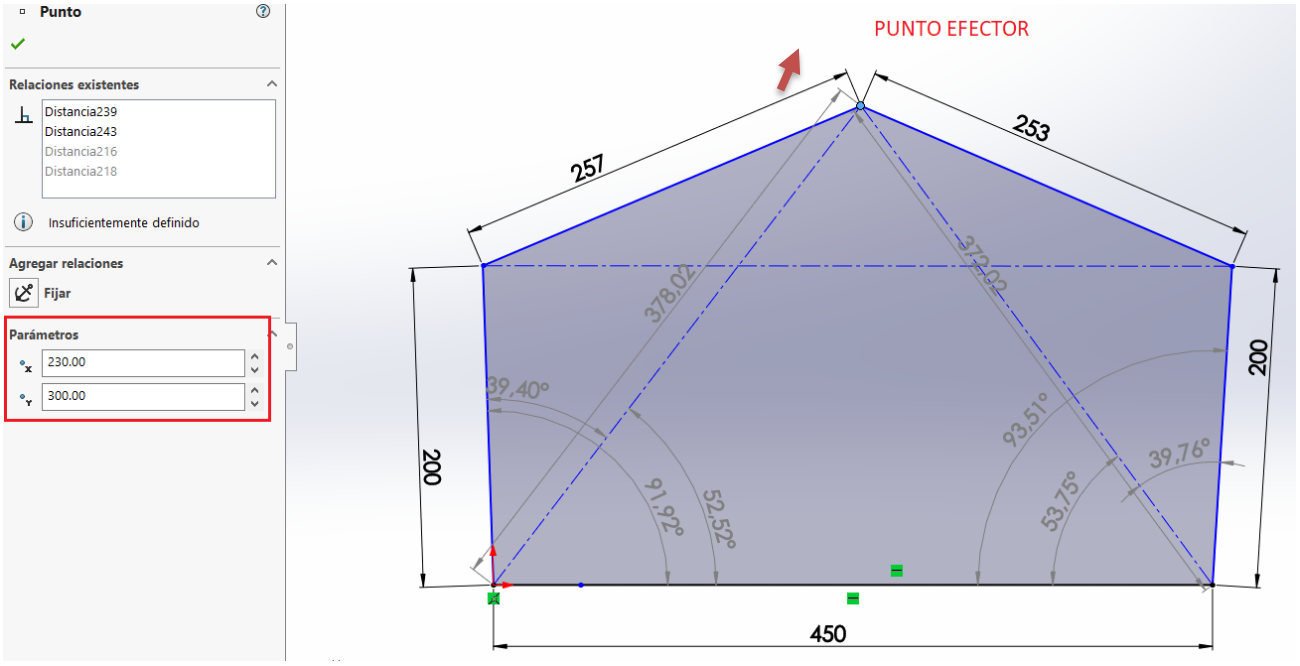


Figura 6-26: Croquis 2D SolidWorks

Una vez elegidos estos puntos se introducen los valores cartesianos *xp* e *yp* en MATLAB junto con el modo de trabajo elegido. El programa de MATLAB entrega como resultado los ángulos de giro en pulsos (ver Figura 6-27) que se introducen en el tablero creado en APPBUILDER.

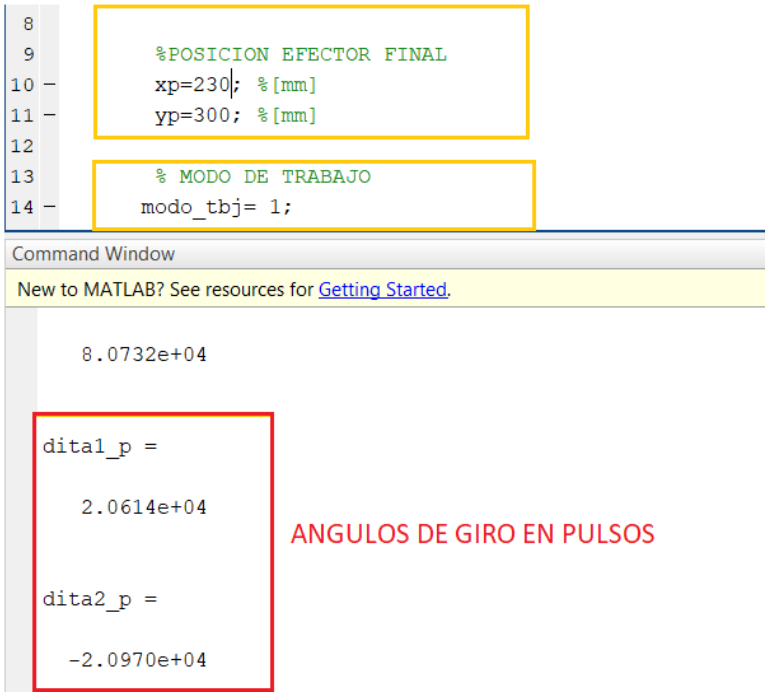


Figura 6-27: Angulos de giro en pulsos en MATLAB

Por último se usa el tablero virtual para entregarle la orden al controlador para que vaya a las posiciones deseadas en un modo paso a paso. En cada posición se hace una medición aproximada de la posición del punto final (observar la precisión) y se repiten los 4 pasos de forma interativa y en distinto orden para verificar la repetitividad del sistema.

Prueba: mismo punto en los 4 modos de trabajo

1. Se elige el punto P(250, 370). Se observa en la tabla los ángulos de giro absolutos en pulsos de cada motor para llegar a la posición del efecto en cada modo de trabajo y en imagen la configuración del robot para los 4 modos de trabajo.

2.

3. *Tabla 6-2: Ángulos de giro en pulsos para los 4 modos de trabajo entregados por MATLAB*

MODO	DITA 1	DITA 2
1 (++)	(-15676)	19347
2 (+-)	(-15676)	8285
3(-+)	(-9420)	19347
4 (--)	(-9420)	8285

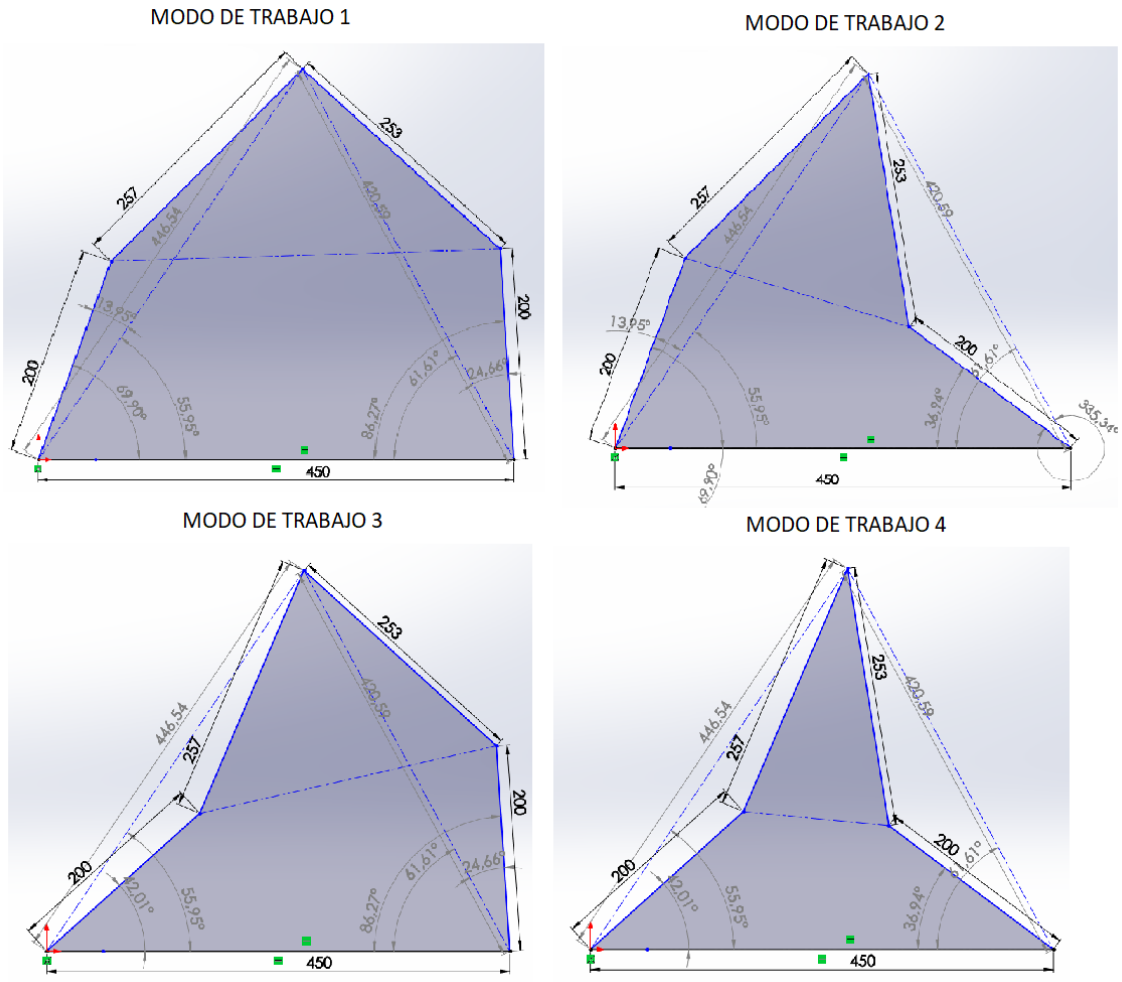


Figura 6-28: Configuración del mecanismo para cada modo de trabajo

4. Como se ve en la figura, se cargar los valores en el tablero virtual y se le indica al controlador trabajar paso a paso

ANGULO DE GIRO POR CONSEGUIR DE LOS MOTORES

	VARIADOR IZQ	VARIADOR DERECHIO
POSICION 1	-15676	19347
POSICION 2	-15676	8285
POSICION 3	-9420	19347
POSICION 4	-9420	8285

Figura 6-29: Tablero virtual con los valores definidos para la prueba

5. Los resultados obtenidos en el prototipo físico tienen un grado relativamente alto de imprecisión debido a los juegos mecánicos y falta de precisión en el largo de los eslabones. Se observa que hay semejanza en el resultado de los modos 1 y 3, y en los resultados de los modo 2 y 4.

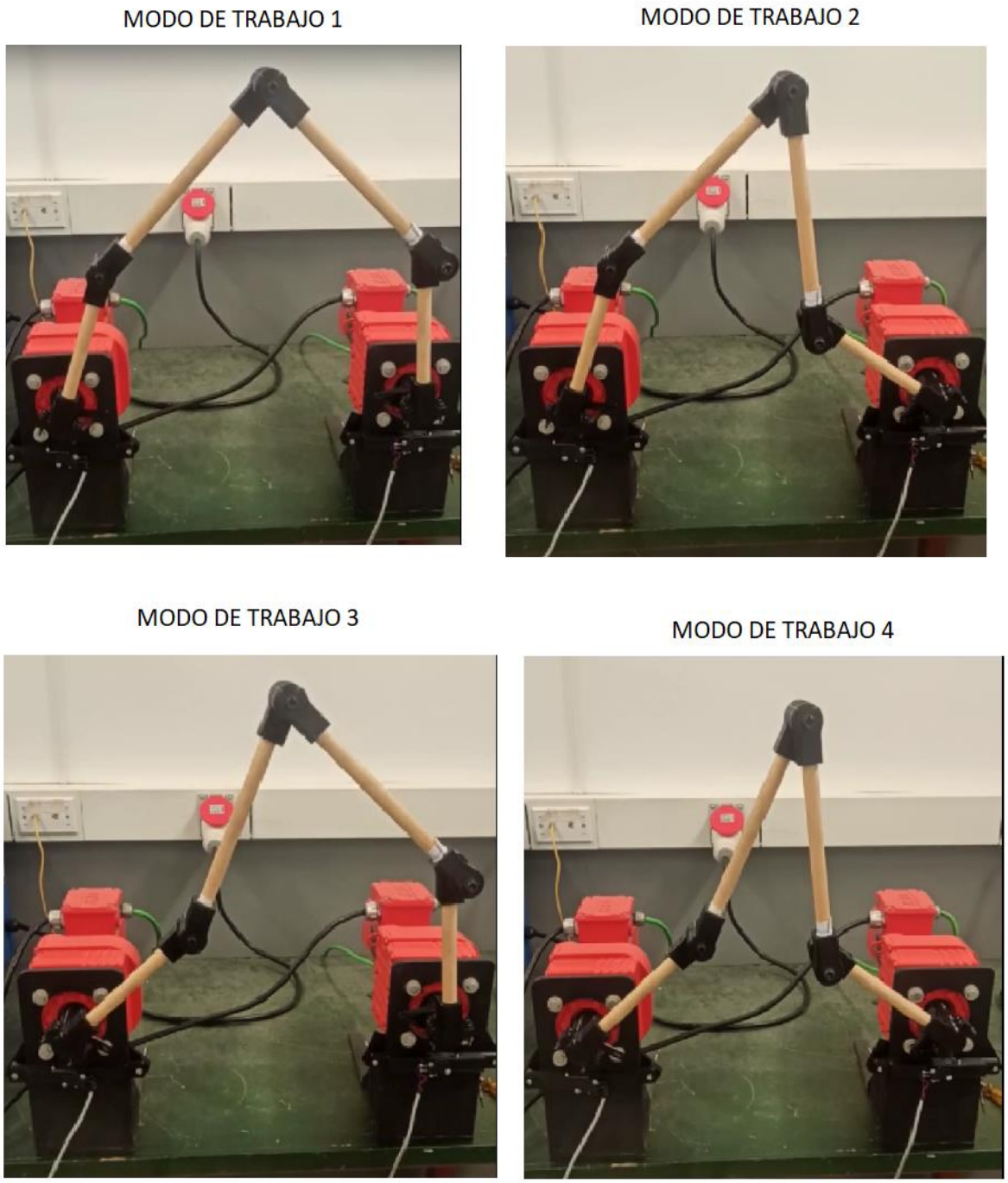


Figura 6-30: Vista frontal del prototipo en cada modo de trabajo

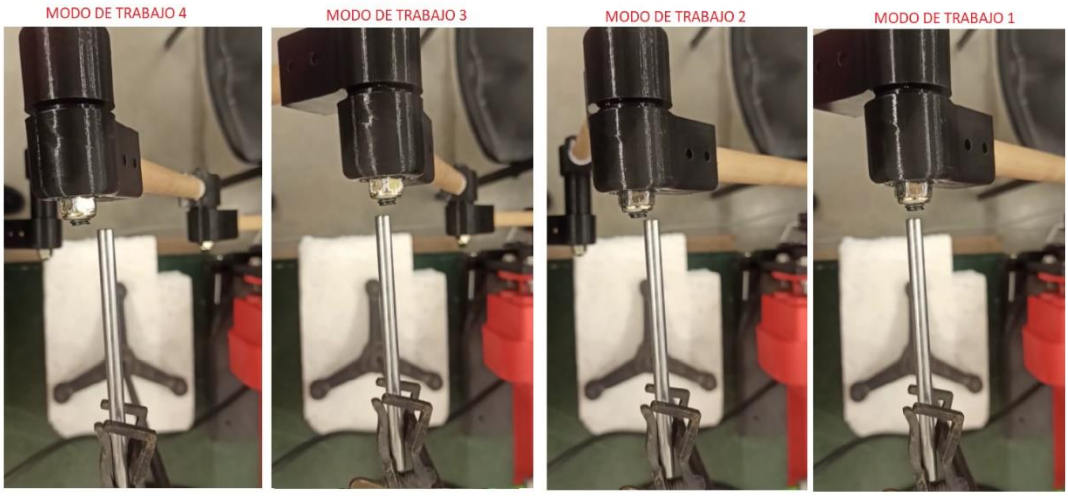


Figura 6-31: vista superior del prototipo con uso de referencia para el control de su precisión

7. Conclusiones

El desarrollo del proyecto de final de Carrera de Grado de Ingeniería Mecánica fue un proceso que exigió un cambio sobre la idea original una vez que se accedió al servomotor y a las herramientas disponibles.

En un primer momento el objetivo del trabajo fue desarrollar el control del robot con PLC Siemens disponible en el Laboratorio de Sistemas Digitales Industriales perteneciente a la Facultad de Ingeniería Electrónica y realizar el proyecto de forma colaborativa con Ingeniería Electrónica. Era poner en acto una modalidad de trabajo que es indispensable para el abordaje de los nuevos problemas con que se encuentra la Ingeniería Mecánica en particular, pero que también atañe a todas las especialidades de la Ingeniería.

Esto no fue posible porque al poner los robots en condiciones de trabajo se vio que los controladores no contaban con la placa de comunicación necesaria para la conexión con el PLC y la adquisición de la misma no estaba al alcance del proyecto.

Ante esta situación se indagó sobre posibles alternativas y se solicitó asesoramiento a los representantes post venta de la empresa SEW EURODRIVE quienes advirtieron del potencial del controlador MOVIDRIVE-B y sus herramientas (software MOVITOOLS y IPOSPlus) disponible en la Escuela de Ingeniería Mecánica. Se toma esta sugerencia y se redefine el proyecto dejando de lado el PLC y por lo tanto la actividad articulada con la Carrera de Ingeniería Electrónica. El proyecto se desarrolla, entonces, exclusivamente en el Laboratorio FABLAB.

Si bien se tuvo que sobrellevar estos inconvenientes no fue necesario cambiar los objetivos esenciales del proyecto, pero fue necesario adaptarlo a las nuevas condiciones de desarrollo que llevo a cierta limitación de los alcances del mismo. Se tuvo que usar un software de cálculo numérico MATLAB y se resignó el control de la curva de trayectoria del efector del robot para llegar de una posición a otra.

El primer objetivo, poner en marcha los dos servomotores y los variadores de frecuencia, se alcanzó ampliamente poniendo en valor una importante inversión realizada por la Escuela de Ingeniería Mecánica y quedando así los servomotores en el Laboratorio FABLAB a disposición de los estudiantes para futuros desarrollos. Asimismo se deja a disposición un manual de uso tercer objetivo del proyecto) para facilitar el acercamiento de los estudiantes a estas herramientas (ver cap. 2,3 y 4).

El desarrollo y fabricación del prototipo robot 5 barras con dos grados de libertad está disponible para los interesados en el Laboratorio. Este prototipo se concretó, se probó con diferentes secuencias de posiciones poniendo a prueba las singularidades y modos de trabajo.

En el prototipo se usan bujes espaciadores entre las barras que requieren de un juego mecánico para lograr el deslizamiento entre ellas que permite la rotación; es este juego mecánico el responsable de la baja precisión de la posición final del efector. Este déficit es subsanable con el uso de rodamientos que evitan el juego mecánico, para el proyecto no fue posible su utilización porque no están disponibles en el laboratorio.
Otra característica que se observa al llevar al motor a cuatro posiciones distinta de forma continua es que la precisión del punto efector va disminuyendo debido al juego mecánico y al deslizamiento entre las piezas. Esto se puede salvar usando otro método de fabricación de las piezas como por ejemplo el mecanizado en reemplazo a la impresión 3D de baja calidad como la usada en el proyecto.

También se observa que la repetitividad de la posición final del efector es baja en el uso del robot en modo automático la propagación del error de precisión en el paso de las posiciones.
A pesar de las limitaciones descriptas el prototipo logra cumplir con una tolerancia de 2 centímetros manteniendo su modo de trabajo y superando las singularidades por las que atraviesa.

Bibliografía

[1] Aldazaba, I. G. (2022). *Start Up MOVITOOLS Motion Studio.* SEW EURODRIVE.

[2] Alexandre Figielski, I. A. (2014). *Towards Development of a 2-DOF Planar Parallel Robot* . Quebec.

[3] Gamble, B. J. (s.f.). *5-Bar Linkage Kinematic Solver and Simulator* . University of Vermont: 2020.

[4] Ilian Bonev, F. B. (2010). *Development of a Five-Bar Parallel Robot With Large Workspace.* Quebec.

[5] Monroy, Á. A., & Aedo, R. F. (2019). *"INVESTIGACIÓN PARA COMUNICACIÓN DE SEW MOVIDRIVE CON PLC".* Hualpén, Chile: UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA SEDE CONCEPCIÓN – REY BALDUINO DE BELGICA.

[6] MUÑOZ, C. G. (2017). *DISEÑO Y SIMULACIÓN DE UN CONTROLADOR DE POSICIÓN* . Quito.

[7] PARDO, N. G. (2018). *CONTROL AUTOMÁTICO Y MANDO DE UN SISTEMA EN FIBRA DE CARBONO POR HILADO.* SANTIAGO DE CALI: Universidad del Valle.

[8] Rodríguez, E. Y., Mckinley, J. R., & J. V. (2017). *Control por par calculado de un robot paralelo planar 2-RR.* Colombia .

[9] SEW EURODRIVE . (2009). *Manual IPOSplus® Positioning and Sequence Control System.*

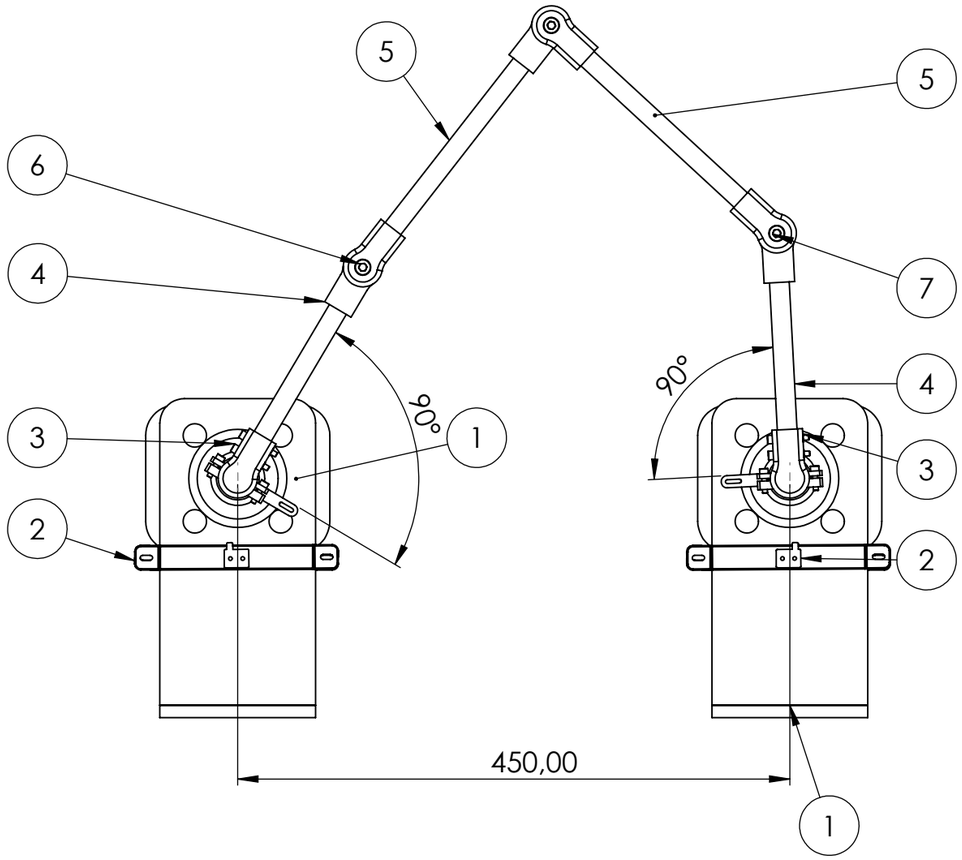
[10] SEW EURODRIVE. (2009). *Comunicación en serie MOVIDRIVE.*

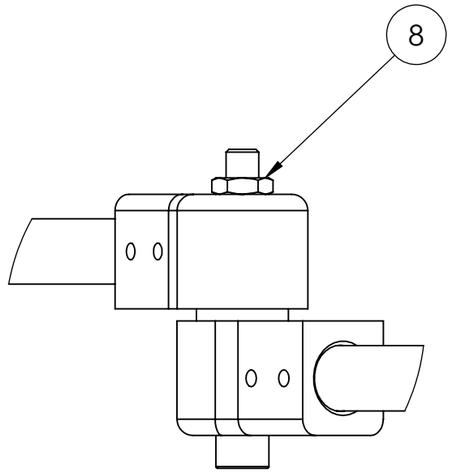
[11] SEW EURODRIVE. (2017). *Manual MOVITOOLS® MotionStudio.*

[12] Thomas Frank, G.-A. (2002). *Application Builder with MOVITOOLS®- Workshop for Self-Study.* SEW EURODRIVE.

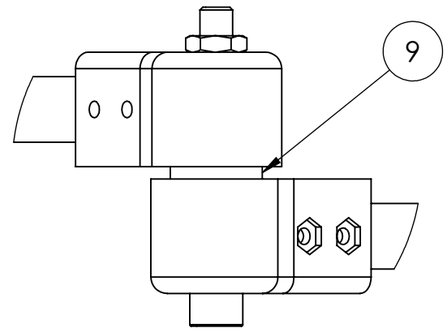
Anexo A: Programa de Matlab

```
clear all; clc;
% Longitud de las Barras:
     w = 450;    % [mm]
    a1 = 200;    % [mm]
    b1 = 257;    % [mm]
    b2 = 253;    % [mm]
    a2 = 200;    % [mm]

    %POSICION EFECTOR FINAL
    xp=3; %[mm]
    yp=62; %[mm]

    % MODO DE TRABAJO
   modo_tbj= 1;
   %1 = beta1 y beta2 positivos (++)
   %2 =beta1 positivo y beta2 negativo (+-)
   %3 == beta1 negativo y beta1 posutivo (-+)
   %4 ==beta1 y beta2 negativos(--)

    %====================CALCULO DE POSICIONES============================
    c1= sqrt(xp^2+ yp^2);
    c2= sqrt(c1^2- 2*xp*w+w^2);
    alfa1= acos(xp/c1);
    alfa2=acos((-xp+w)/c2);
    beta1= acos((b1^2-a1^2-c1^2)/(-2*c1*a1));
    beta2= acos((b2^2-a2^2-c2^2)/(-2*c2*a2));

    if modo_tbj==1;
dita1=alfa1+beta1;
dita2= alfa2+beta2;
     else if modo_tbj==2;
dita1=alfa1+beta1;
dita2= alfa2-beta2;
     else if modo_tbj==3;
dita1=alfa1-beta1;
dita2= alfa2+beta2;
     else if modo_tbj==4;
dita1=alfa1-beta1;
dita2= alfa2-beta2;
      end
    end
    end
    end
  %CONVERSION DE RAD A
   alfa1_g=rad2deg(alfa1);
   alfa2_g=rad2deg(alfa2);
   beta1_g=rad2deg(beta1);
   beta2_g=rad2deg(beta2);
   dita1_g=rad2deg(dita1)
   dita2_g=rad2deg(dita2)
  %GIRO DE SERVOMOTOR 1 (DITA1) y 2(DITA 2)
  %LISTO PARA INGRESAR A APPBUILDER
pulso_giro=1024*19.71*4
dita1_p=dita1/(2*pi)*pulso_giro
dita2_p=dita2/(2*pi)*(-pulso_giro)
```

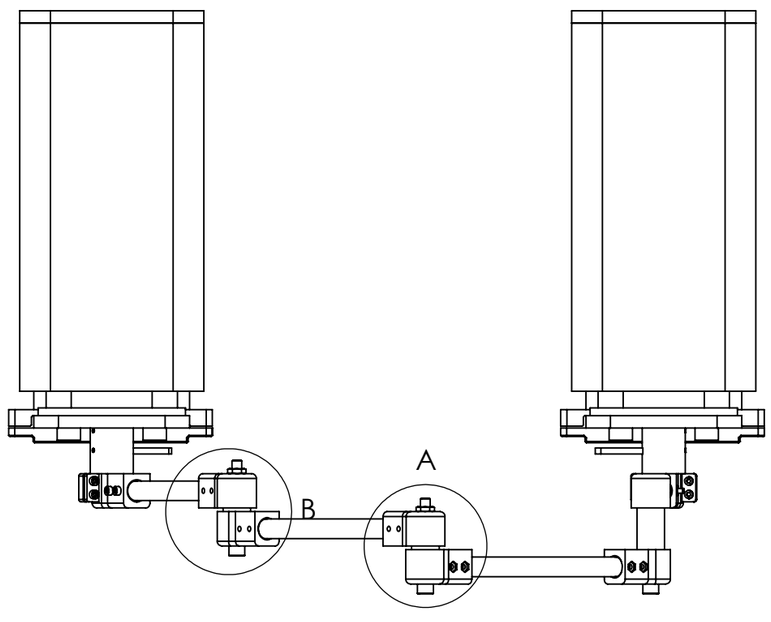




DETALLE B
ESCALA 1 : 1.5



DETALLE A
ESCALA 1 : 1.5



N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	conjunto motor		2
2	conjunto fin de carrera		2
3	acople barra- motor		2
4	Barra comandada		2
5	Barra flotante		2
6	ISO 4762 M8 x 70 - 28N		2
7	ISO 4762 M8 x 80 - 28N		1
8	ISO - 4035 - M8 - N		3
9	Buje separador		1

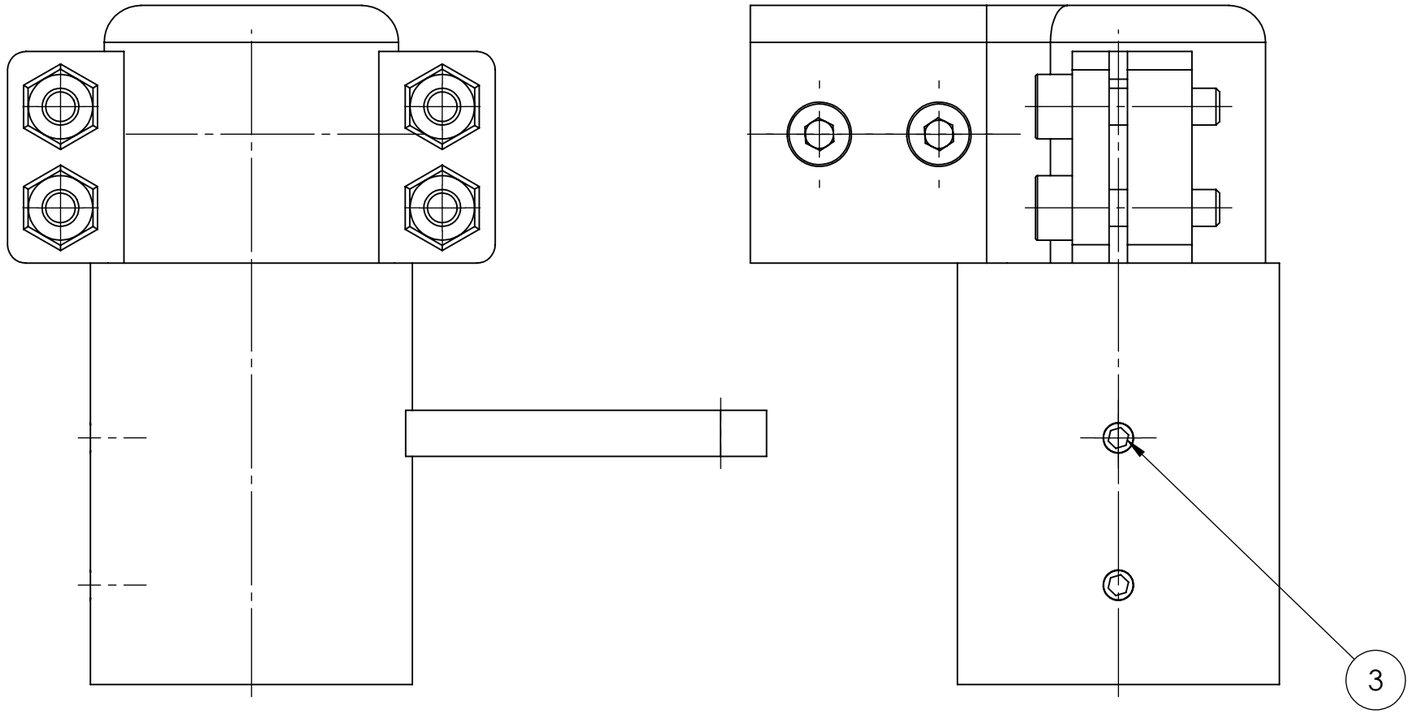
DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

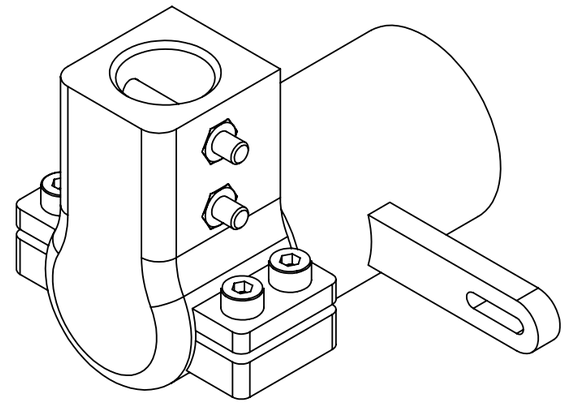
ROBOT 5 BARRAS 2 GDL

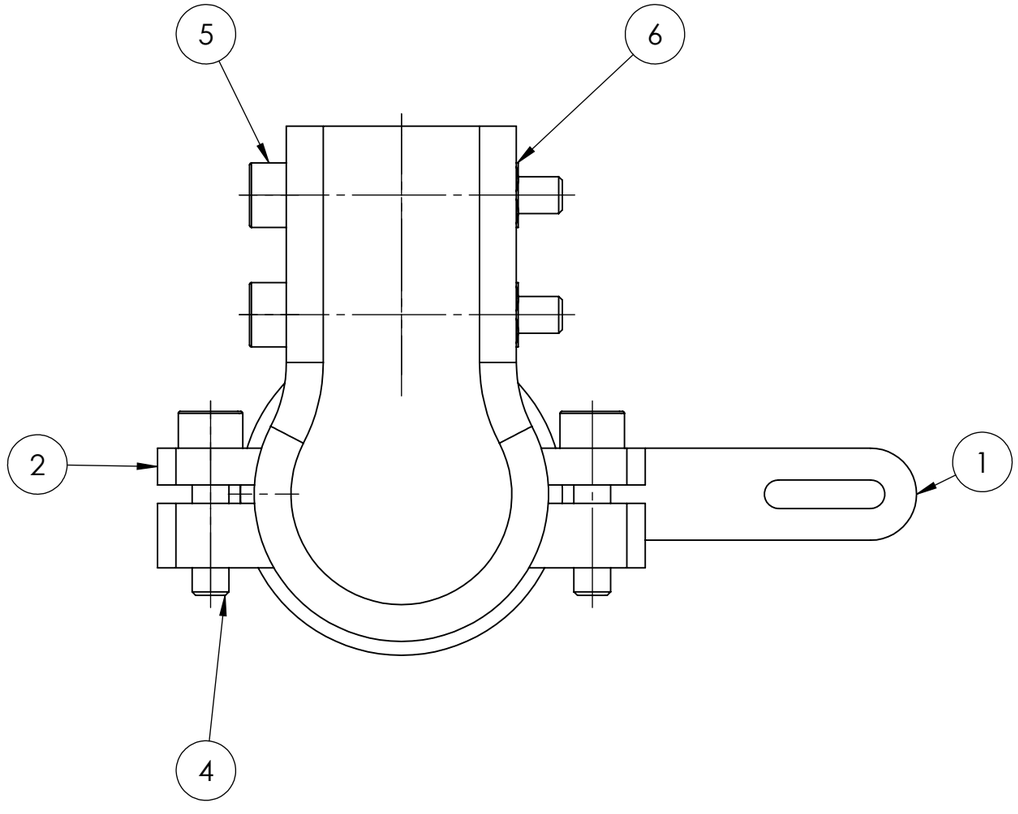
	FECHA	NOMBRE
DIBUJO		PALOMA
APROBO		BARAT

ESCALAS
1:5

N° 0000







N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	Acople 1/CANTIDAD
1	Acople motor		1
2	Acople barra		1
3	ISO 4029 - M4 x 5-N		2
4	ISO 4762 M4 x 16 - 16N		4
5	ISO 4762 M4 x 30 - 20N		2
6	ISO - 4035 - M4 - N		6

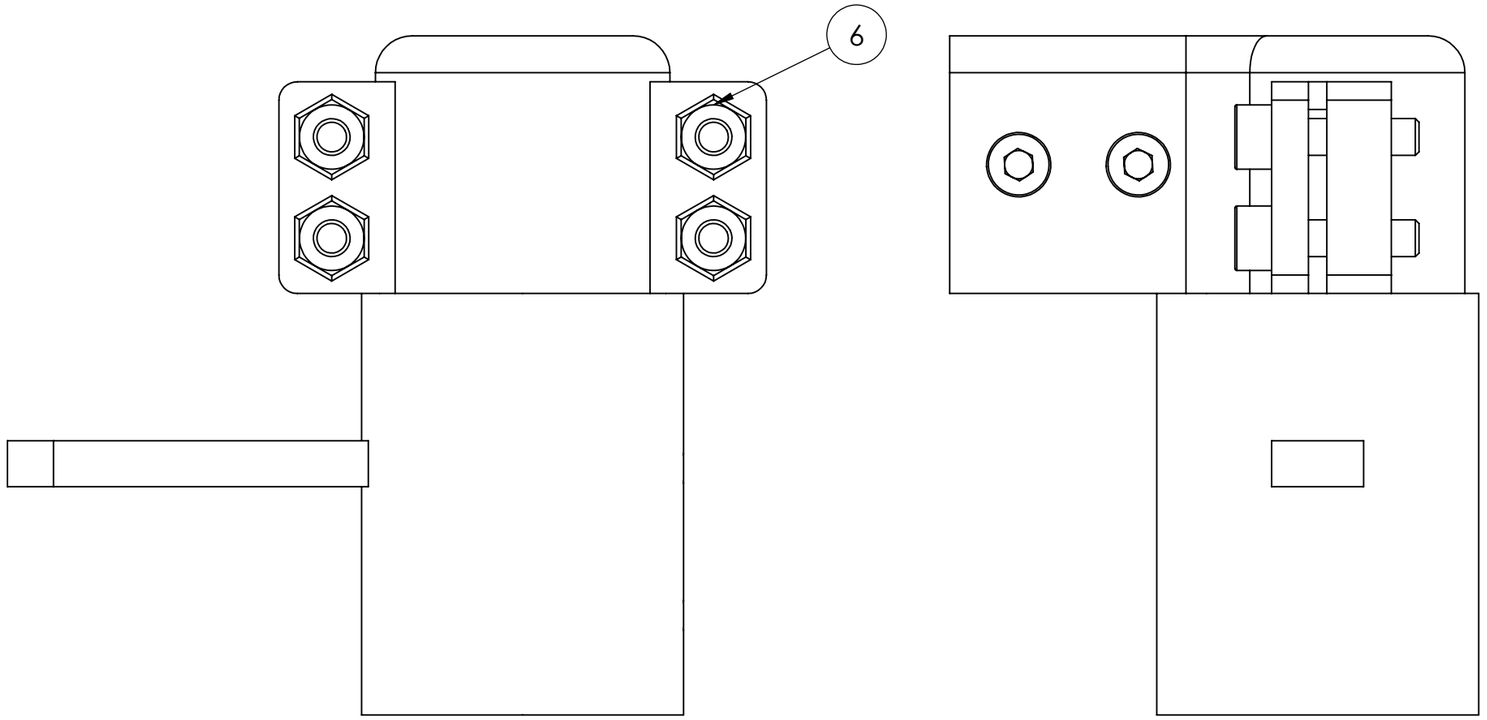
DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

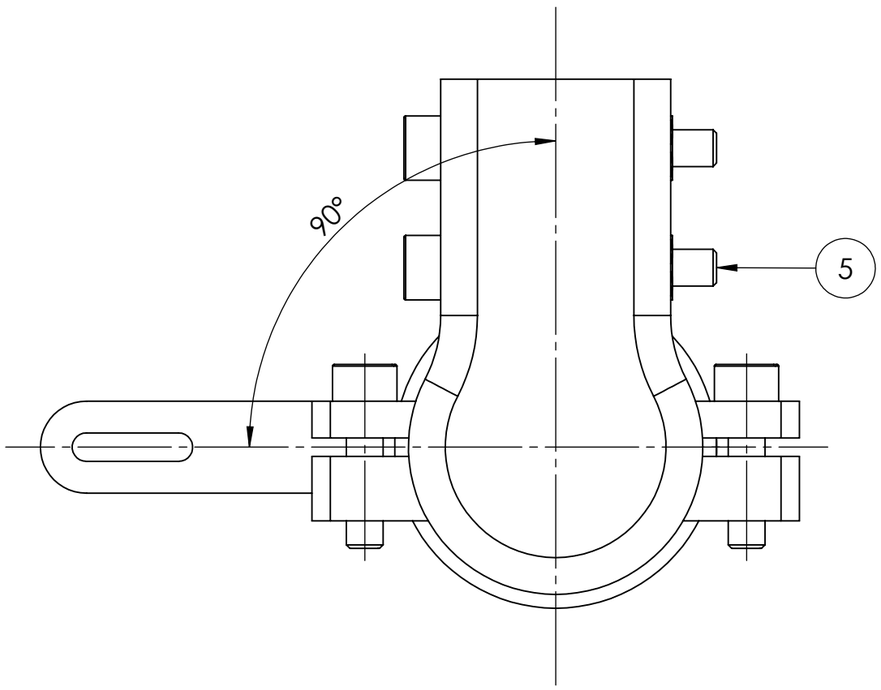
SUBCONJUNTO
ACOPLE BARRA MOTOR IZQ.

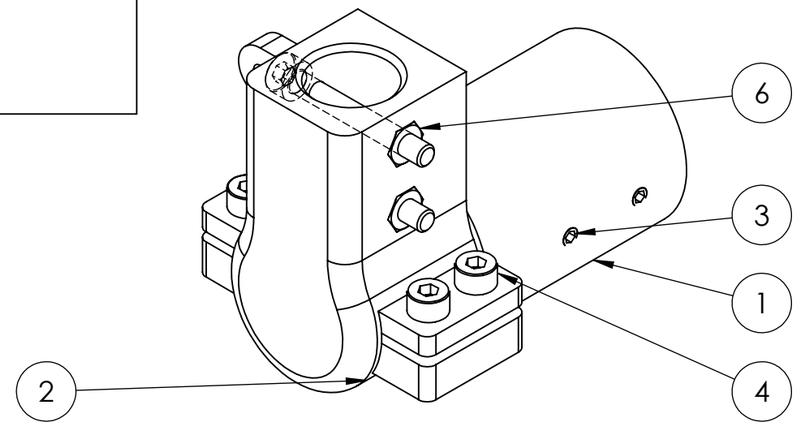
	FECHA	NOMBRE
DIBUJO		PALOMA
APROBO		BARAT

ESCALAS
3:2

N° 0011







N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	Acople motor		1
2	Acople barra		1
3	ISO 4029 - M4 x 5-N		2
4	ISO 4762 M4 x 16 - 16N		4
5	ISO 4762 M4 x 30 - 20N		2
6	ISO - 4035 - M4 - N		6

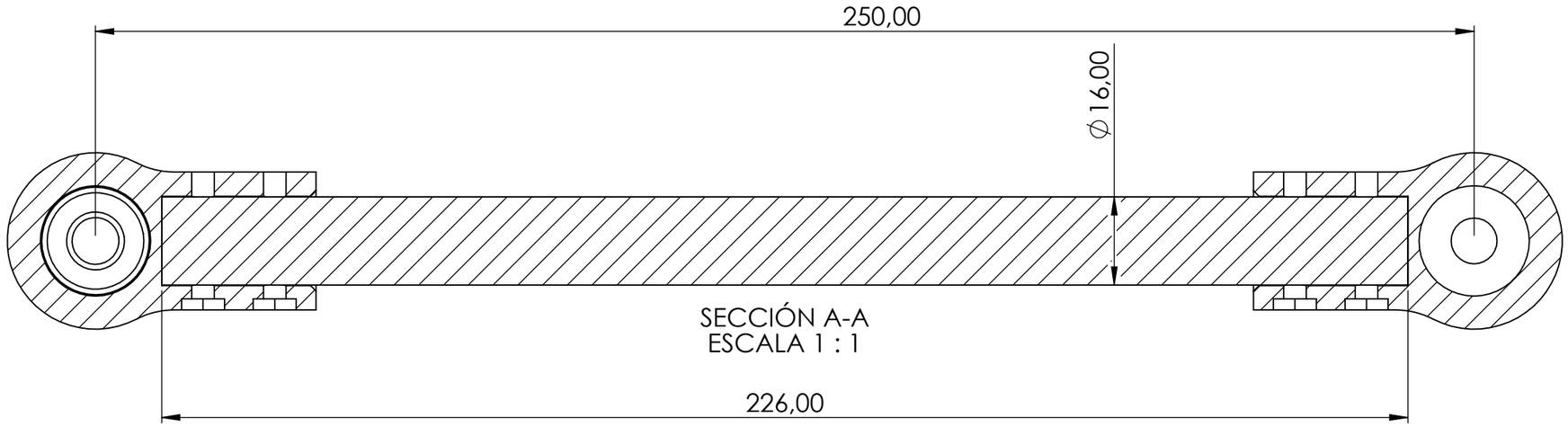
DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

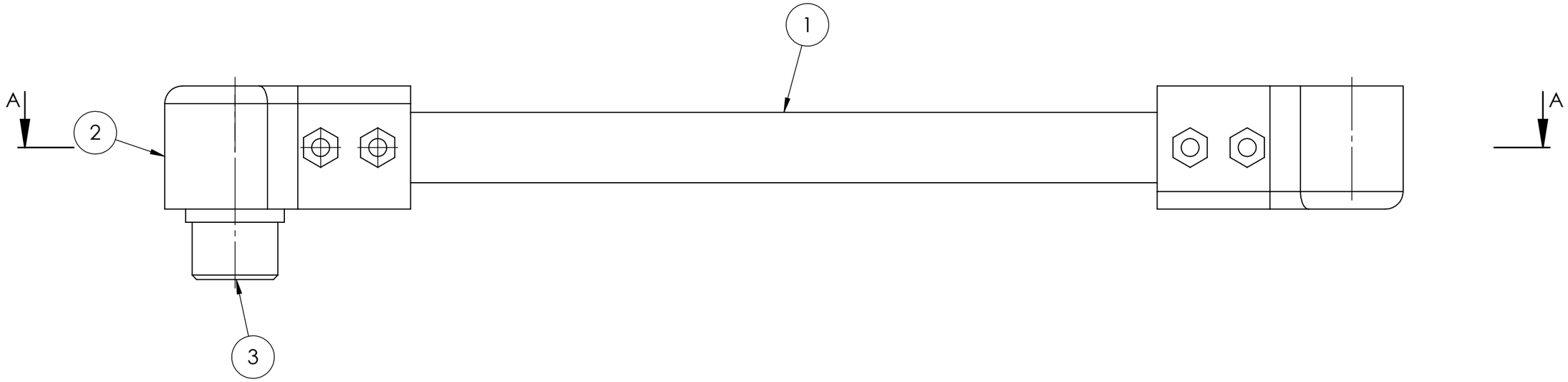
SUBCONJUNTO ACOPLE
BARRA MOTOR DER.

	FECHA	NOMBRE
DIBUJO		PALOMA
APROBO		BARAT

ESCALAS
2:3

N° 0012





N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	mar largo/CANTIDAD
1	BARRA MADERDA	DIAMETRO 16MM	1
2	PIEZA ARTICULACION		2
3	BUJE SEPARADOR TIPO 1		1

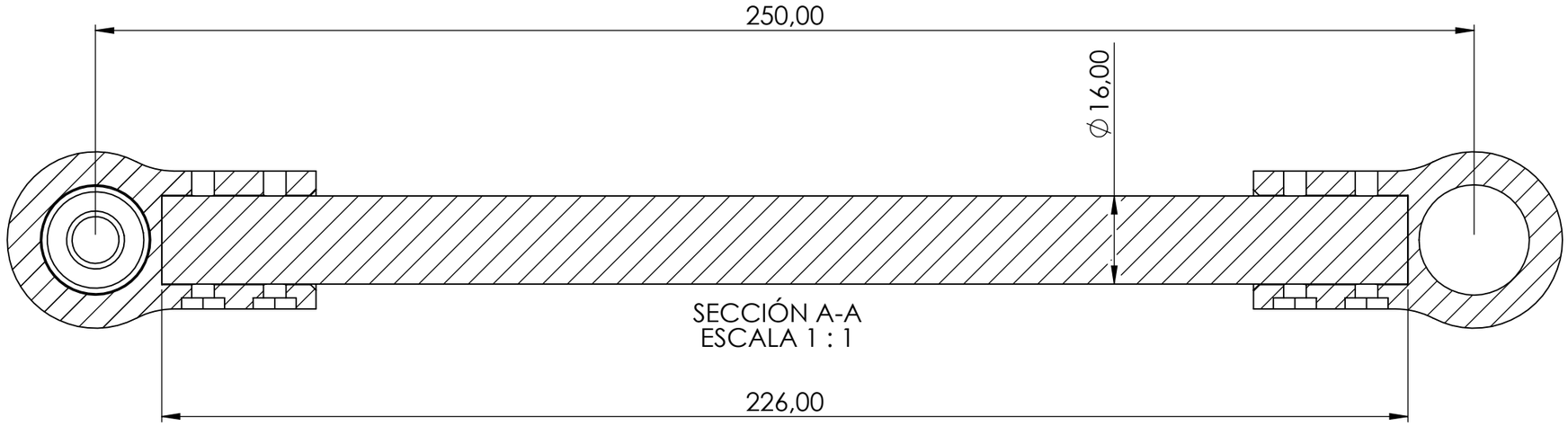
DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

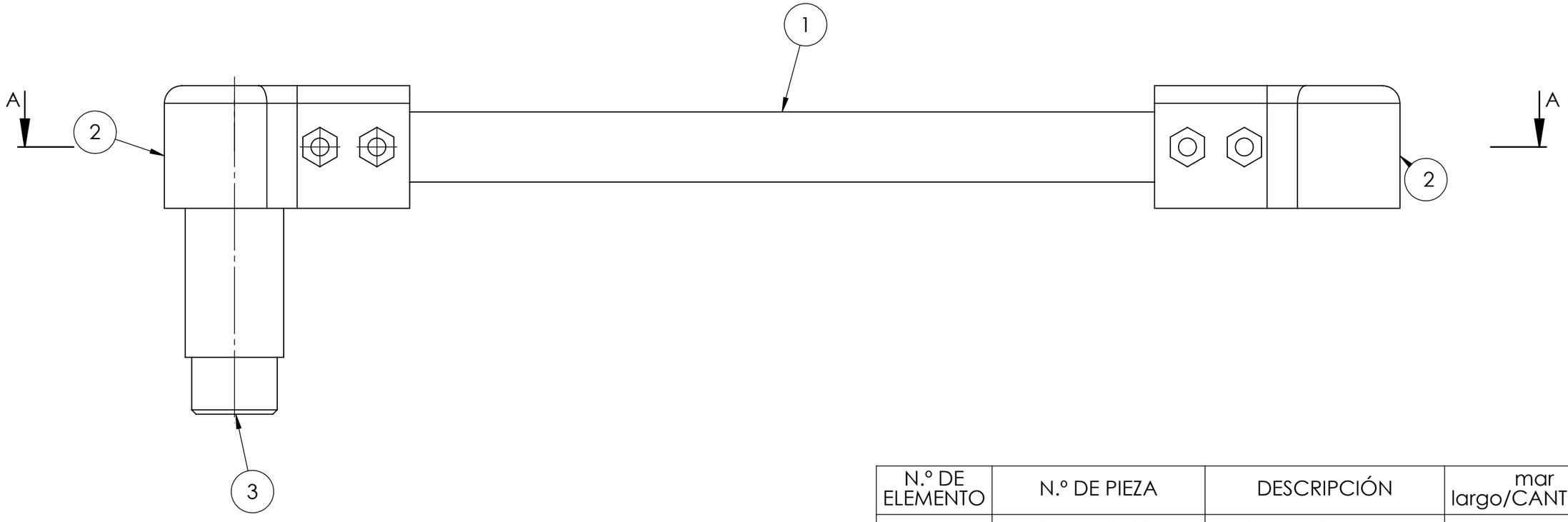
SUBENSAMBLE
BARRA FLOTANTE

	FECHA	NOMBRE
DIBUJO		PALOMA
APROBO		BARAT

ESCALAS
1:1

N° 0201





N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	mar largo/CANTIDAD
1	BARRA MADERDA	DIAMETRO 16MM	1
2	PIEZA ARTICULACION		2
3	BUJE SEPARADOR TIPO 2		1

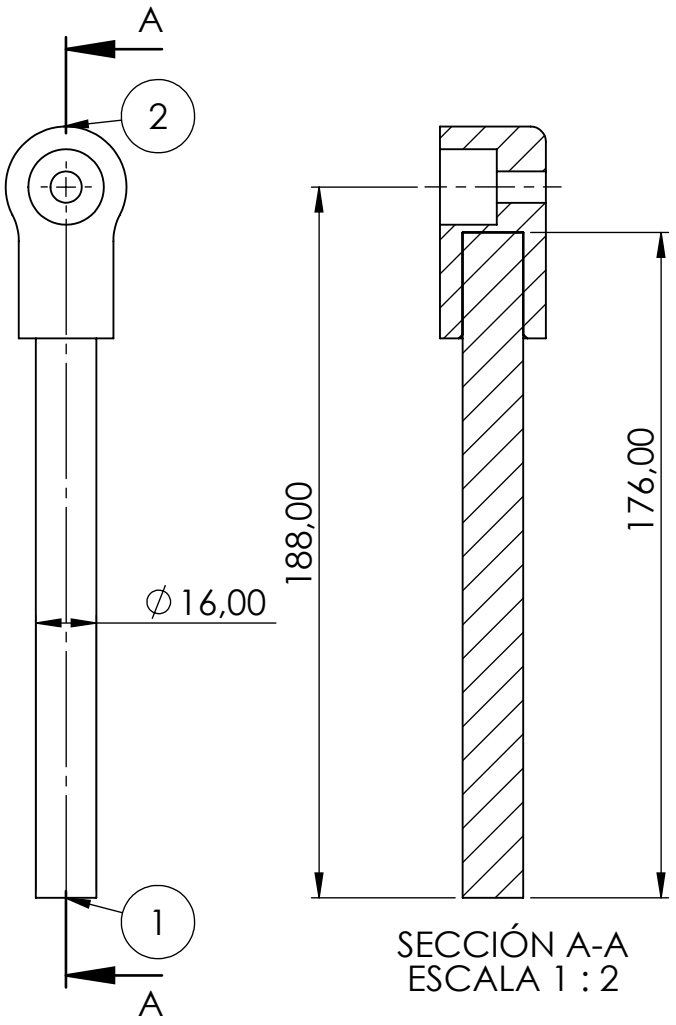
DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

SUBENSAMBLE
BARRA FLOTANTE

	FECHA	NOMBRE
DIBUJO		PALOMA
APROBO		BARAT

ESCALAS
1:1

N° 0200



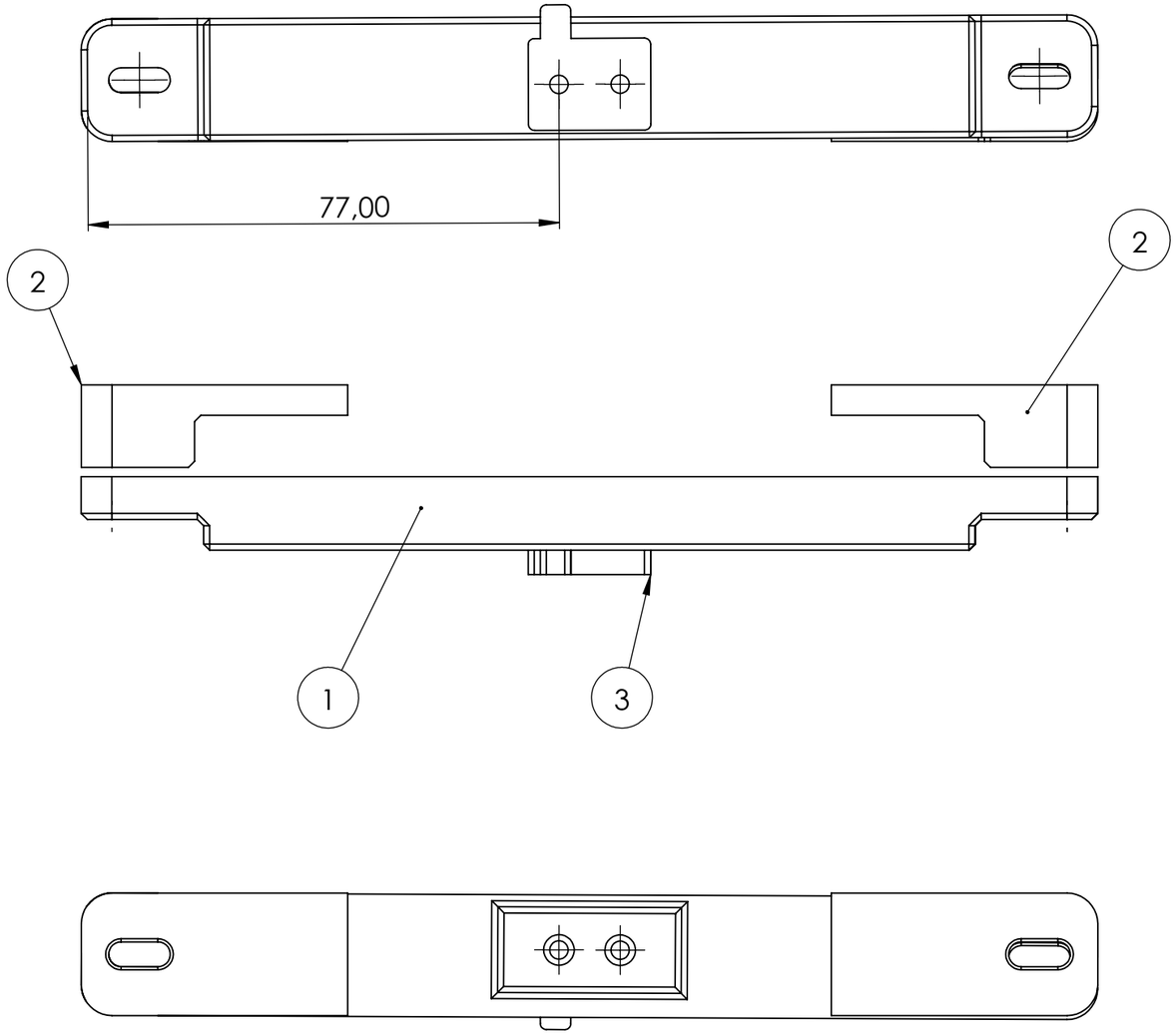
N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	BARRA MADERA	DIAMETRO 16MM	1
2	PIEZA NODO		1

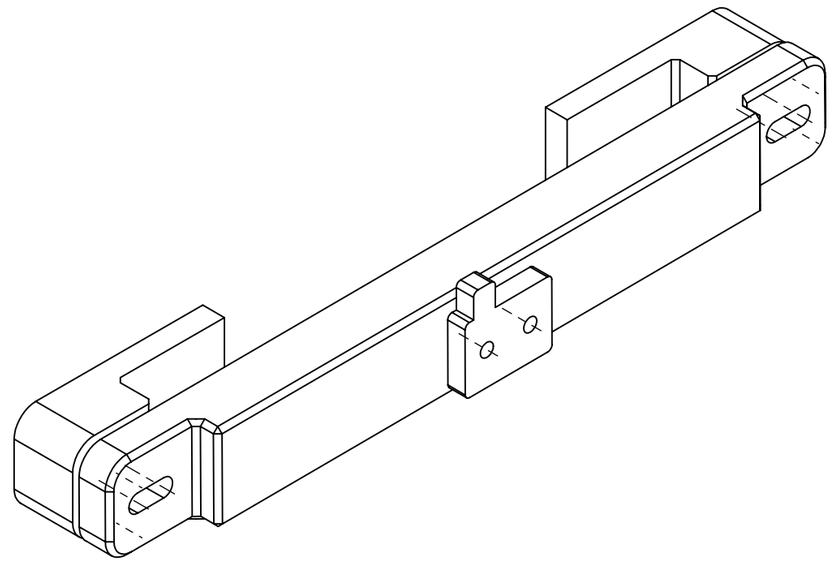
DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

sUBENSAMBLE
BARRA COMANDADA

	FECHA	NOMBRE	ESCALAS
DIBUJO		PALOMA	1:2
APROBO		BARAT	

N° 0021



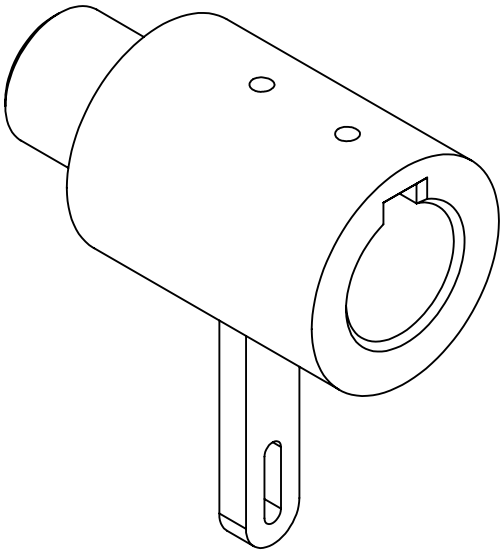


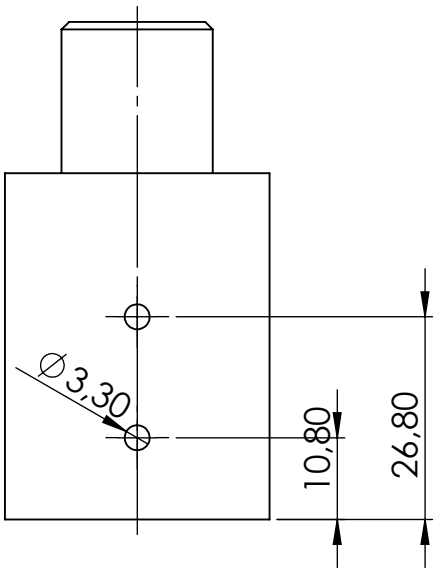
N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	sop. delant fin C		1
2	sop. tras fin c		2
3	findecarrera		1

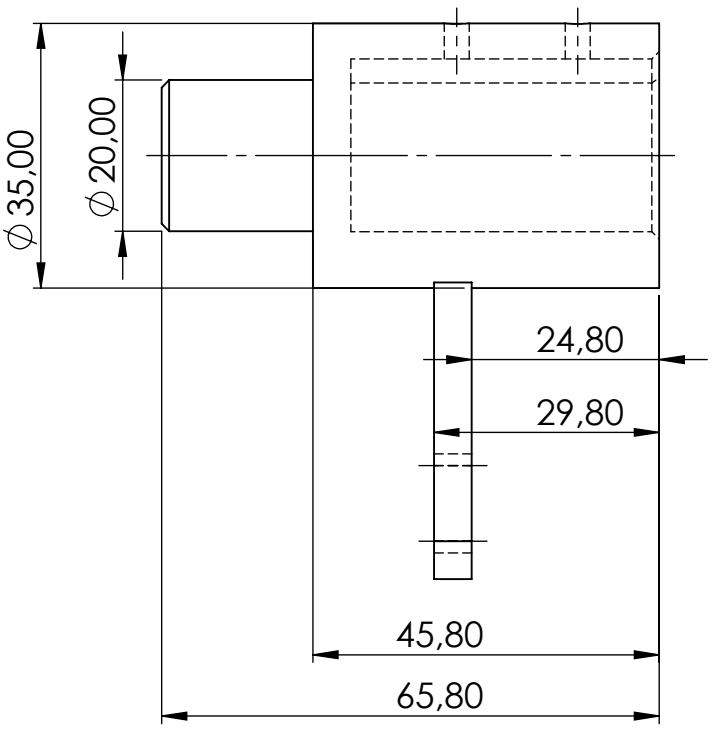
DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

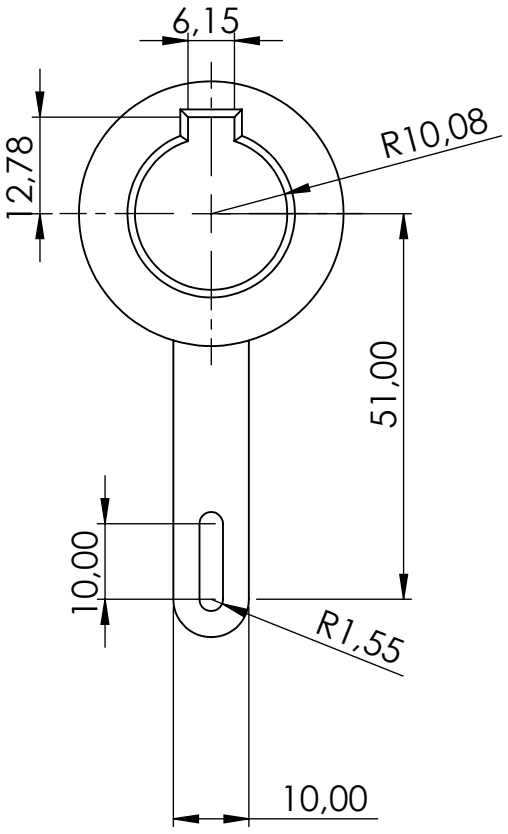
subconjunto fin de carrera

	FECHA	NOMBRE	ESCALAS	N° 0030
DIBUJO		PALOMA	1:1	
APROBO		BARAT		

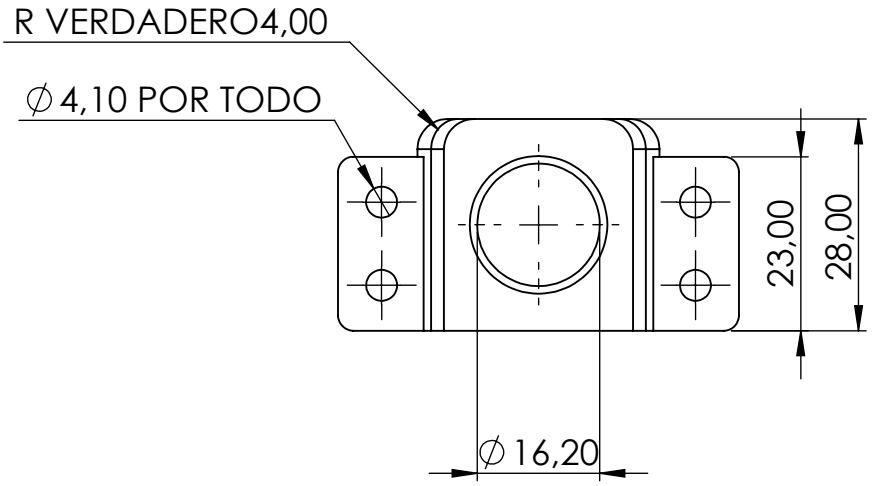


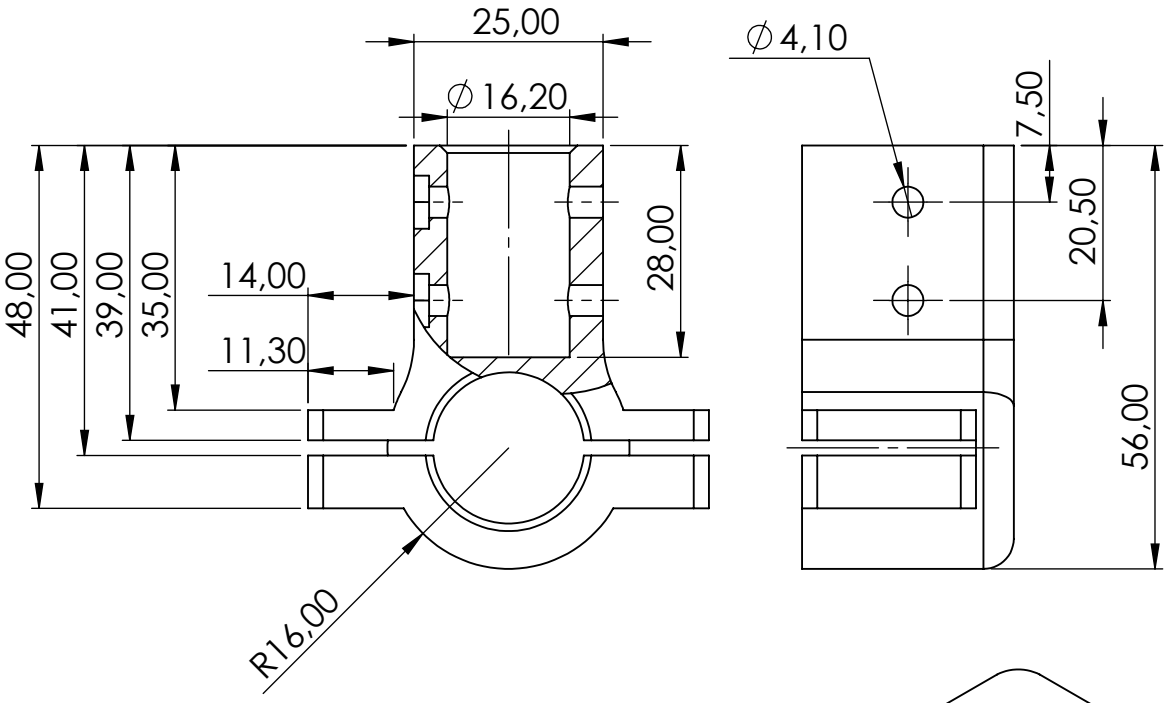


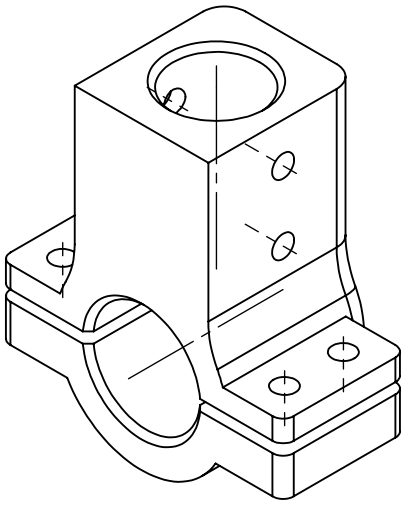




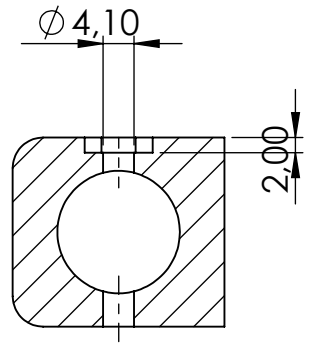
DISEÑO INDUSTRIAL ESCUELA INGENIERIA MECANICA ROSARIO				Acople motor	
	FECHA	NOMBRE	ESCALAS		N° 00012-1
DIBUJO		PALOMA	1:1		
APROBO		BARAT			



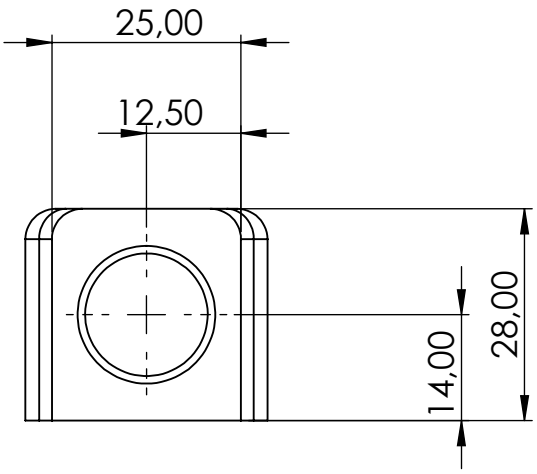


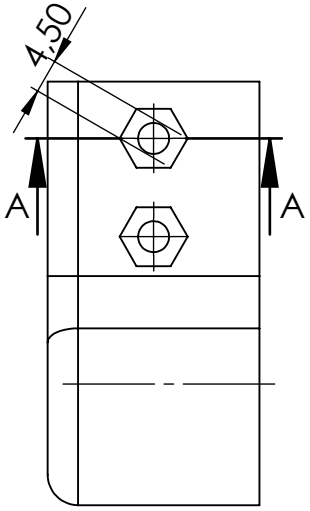


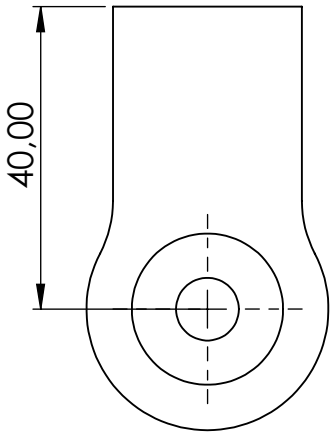
DISEÑO INDUSTRIAL ESCUELA INGENIERIA MECANICA ROSARIO				Acople Barra	
	FECHA	NOMBRE	ESCALAS		N° 00012-2
DIBUJO		PALOMA	1:1		
APROBO		BARAT			

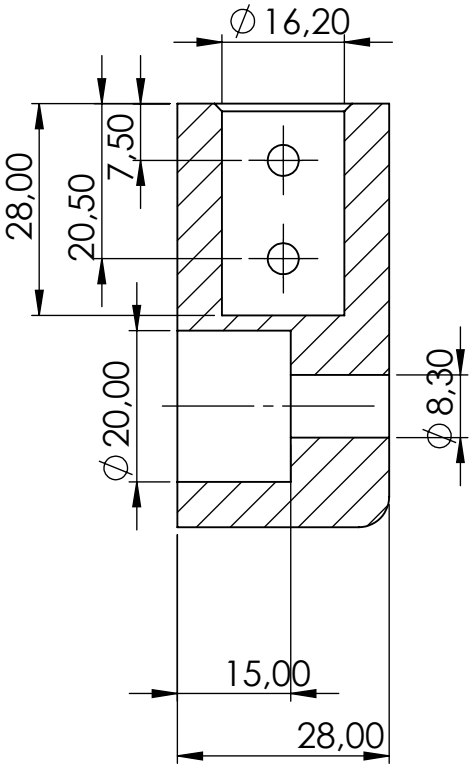


SECCIÓN A-A

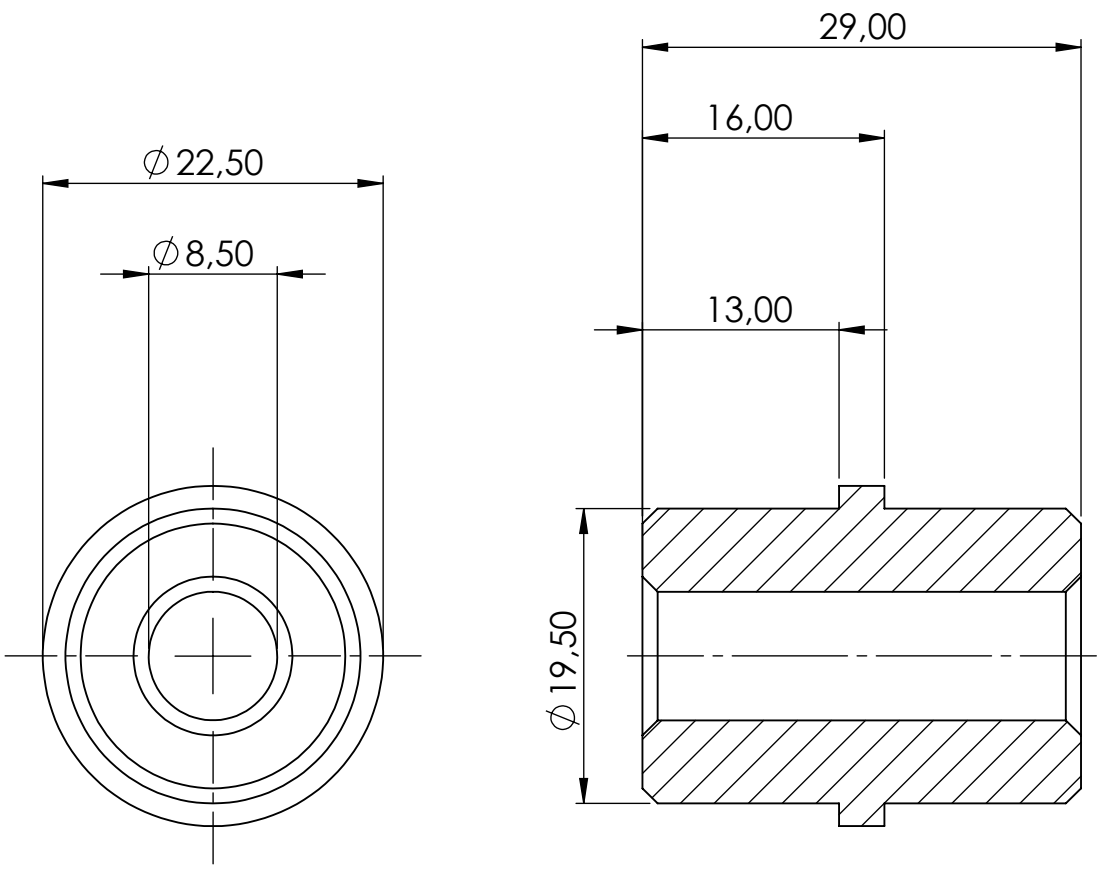








DISEÑO INDUSTRIAL ESCUELA INGENIERIA MECANICA ROSARIO				PIEZA NODO	
	FECHA	NOMBRE	ESCALAS		N° 00011-0
DIBUJO		PALOMA	1:1		
APROBO		BARAT			

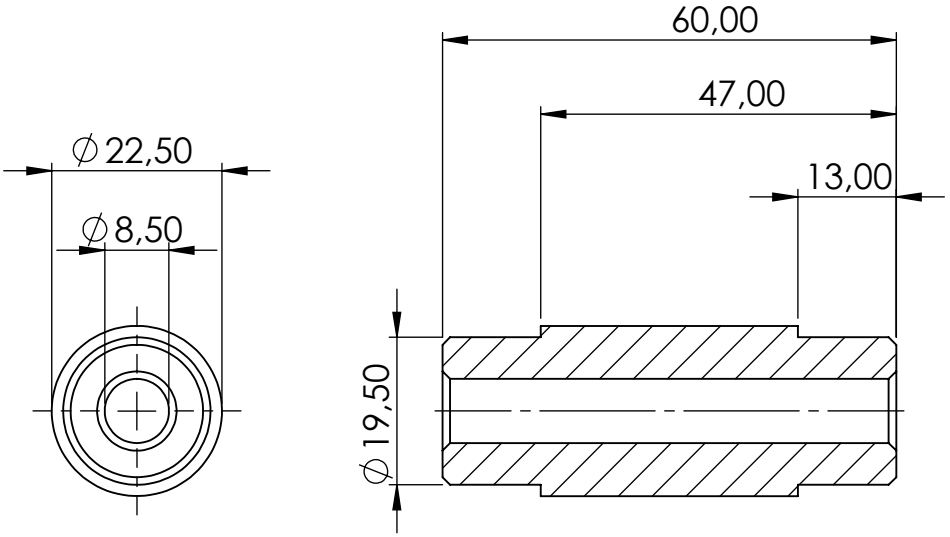


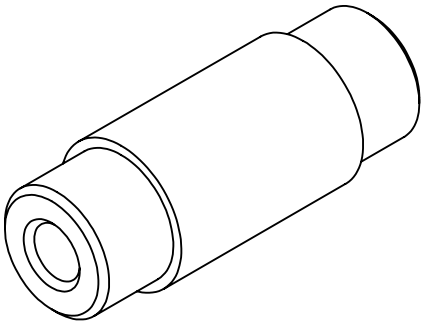
DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

PIEZA SEPARADOR TIPO 1

	FECHA	NOMBRE	ESCALAS
DIBUJO		PALOMA	2:1
APROBO		BARAT	

N° 00011-1



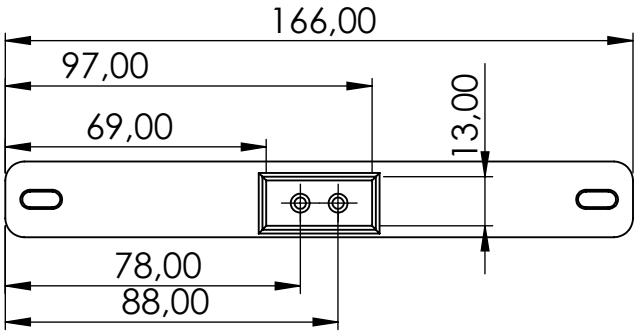


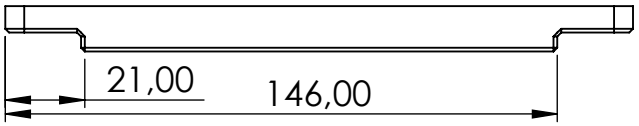
DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

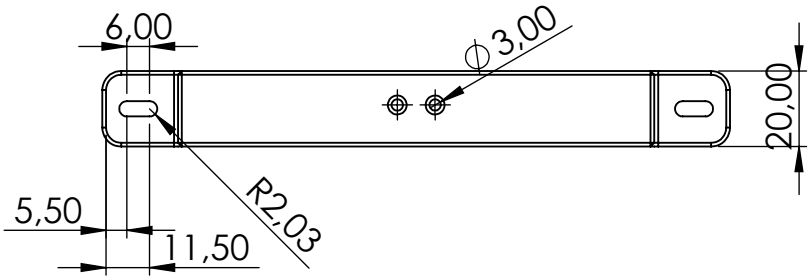
PIEZA SEPARADOR TIPO 2

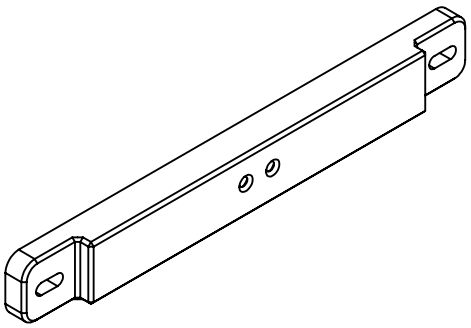
	FECHA	NOMBRE	ESCALAS
DIBUJO		PALOMA	2:1
APROBO		BARAT	

N° 00011-2

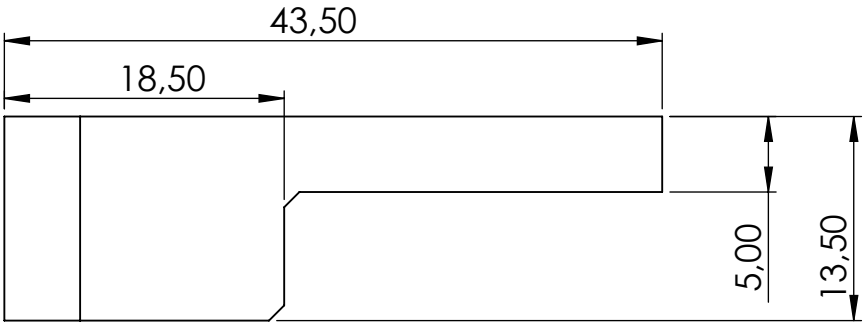


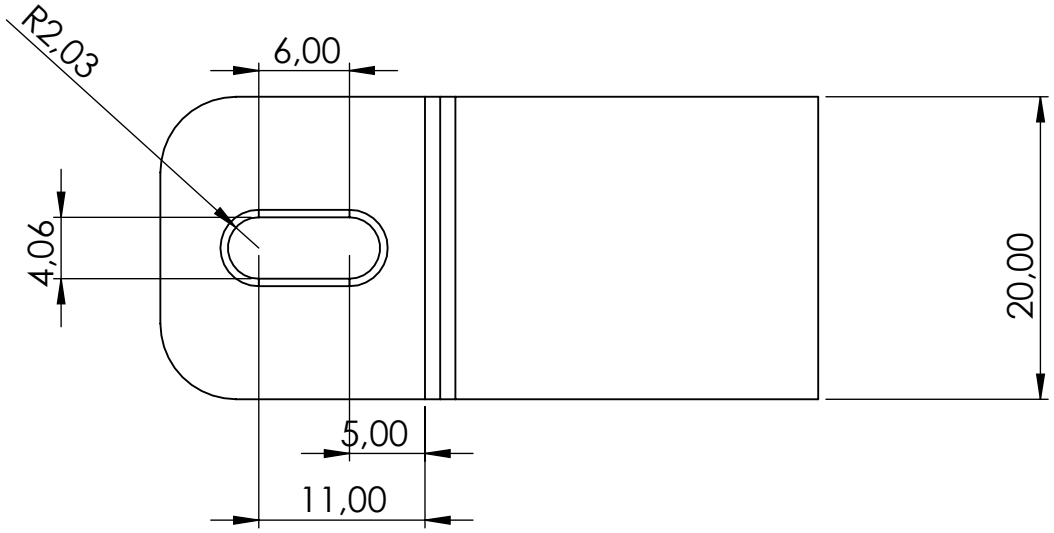


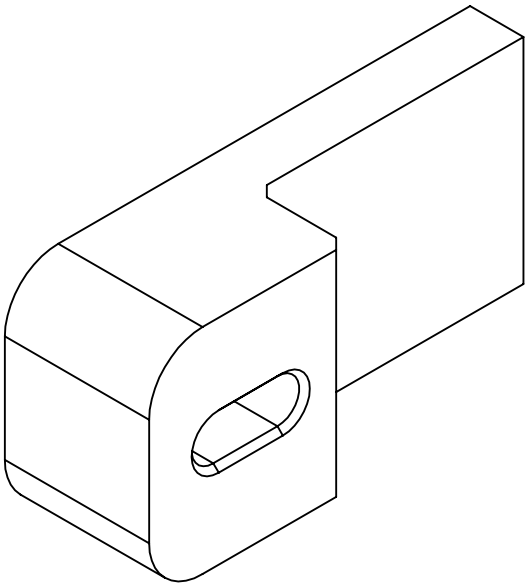




DISEÑO INDUSTRIAL ESCUELA INGENIERIA MECANICA ROSARIO				SOPORTE DELANTERO FIN DE CARRERA	
	FECHA	NOMBRE	ESCALAS		N° 0002-2
DIBUJO		PALOMA	1:2		
APROBO		BARAT			







DISEÑO INDUSTRIAL
ESCUELA INGENIERIA MECANICA
ROSARIO

soporte trasero fin de carrera

	FECHA	NOMBRE	ESCALAS
DIBUJO		PALOMA	1:1
APROBO		BARAT	

N° 0002-1