UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño e implementación de un manipulador móvil de bajo coste

Trabajo Fin de Máster

Máster Universitario en Ingeniería Mecatrónica

AUTOR/A: Shan , Xiangxiang

Tutor/a: Zotovic Stanisic, Ranko

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO E IMPLEMENTACION DE UN MANIPULADOR MOVIL DE BAJO COSTE

TRABAJO FINAL DEL

Máster en Ingeniería Mecatrónica

REALIZADO POR

XIANGXIANG SHAN

TUTORIZADO POR

RANKO ZOTOVIC STANISIC

CURSO ACADÉMICO: 2022/2023

Resumen

El brazo robot es un tipo de robot más común en aplicaciones industriales, donde se ejecuta movimientos repetitivos mediante instrucciones programadas previamente, con ventaja de rapidez, preciso y económico, pero con restricciones de la zona de trabajo, debido e l robot está colocado en un puesto fijo. Últimamente, la aplicación del robot no solo en las líneas de producción, sino también en la vida cotidiana, por ejemplo, el robot móvil en los almacenes de Amazon, gracias al IoT, nada más que realizar el pedido, dichos robot se va automáticamente a la zona donde está situado el producto y lo lleva a la cinta de transportadora para enviar el paquete. También existe los robots camarero que sirven a los clientes en los restaurantes, o los robots domésticos que realiza tarea de limpieza.

En el presenta trabajo, se pretende diseñar un brazo robot móvil (o manipulador móvil) de bajo coste para uso doméstico o los pequeños comercios mediante todos los conocimientos adquiridos durante el Máster Mecatrónica. Contiene varias etapas:

-Diseño mecánico en el SolidWorks.

-Análisis de la cinemática directa, inversa y velocidad del robot.

-Obtención del modelo dinámico a través del método Lagrange-Euler.

-Diseño del sistema de control.

-Comunicación entre el brazo robot y el controlador mediante puerto serie.

-Implementación de los algoritmos.

Las piezas para construir el robot son todos diseñado mediante programa de CAD, y posteriormente a impresión 3D, los materiales que se utiliza para la impresión son de polvo metálica(sinterizado) y metacrilato (los eslabones).

El brazo robot es de tipo 3 grados de libertad, con la que es suficiente para realizar tarea de pick and place, tiene el modo de funcionamiento automática, mediante el orden de pick and place enviado, a través de la coordenada cartesiana (XYZ) introducida por el usuario, la plataforma móvil se moverá automáticamente a la zona adecuada donde el brazo robot pueda alcanzar el objeto.

En la parte del control se puede ver por dos partes, el control del brazo robot y el control de la plata forma móvil. En la parte del brazo robot, se ha utilizado el método de la cinemática inversa y se construyó el modelo dinámico del brazo robot mediante el método de Lagrange-Euler. Para la base móvil, considerando como un vehículo diferencial, se aplicó el criterio de cinemática de robots móviles para el control.

Palabra clave: Brazo Robot, Robot móvil, Impresión 3D, Control, Bajo coste.

Abstract

The robot arm is a type of robot more common in industrial applications, where it executes repetitive movements through previously programmed instructions, with the advantage of speed, accuracy and economy, but with restrictions of the work area, because the robot is placed in a fixed position. Lately, the application of the robot not only in production lines, but also in everyday life, for example, the mobile robot in Amazon warehouses, thanks to the IoT, as soon as the order is placed, the robot automatically goes to the area where the product is located and takes it to the conveyor belt to send the package. There are also waiter robots that serve customers in restaurants, or domestic robots that perform cleaning tasks.

In the present work, we intend to design a low-cost mobile robot arm (or mobile manipulator) for domestic use or small stores using all the knowledge acquired during the Master Mechatronics. It contains several stages:

-Mechanical design in SolidWorks.

-Analysis of the direct and inverse kinematics and velocity of the robot.

-Obtaining the dynamic model through the Lagrange-Euler method.

-Design of the control system.

-Communication between the robot arm and the controller by means of serial port.

-Implementation of the algorithms.

The parts to build the robot are all designed by CAD program, and subsequently to 3D printing, the materials used for printing are powder metal (sintered) and methacrylate (for the links of robot arm).

The robot arm is of type 3 degrees of freedom, with which it is sufficient to perform pick and place task, it has the automatic operation mode, by the pick and place order sent, through the Cartesian coordinate (XYZ) entered by the user, the mobile platform will automatically move to the appropriate area where the robot arm can reach the object.

In the control part, it can be seen by two parts, the control of the robot arm and the control of the moving platform. In the robot arm part, the inverse kinematics method has been used and the dynamic model of the robot arm was constructed by Lagrange-Euler method. For the mobile base, considering it as a differential vehicle, the criterion of mobile robot kinematics was applied for control.

Keyword: Robot Arm, Mobile Robot, 3D Printing, Control, Low Cost.

INDICE

1.Objetivo

El objetivo de este proyecto es diseñar un brazo robot móvil de 3 grados de libertad para el pick and place. Implementando todos los conocimientos adquiridos durante el periodo que se ha estudiado en el Máster en Ingeniería Mecatrónica de UPV. El proyecto contiene diferentes fases para alcanzar el objetivo.

La **primera fase** es diseño de las piezas necesario del modelo robot mediante la tecnología de impresión 3D, utilizando el programa SolidWorks, familiarizando a las nuevas tecnologías de impresión 3D y los programas de CAD.

La **segunda fase** es selección de los componentes necesario para montar el manipulador móvil, como el modelo del motor, el microcontrolador y los componentes electrónicos necesarios para el control del brazo robot móvil.

La **tercera fase** es la etapa de control, se realiza la cinemática directa, cinemática inversa y la cinemática de velocidad para tener un modelo matemático del manipulador móvil, a parte, también la obtención de los parámetros del controlador PID en bucle cerrado para tener un control de respuesta rápida y sin error de posición.

En la parte del **brazo robot**, se ha utilizado el método de la cinemática inversa para obtener los ángulos que debe girar cada articulación para llegar a la posición objetiva y se construyó el modelo dinámico del brazo robot mediante el método de Lagrange-Euler para realizar análisis de la fuerza en diferentes modos de funcionamiento, sobre todo para la compensación de gravedad del brazo robot. La trayectoria que se ha generado para el brazo robot es una trayectoria cubica.

Para la **base móvil**, considerando que es un vehículo diferencial, se aplicó el criterio de cinemática de robots móviles, para el control también se ha generado una trayectoria cubica en periodo de muestreo pequeño, durante cada muestreo el algoritmo calculará la posición actual y la posición objetiva y se realiza una comparación, el resultado de la comparación decidirá el movimiento que se va a realizar la plataforma móvil.

La **cuarta fase** es partiendo de los resultados obtenido en la anterior fase, generar los algoritmos necesarios de control tanto para el brazo robot o para la plataforma móvil en lenguaje c que se implementa en el microcontrolador.

Una vez completado todas las fases anteriores, se procede el montaje final del proyecto y realizar el testeo del KIT para comprobar el correcto funcionamiento del manipulador móvil diseñado.

2.Diseño mecánico del robot

El primer paso consiste en el diseño mecánico del manipulador móvil, para el diseño mecánico del modelo se ha utilizado el programa Solidwork, es un programa CAD (Diseño Asistido por Computadora) para realizar modelados mecánicos en 2D y 3D.

El modelo robot que se pretende a diseñar es un tipo de brazo robot de 3 grado de libertad, a diferencia con los otros brazos robot tradicionales, nuestro robot contiene una base móvil, eso implica un área de trabajo sin restricciones de zonas.

En la siguiente imagen tenemos el modelo en 3D del motor Dynamixel XL330-M288-T, la dimensión es de 20x34x26mm. Donde se observa la circunferencia delantera es el rotor del motor.

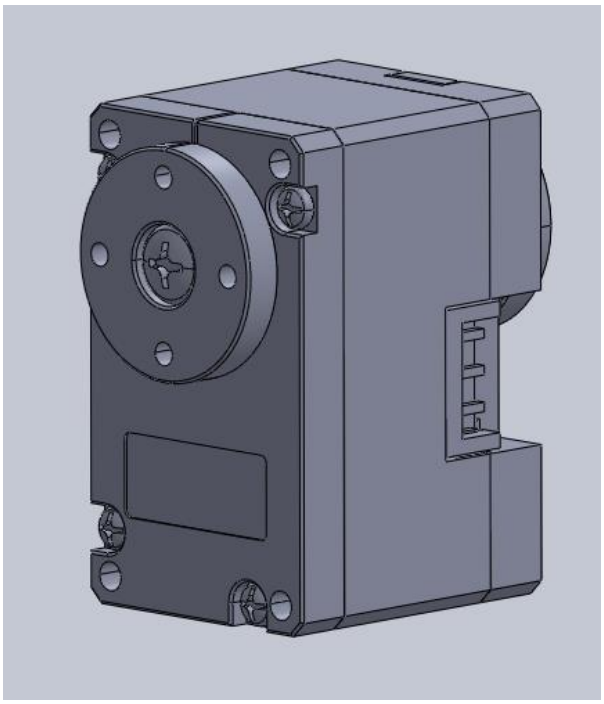


Figura 1. Servomotor Dynamixel

Los componentes más importantes son las piezas que acoplan con el motor, tras diseño de varias versiones, considerante el efecto de la gravedad, rozamiento y dimensión global del robot, las piezas que acoplan con el motor son las siguientes, la pieza de la izquierda tiene función de sujetar el motor, y la pieza de la derecha es la que pieza móvil que se acopla en el rotor del motor, ambas este fabricado mediante sinterizado con los polvos metálicos.

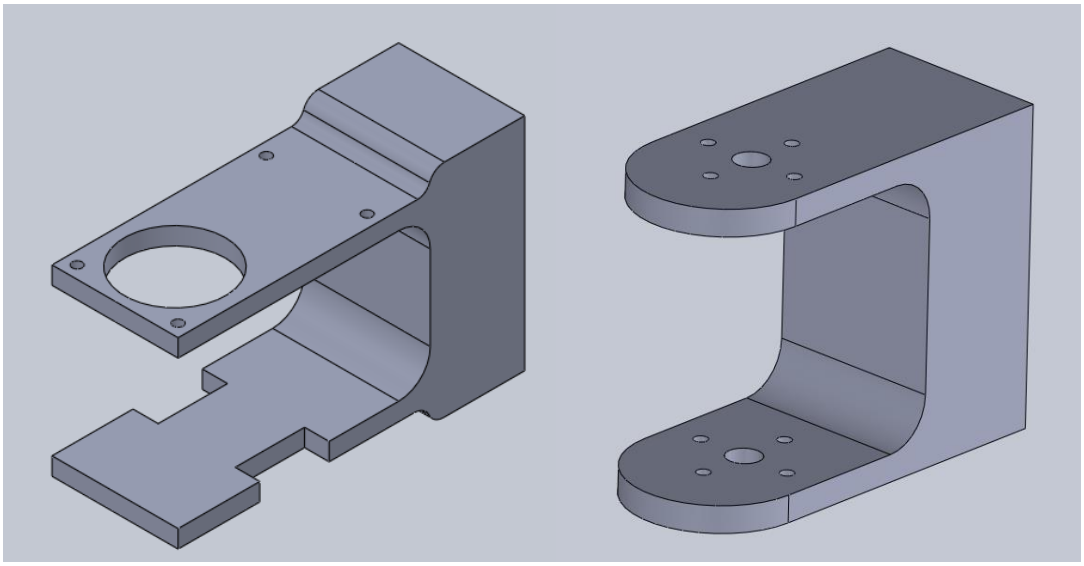


Figura 2. Pieza auxiliar para servomotor

Considerando el servomotor es un motor pequeño, el par que genera tambien es menor, solo de 0,52 Nm. Para reducir las cargas innecesaria , se ha diseñado el eslabon de forma “H”, dicha forma es estable y puede ahorrar tanto material de impresión 3D y tambien el peso del eslabon. Al tratarse de un componente plano, se ha realizado por corte de laser con el material metacrilato.

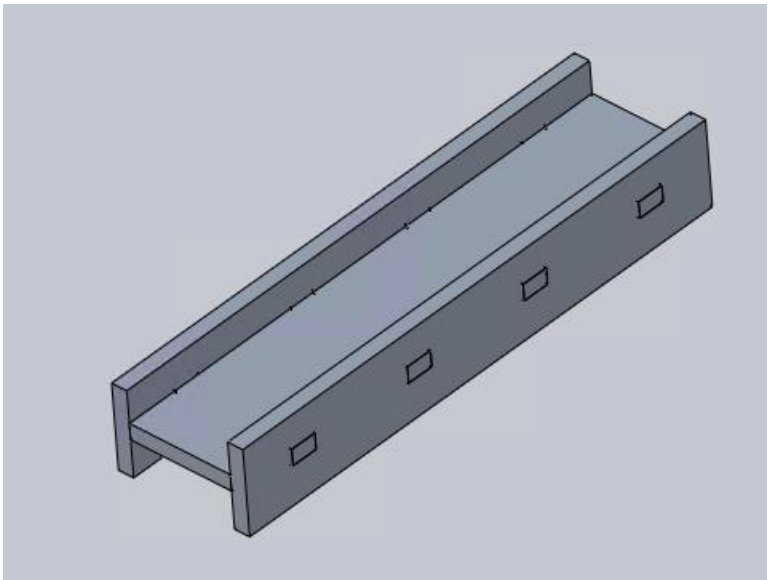


Figura 3. Eslabón del brazo robot

En la siguiente figura, muestra la pinza del manipulador movil, lo que se ha diseñado es una pinza simpre unido con un engranaje, y dicho engranaje esta acoplado directamente en el rotor del servomotor. La pinza se ha imprimido por sinterizado con el mismo material que se imprimio las piezas de acople del servomotor.

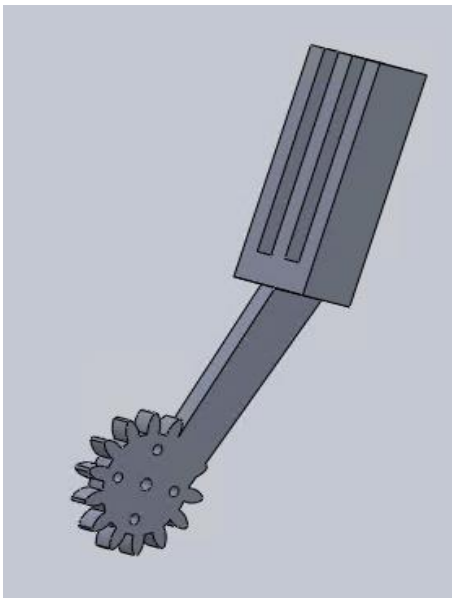


Figura 4. Pinza del manipulador móvil

Una vez diseñado todas las piezas necesarias, se ha realizado el ensambleje en el mismo programa, mediante el comando “relacion de posicion” se ha obtenido el modelo robot de la siguiente figura,en la parte del brazo robot tiene cuatro motores, tres de ellos son de la articulacion rotativa y una para la muñeca. En la parte inferior, esta la base movil del robot, tambien se coloca el microcontrolador en dicho sitio, tiene acoplado dos motores, y la rueda delantera es una rueda de tipo giro libre. Mediante el control de velocidad de los dos servomotores se consigue todos los tipos de movimiento, dicha parte se detallara en el apartado de la cinematica movil.

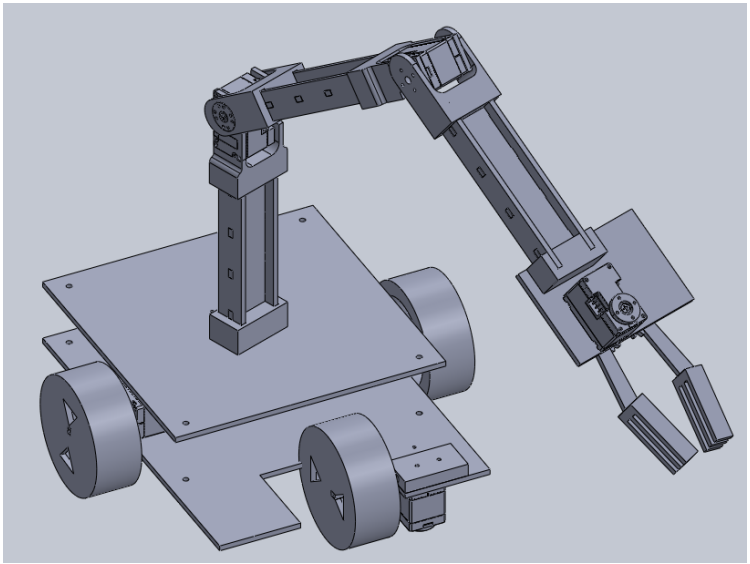


Figura 5. Ensamblaje del manipulador móvil

Por otra parte, se ha creado un dibujo en el propio programa SolidWorks para visualizar el manipulador móvil desde plano alzado, en la siguiente imagen muestra la longitud de los eslabones del brazo robot, donde las unidades son mm.

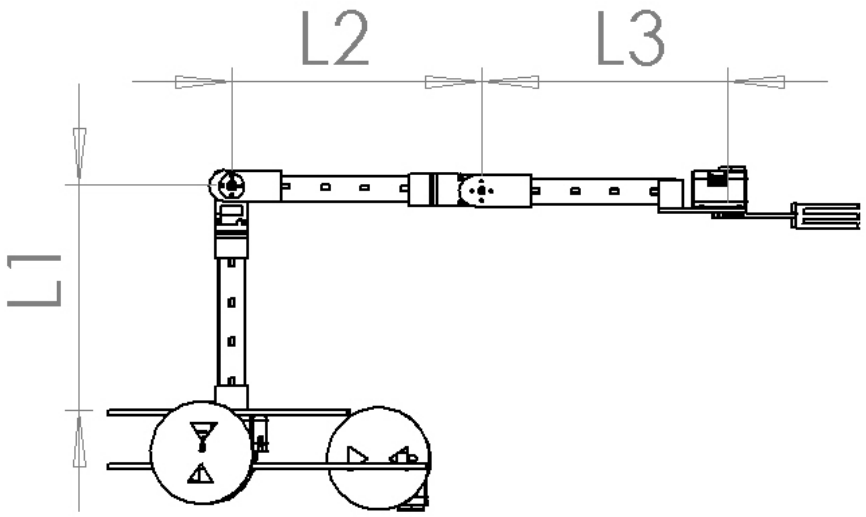


Figura 6. Plano alzado del manipulador móvil

También se ha tomada el peso de cada componente para posteriormente poder calcular el par necesario del motor, en la siguiente tabla, el peso del motor es provisional, ya que dependiendo del par calculo, se elegirá uno u otro.

Componente	**Peso(g)**
Eslabón de metacrilato	50
motor	30
acople del motor	50
Pinza	20

Tabla 1. Peso de los componentes

3.Selección de componentes

Uno de los requisitos más importante es el par necesario del motor, supongamos el peor caso, el motor de la articulación base, es el cual que tiene que mover todo el brazo robot, también considerando el efecto de la gravedad. Supongamos la pieza tiene 50g, con la aceleración y deceleración de la trayectoria trapezoidal de $\frac{\pi}{2}$ rad/s2 y una velocidad de π rad/s.

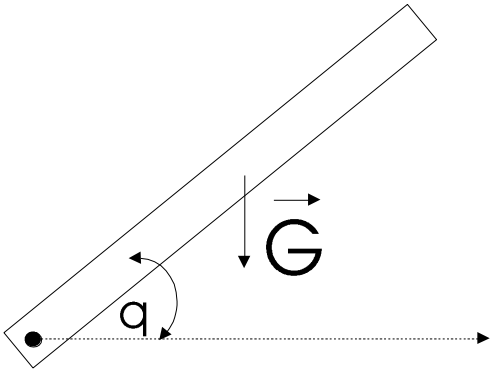


Figura 7. Eslabón con efecto de gravedad

La expresión del par se queda de siguiente forma:

$$M = \left(J_{ele} + J_{pieza}\right) \cdot \alpha + m_{motor} + G \tag{3.1}$$

J_{ele}:Momento inercia del conjunto de elemento

J_{pieza}: Momento inercia de la pieza respecto eje de giro

α: Aceleración angular

m_{motor}: Momento inercia del motor

m_{pieza}: Masa del elemento

G: Efecto de gravedad

l: Longitud del eslabón

Segundo el diseño del brazo robot, el servomotor que necesita generar mayor par es servo que está redondeado en la figura 8, Considerando el peor caso, cuando el eslabón 2 y el eslabón 3 están en perpendicular con la fuerza de gravedad, el ángulo es 0 y $\cos q = 1$

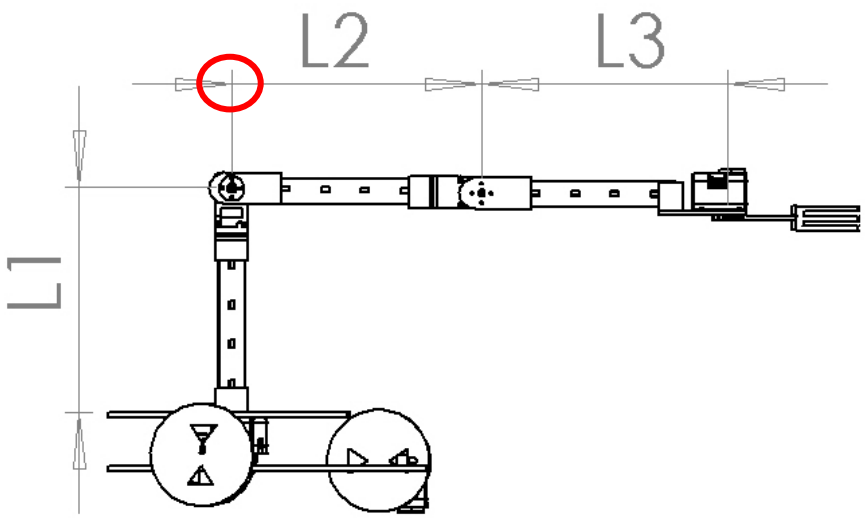


Figura 8. Plano alzado del manipulador movil

Sustituyendo los valores en la ecuación, el par mínimo requerido es:

$$J_{ele} = \frac{1}{2} \cdot m_{ele} \cdot r^2 = \frac{1}{2} \cdot 0{,}122 \cdot 0{,}017^2 = 0{,}001037\ kgm^3$$

$$J_{pieza} = m_{pieza} \cdot r^2 = 0{,}05 \cdot 0{,}3^2 = 0{,}0045\ kgm^3$$

$$m_{motor} = 0{,}00000638\ Nm$$

$$* G = \frac{m_2}{2} \cdot g \cdot l_2 + \frac{m_3}{2} \cdot g \cdot (2 \cdot l_2 + l_3) = 0{,}7467\ Nm$$

$$M_{req} = (0{,}001037 + 0{,}0045) \cdot \frac{\pi}{2} + 0{,}00000638 + 0{,}7467 = 0{,}7554\ Nm$$

Sumando un valor de seguridad de 20%, en el peor caso, el motor tiene que generar siguiente par.

$$M_{req} = 0{,}7554 + 20\% = \mathbf{0{,}906\ Nm}$$

*Esta ecuación se ha extraído de la ecuación de dinámica del robot.

Como se ha empezado trabajar con el motor Dynamixel XL330-M288-T, que tiene un par de parada de 0,52 Nm, no es suficiente para trabajar en la situación extrema, por lo que se ha diseñado un engranaje que se acople al motor para generar más par, en consecuente, se pierde la velocidad, la relación de transmisión necesaria se puede calcular de siguiente forma:

$$i = \frac{M_{req}}{M_{motor}} = \frac{0{,}906}{0{,}52} = 1{,}743$$

Por otra parte, se conoce que la potencia de transmisión de la entrada y de la salida es la misma, se puede obtener la siguiente expresión:

$$P_1 = P_2 \rightarrow T_1 \cdot \omega_1 = T_2 \cdot \omega_2 \tag{3.2}$$

De la ecuación 3.2 se puede obtener relación de transmisión entra número de diente, la velocidad angular y el par generado, como lo que representa en la siguiente ecuación:

$$i = \frac{Z_1}{Z_2} = \frac{\omega_2}{\omega_1} = \frac{T_1}{T_2} \tag{3.3}$$

Los engranajes que se ha diseñado tienen 40 y 22 dientes, que se obtiene una relación de 1,818 entre piñón y corona.

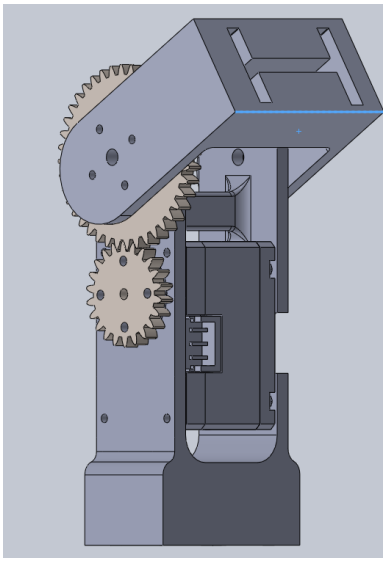


Figura 9. Servomotor con engranaje

Aplicando la ecuación 3.3, el nuevo par que genera el servomotor es:

$$M_{motor} = 1{,}818 \cdot 0{,}52 = 0{,}945\ Nm\ > 0{,}906\ Nm$$

Mediante el diseño mecánico, se ha resuelto el problema del par insuficiente del servomotor en el peor caso que es cuando el eslabón 2 y 3 está en perpendicular con la dirección de la fuerza de gravedad.

3.1. Servomotor

En el presente proyecto, para el control de las articulaciones del robot móvil diseñado, se utilizó el servo de la compañía de robótica Dynamixel, considerando los requisitos calculados en el apartado anterior, el modelo XL330-M288-T seleccionado cumple con los requisitos, puede generar un par de 0,52 Nm y la velocidad máxima llega hasta 3π rad/s. Es un tipo de servo de alta precisión, su resolución en posición es de 0.088º.

Figura 10. Servomotor XL330-M288-T Dynamixel

De la ficha técnica que se proporcionó el fabricante, se ha extraído las propiedades más destacadas del servomotor Dynamixel XL330-M288-T, en la siguiente tabla muestra las características más relevantes del servo motor.

Resolución	4096(pulso/rev)
Velocidad sin carga	**103 rpm**
Par de parada	**0,52 Nm**
tensión entrada	5V
Corriente de espera	17mA
Peso	18 g
dimensión	20x34x26 mm
Algoritmo de control	Control PID

Tabla 2. Propiedad del servomotor

El servo dispone distintos modos de control, que son los siguientes 6 modos de control, control de corriente, control de velocidad, control de posición, control de posición extendido, control de posición basando en corriente y control del PWM. En el propio servo esta ya integrado los controladores y los sensores correspondiente.

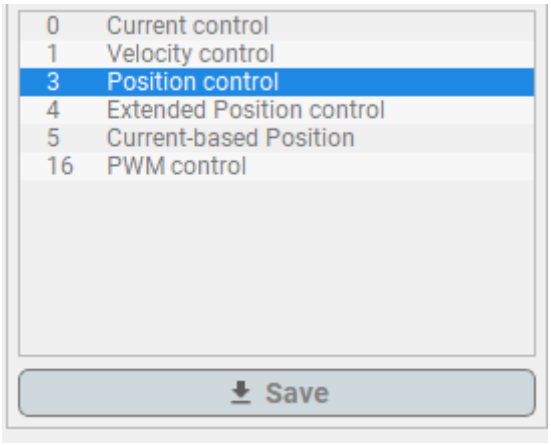


Figura 11. Modo de control del servomotor

Otra característica destacada es cada servo dispone un ID propio que la podemos asignar a la hora de configuración, de esta manera, con un mismo driver, conectando los servos en seria, se puede de forma distinta a cada motor.

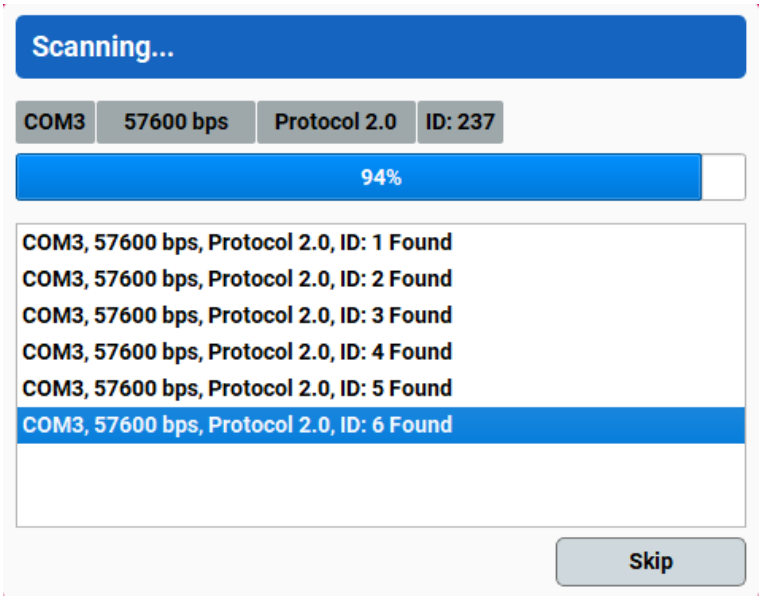


Figura 12. Resultado de la búsqueda de servomotor

Por otra parte, comparando con los servomotores más baratos que solo puede recibir señales de PWM para realizar el movimiento, la comunicación de los servos de Dynamixel se realiza a través de conexión serie mediante TTL Multidrop Bus, y el tipo de transmisión es de Half-Duplex, es decir se puede enviar y recibir datos, dicho tipo de transmisión nos facilitar leer las respuestas de la velocidad, posición, corriente en tiempo real. En la figura siguiente muestra las conexiones que dispone del motor Dynamixel.

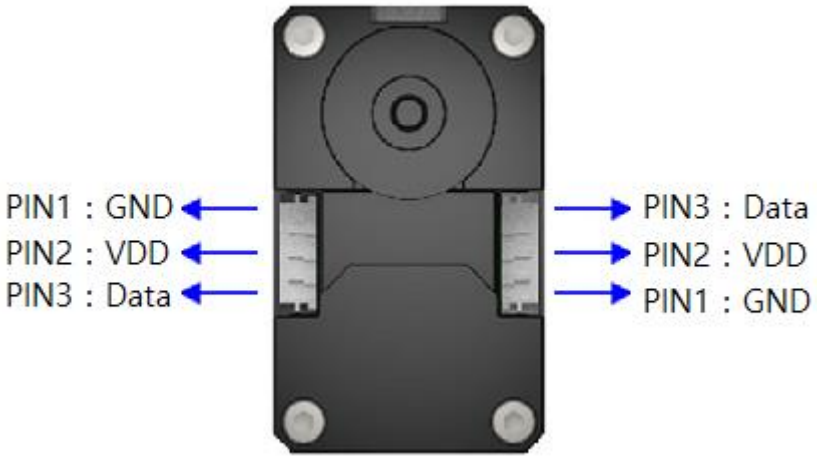


Figura 13. Conexión del servo XL330-M288-T

3.2. Driver

Para poder controlar el servo con PC u otros microcontroladores, es necesario un driver. Se utilizo el driver diseñado de la misma compañía, el U2D2 y U2D2 Power Hub.

U2D2

Es un convertidor de comunicación de USB que se utiliza para mandar datos al servo motor mediante conexión serie, para poder controlar desde PC o microcontrolador.

Figura 14. U2D2

U2D2 Power Hub

Es el driver que se combina con U2D2 para suministrar a los servos, dependiente del tipo de servo motor, la entrada de suministro puede ser de 3,5V hasta 24V, y la corriente máxima puede llegar hasta 10A.

En el caso de servo XL330-M288-T, se suministra con una tensión de entrada de 5V.

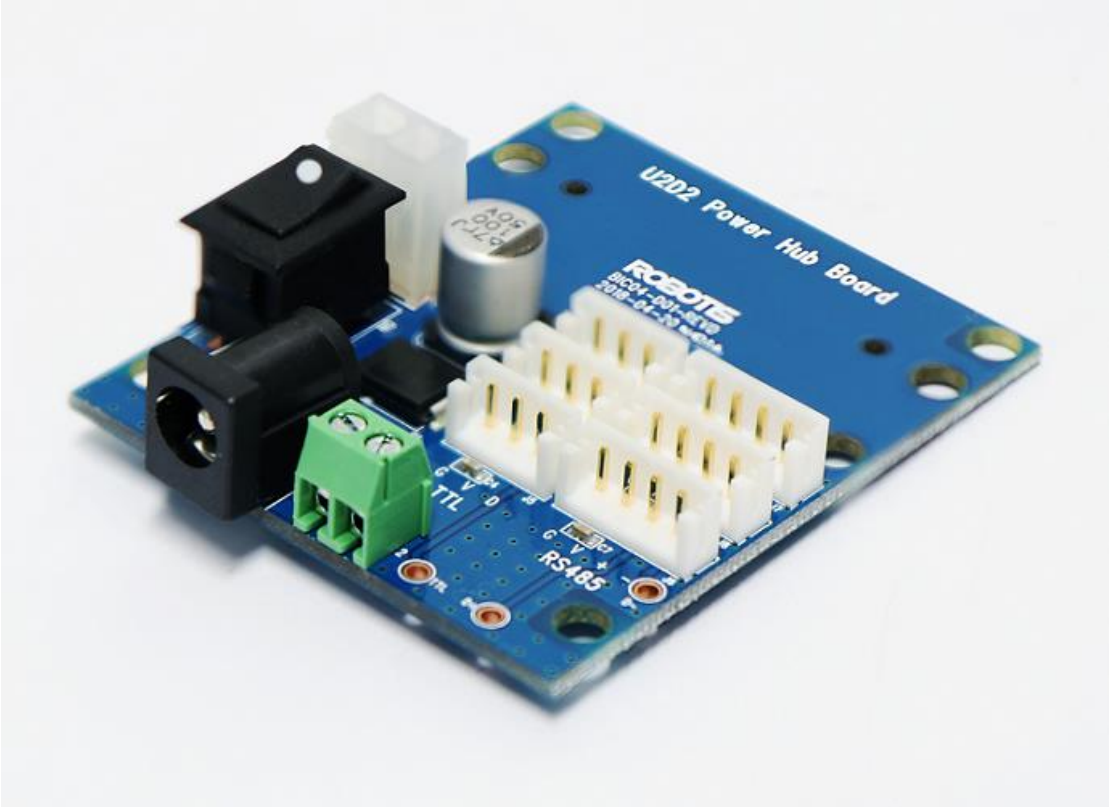


Figura 15. U2D2 Power Hub

3.3. Microcontrolador

Los microcontroladores son dispositivos de pequeño tamaño y cuyos circuitos integrados para ejercer tareas programada en su memoria. Un microcontrolador contiene los principales elementos de una computadora, que son la unidad central de procesamiento (CPU), la memoria y los periféricos de E/S. Dependiendo del tipo de microcontrolador, puede contener más bloques de funciones, como bloque de bluetooth, WI-FI, etc.

Para el control de manipulador móvil, en primer lugar, en el futuro se implementará la tecnología de la visión artificial, por lo que es necesario instalar una cámara para aplicar la visión artificial, a parte, una alta velocidad de resolución y ejecución del algoritmo en tiempo real, además, capaz de manipular por medio inalámbrico. Considerando todas las necesidades para procesar el control del manipulador móvil, se ha elegido el modelo RASPBERRY PI 4 model B, más que un microcontrolador, es un micro PC, basado en la arquitectura ARM. La figura siguiente es una placa de Raspberry que se implementó en el presente proyecto.

Figura 16. Raspberry Pi 4 model B

Por otra parte, Raspberry cuenta con un sistema de código abierto, de manera que todo el mundo pueda contribuir una parte, y los sucesores pueda desarrollar más profundo basando los códigos abiertos existentes.

En el caso del servo Dynamixel XL330-M288-T, el fabricante ya nos proporcionó las librerías e instrucciones básicas para controlar el servo en la plataforma GitHub, lo que se desarrolla en este proyecto es implementar el algoritmo de control para la cinemática inversa y la cinemática móvil, y posiblemente en el futuro implementar la visión artificial para obtener las coordenadas cartesianas del objeto en tiempo real.

Los pines de entrada y salidas de Raspberry están representados en la siguiente figura, junto con la tabla de las especificaciones más destacadas.

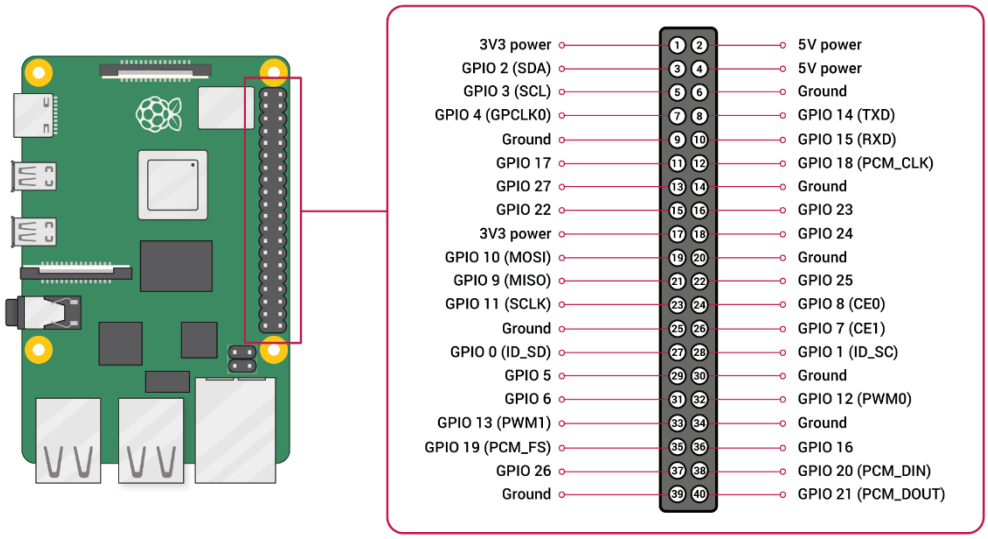


Figura 17. Pines de entradas y salidas del Raspberry

Processor:	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory	4GB
Connectivity:	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports
GPIO:	Standard 40-pin GPIO header
Video & sound:	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
SD card:	Micro SD card slot for loading operating system and data storage
Input/output power:	5V DC via USB-C connector (minimum 3A) 5V DC via GPIO header (minimum 3A)
Environment:	Operating temperature 0–50ºC

Tabla 3. Características del Raspberry

3.4. Batería para Raspberry

Es necesario implementar un sistema de alimentacion para el Raspberry ya que el manipulador diseñado no es fijo, sino movil. Se ha utilizado la siguiente placa con su bateria para alimentar el microcontrolador.

El modelo se llama Geekworm Raspberry Pi 4 UPS X703, utiliza las baterías recargables de 18650. Las especificaciones son los siguientes junto con la imagen.

-Inteligente sistema de alimentación ininterrumpida (UPS)
-Apoya 3A a través de tipo C hembra de carga rápida
cable adicional?
-BATERÍA INTEGRADA circuito de protección
-Integrado sobre la protección actual y de protección
-En placa 4 verde LED indican la batería de carga y descarga de los niveles de 25% de 50% en 75% y 100%
-En placa LED rojo mostrar el estado de carga de la batería o completamente cargado
-En botón de control de encendido/apagado (prensa-en mantenga el botón presionado al menos 2s)
-Permite la entrada de energía a través de tipo C hembra o XH2.54 conector
-En-2-Pin PH2.0 conector permite externo interruptor de potencia
-No apoya software seguro cierre

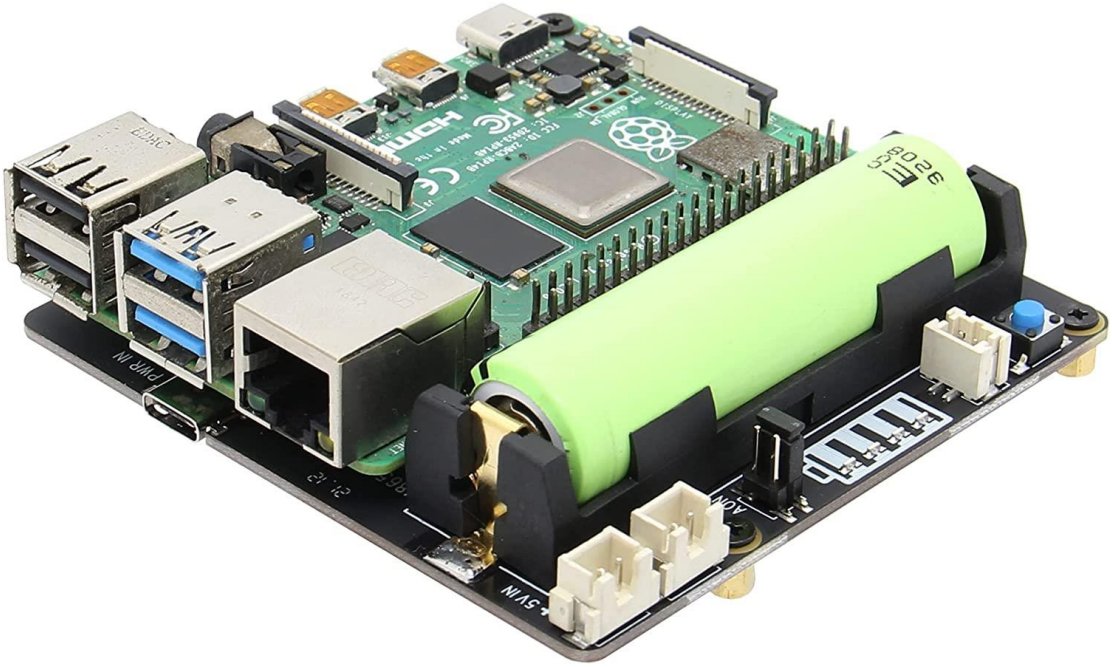


Figura 18. Placa base para alimentar Raspberry

3.5. Batería para motores

La batería que se ha seleccionado para suministrar los motores es de tipo polímero de litio, o también se puede llamar LiPo, dichos baterías son de tipo recargables y normalmente está compuesta por varias celtas para poder aumentar la corriente y la tensión de salida. La tensión de una celta de batería de LiPo es de 3,2-4,2 V, según la ficha técnica de los servomotores, la tensión de entrada es de 5V de manera que se ha seleccionado la siguiente batería de LiPo de 2 celtas que tiene una tensión de salida mayor que 5V con una capacidad de 5200mAh, suficiente para poder estar funcionando el manipulador móvil a rendimiento máximo durante 3 o 4 horas.

Figura 19. Batería LiPo

En la siguiente tabla muestra los datos destacados de dicho batería.

Capacidad	5200mAh
Tensión salida	6,4-8,4V
Tipo de batería	LiPo
Dimensión(mm)	138x47x25
Peso(g)	250

Tabla 4. Características de la batería LiPo

3.6. Step-Down (Convertidor Buck)

En el apartado anterior, se ha optado la batería de LiPo de 2 celtas que proporciona una tensión de salida entre 6,4 y 8,4 V, para poder alimentar a los servomotores con 5V es necesario implementar la electrónica de potencia, en este caso un convertidor Buck (Reductor) para bajar la tensión de la batería LiPo a 5V.

Un convertidor Buck está formado por un interruptor, un diodo, una bobina y un condensador. El esquema eléctrico del convertidor está representado en la siguiente figura, mediante el control del interruptor S, se puede controlar la tensión de la salida.

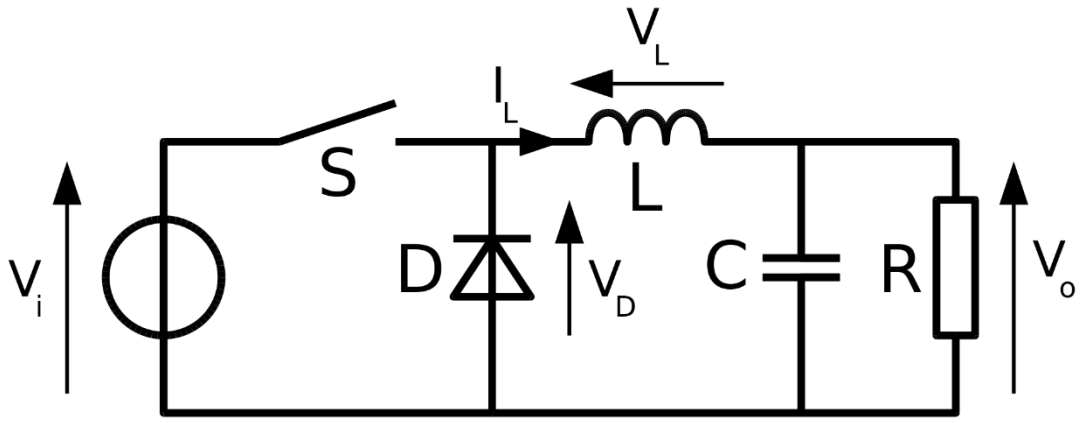


Figura 20. Circuito Buck

La relación entre la tensión de la salida y la tensión de entrada viene representada por la siguiente ecuación:

$$V_o = \delta \cdot V_i \tag{3.6}$$

V_o: Tensión de salida

δ: Ciclo de trabajo (Duty cycle)

V_i: Tensión de entrada

Para el presente proyecto, se ha seleccionado un Step-Down para bajar la tensión de la batería a 5V, en la siguiente figura se puede observar la forma del componente seleccionado, mediante el ajuste del potenciómetro se puede regular la tensión de la salida.

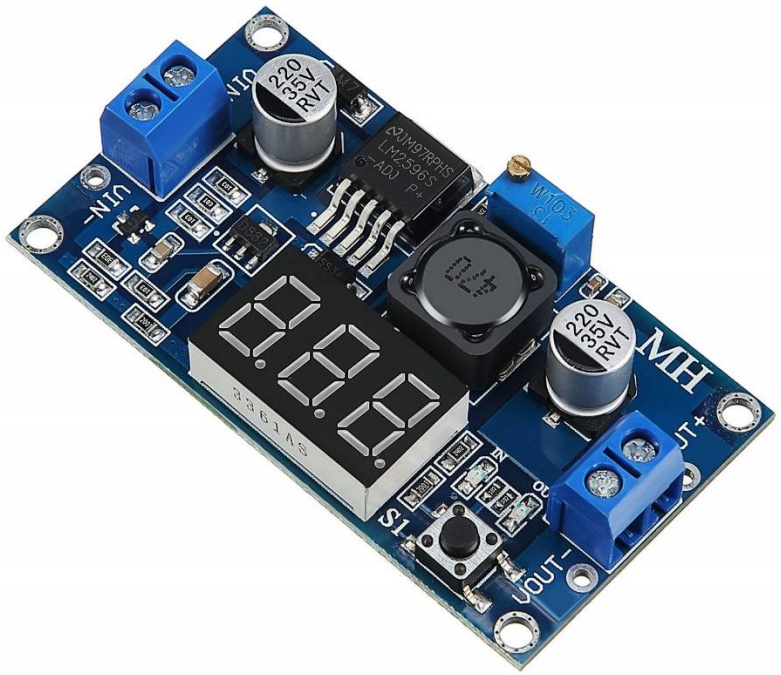


Figura 21. Step-Down

En la siguiente tabla muestra los datos destacados de dicho Step-Down.

Tensión de entrada	4V-40V
Tensión salida	1,25V-37V
Corriente máxima salida	2A
Frecuencia de conmutación	150 kHz
Temperatura de trabajo	-45ºC a 85ºC
Máxima eficiencia de conversión	92%

Tabla 5. Caracteristicas de Step-Down

4. Configuración inicial del servo

En este apartado, consiste en testear los distintos modos de funcionamiento y ajustar los valores del controlador para llegar a tener un resultado satisfecho.

4.1. Modo control de posición

El modo principal que se va a utilizar para el robot móvil es el control de posición, donde el algoritmo de control calcula el ángulo que debe girar cada articulación mediante la cinemática inversa, y enviar el resultado a servo. De esta manera, es imprescindible que el error de posición sea nulo.

En la siguiente figura muestra el comportamiento del servo acoplado con un eslabón, sin ajustar el controlador (PID con valores de defecto del fabricante), se observa claramente hay un error de posición 20 unidades.

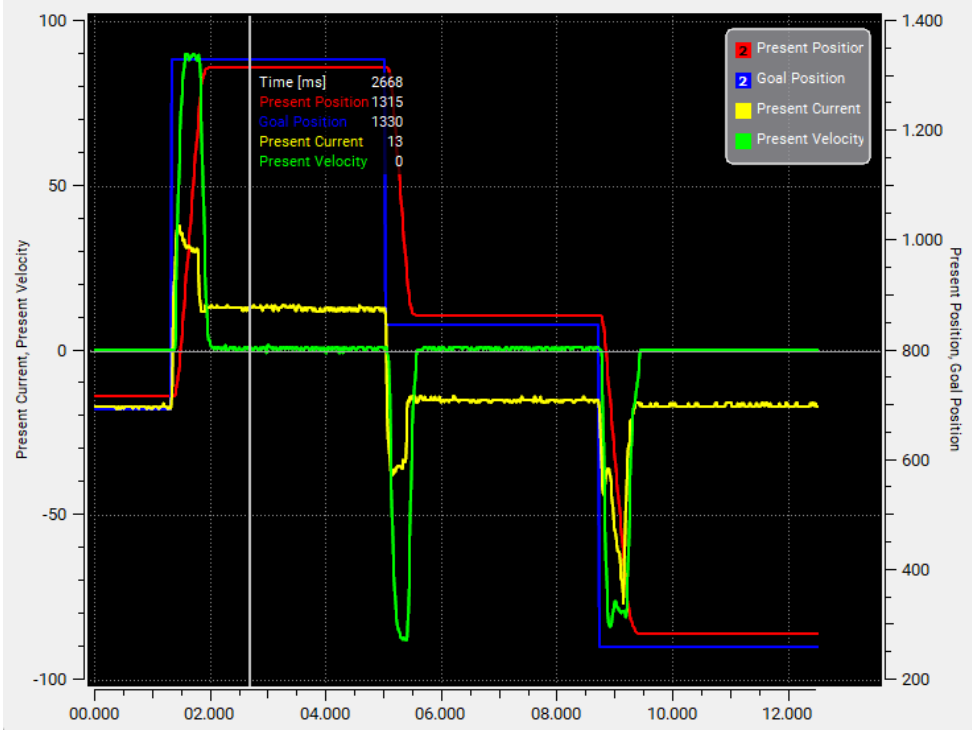


Figura 22.Comportamiento del Servo motor SIN CONTROL

Según los datos proporcionado por fabricante, el control de posición tiene siguiente esquema, la entrada del sistema es Goal PWM y Gold Pos, el Goal PWM es el límite de PWM deseado, y el Goal Pos es la posición deseada. En el control esta implementado un controlador PID donde se puede ajustar los valores Kp,Ki y Kd para regular el controlador, y por otra parte la pre-realimentacion, que tiene como objetivo de mejorar el comportamiento del sistema, sobre todo mantener la estabilidad cuando hay una perturbación.

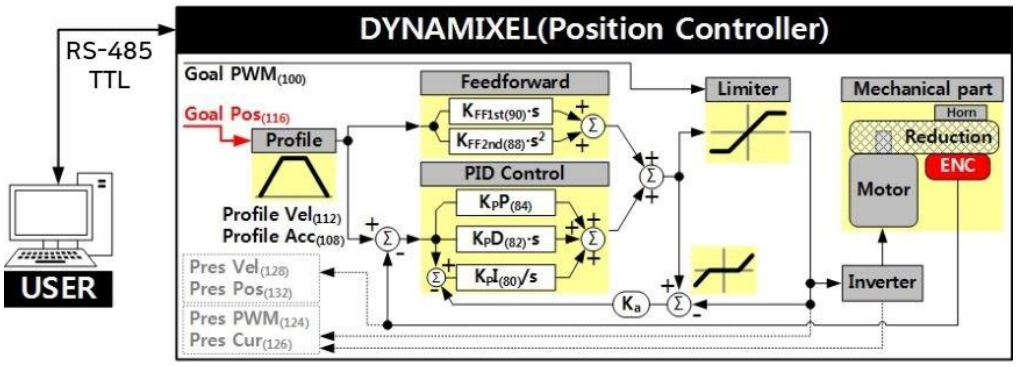


Figura 23. Esquema de control de posición del servo motor

Tras ajustar los parámetros, con un controlador PD ya es suficiente llegar a obtener un error de posición nulo, no se implementó la acción integral porque en robótica, el integrador puede ralentizar el sistema, a parte, los errores acumulados en el integrador pueden causar la oscilaciones y vibraciones no deseados del sistema.

Por último, los parámetros del controlador son los siguiente,

Kp=1700 y Kd=100

Se puede observar mediante los parámetros del controlador ajustado, se ha conseguido eliminar el error de posición, y el tiempo de establecimiento es menos de 0,5 s

Figura 24. Comportamiento del servo motor con PD

Tambien se ha ensayado con distintos perfiles de velocidad y ninguno de ellos presenta error de posicion.

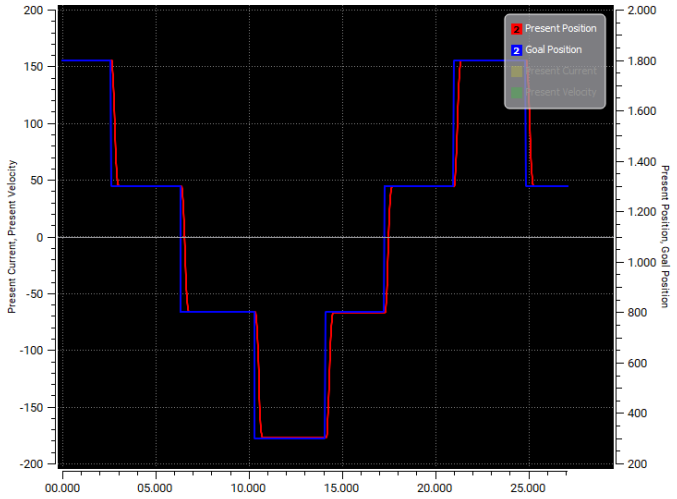


Figura 25. Comportamiento del servo motor con perfil de velocidad 100 unidades

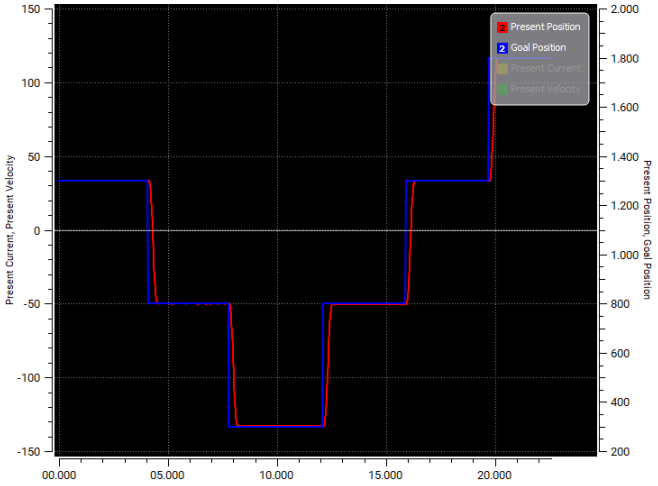


Figura 26. Comportamiento del servo motor con perfil de velocidad 200 unidades

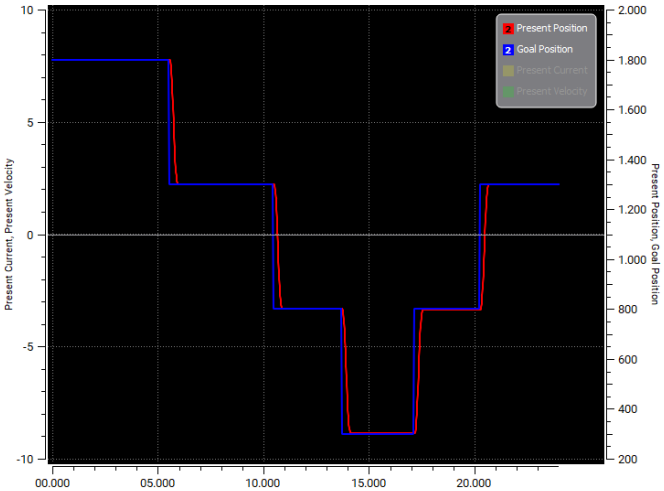


Figura 27. Comportamiento del servo motor con perfil de velocidad 1000 unidades

En cuando las perturbaciones, se ha observado el controlador puede rechazar la perturbación.

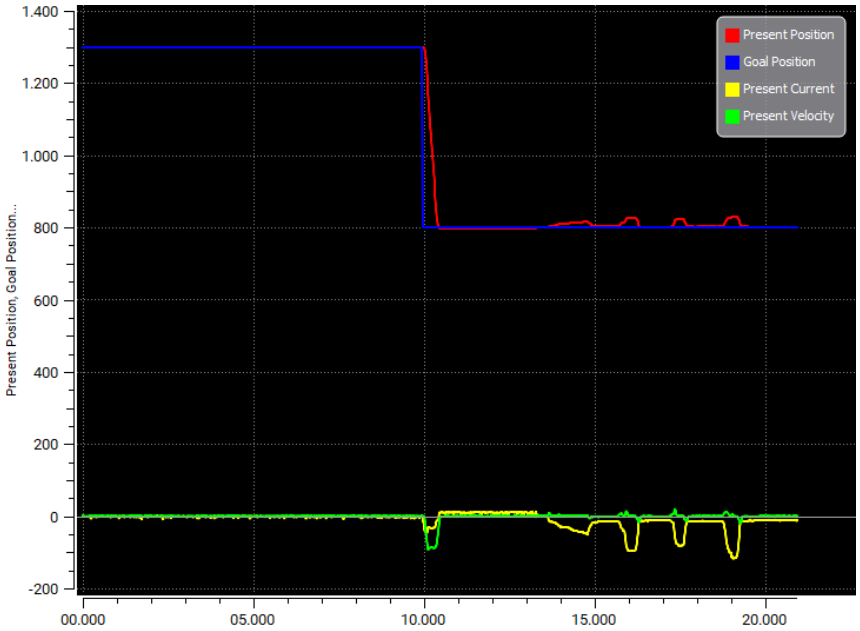


Figura 28. Posición del motor con perturbación

Viendo los resultados obtenidos, se puede concluir que el control PID implementado en su propio servo motor es adecuado para utilizar en el brazo robot.

4.2. Modo control de velocidad

El esquema de control de velocidad es similar al control de posición, pero solo tiene un controlador PI para la velocidad, ya que, en robótica, el control de posición es lo más importante.

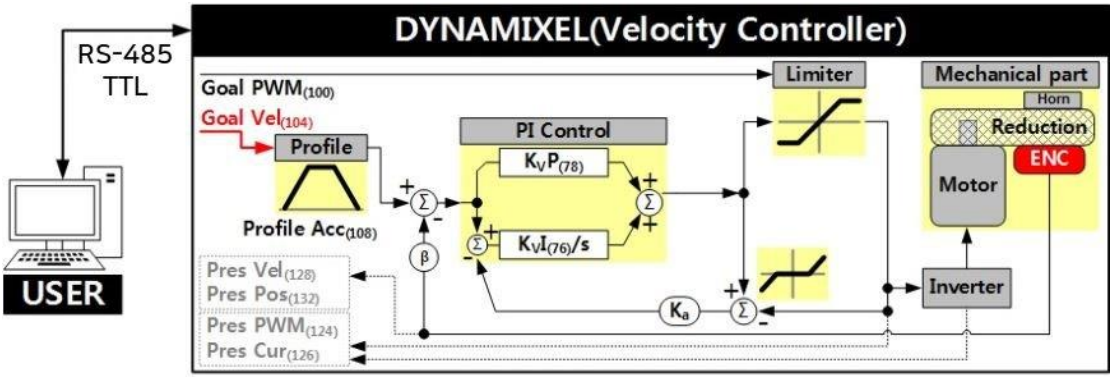


Figura 29. Esquema de control PI de velocidad del servo motor

En cuando los parámetros del controlador de velocidad, los valores de fabricante son Kp=180 y Ki=1600, tras ajustarlo, con los siguientes parámetros tiene mejor comportamiento.

Kp=200 y Ki=2000

Se ha ensayado con dichos parámetros de control y se observa en la siguiente figura, el resultado de la salida seguía perfectamente la señal de referencia.

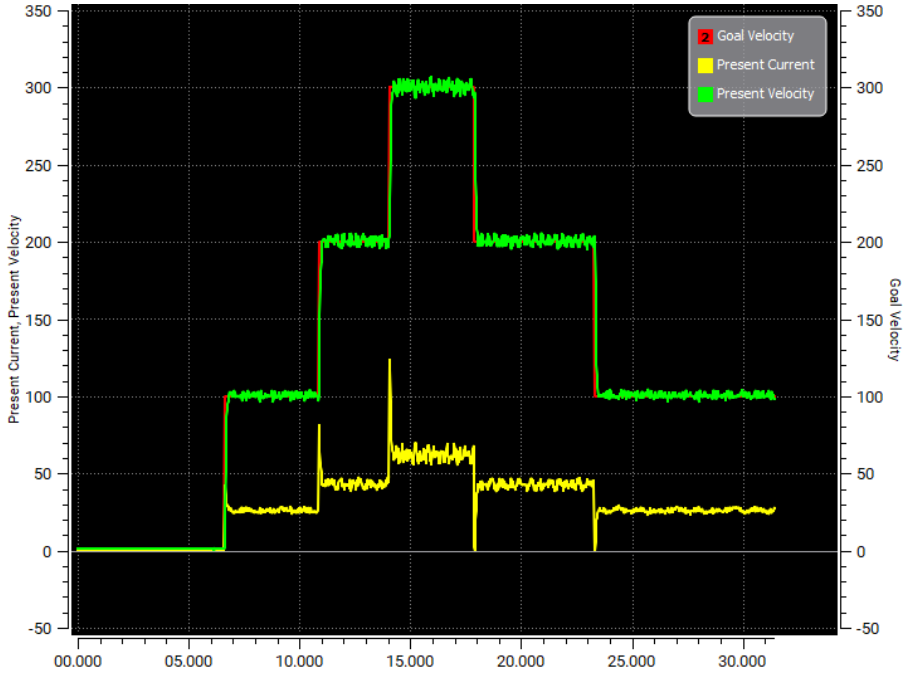


Figura 30. Comportamiento de la velocidad del servo motor

Los ruidos de la señal se pueden visualizar en la siguiente figura, aproximadamente de 5unidades (equivale 1rpm).

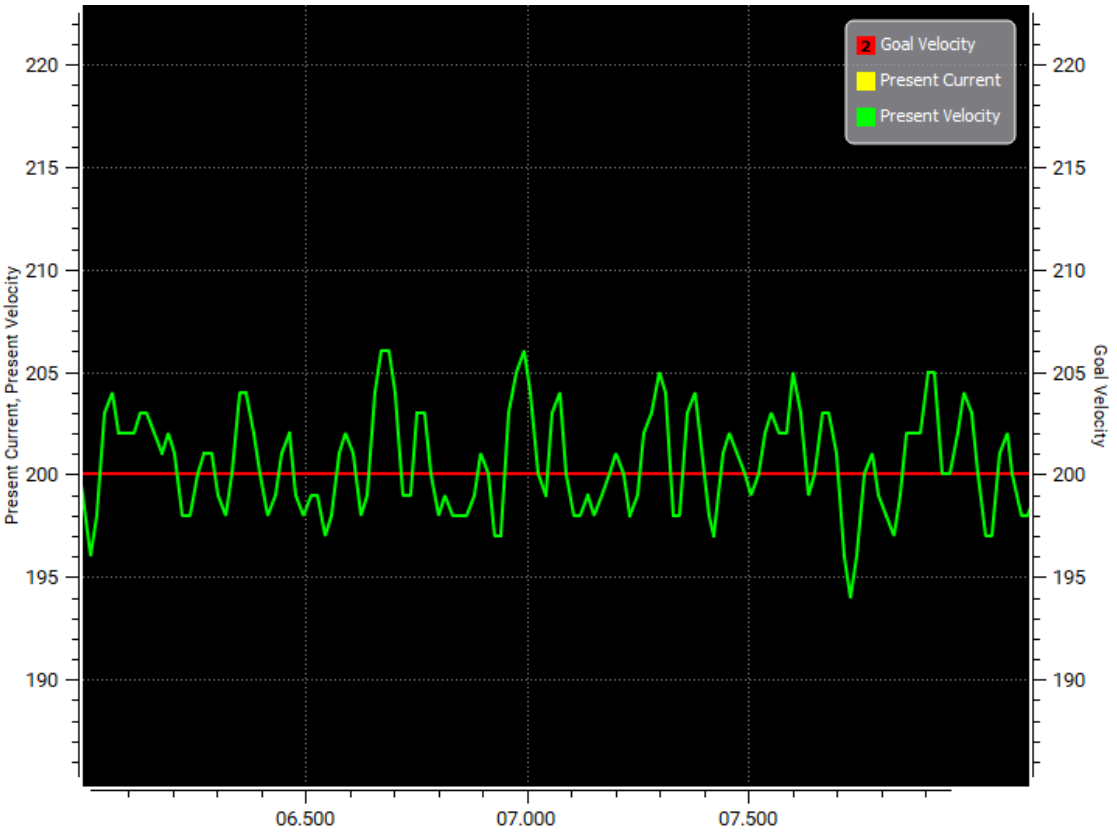


Figura 31. Zoom en la gráfica de la velocidad

En cuando la perturbación, el controlador también rechaza la perturbación, el tiempo de establecimiento es inferior de 0,5 segundo, se puede ver cuando hay una perturbación se disminuye la velocidad y la corriente se aumenta para compensar dicha perturbación, al quitar la perturbación, procede un aumento de velocidad, pero enseguida se la corrige mediante el controlador.

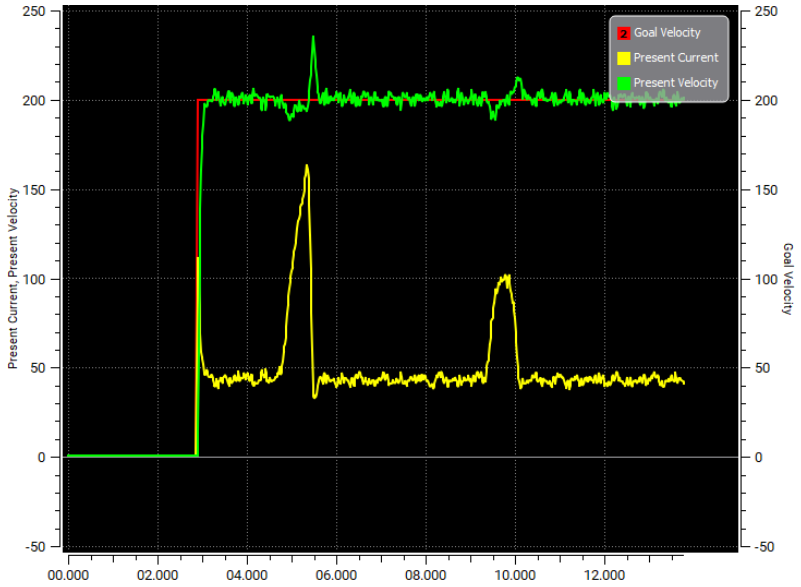


Figura 32. Acción de control en control de velocidad

4.3. Modo control de corriente

El control por modo corriente también se puede denominar control de par del servo motor, ya que la relación entre la corriente y el par es proporcional. Dicho método se utiliza combinando con el control de posición para que el motor pueda generar el par requerido para llegar a la posición deseado.

Se observa en la siguiente figura, la corriente actual seguía la señal de referencia perfecta.

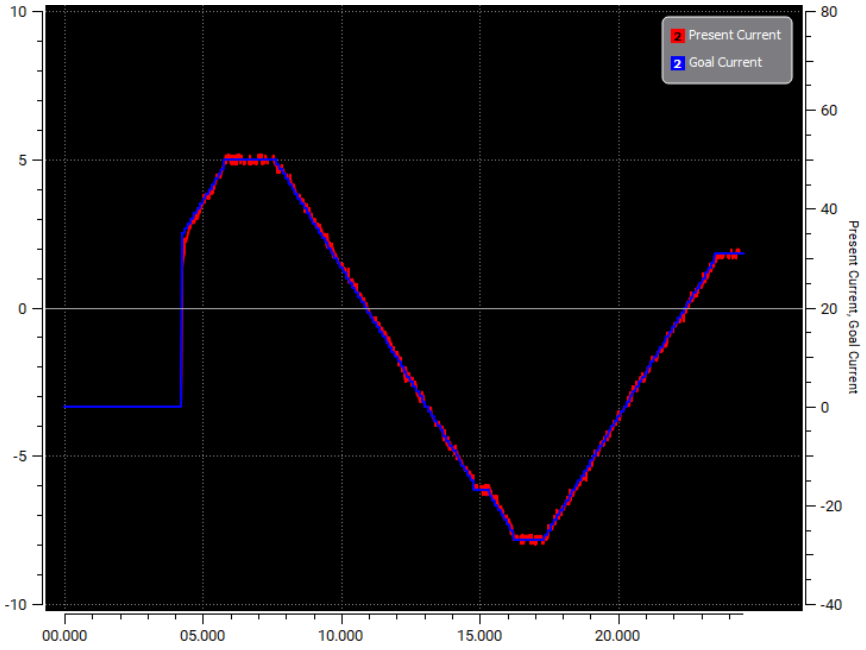


Figura 33. Comportamiento de la corriente

Para ver detalladamente, hay que utilizar la lupa para hacer un ZOOM en la zona interesada, se visualiza los ruidos de la señal que es aproximadamente de 1mA.

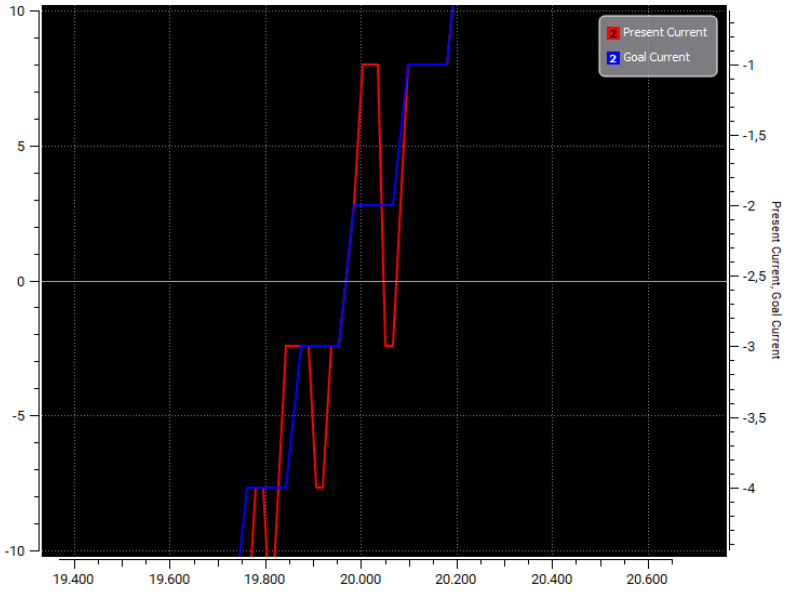


Figura 34. Zoom en la señal de corriente.

También existe una saturación a la hora de aplicar la corriente, en este ensayo vacío, la saturación de la corriente esta alrededor de los 90mA.

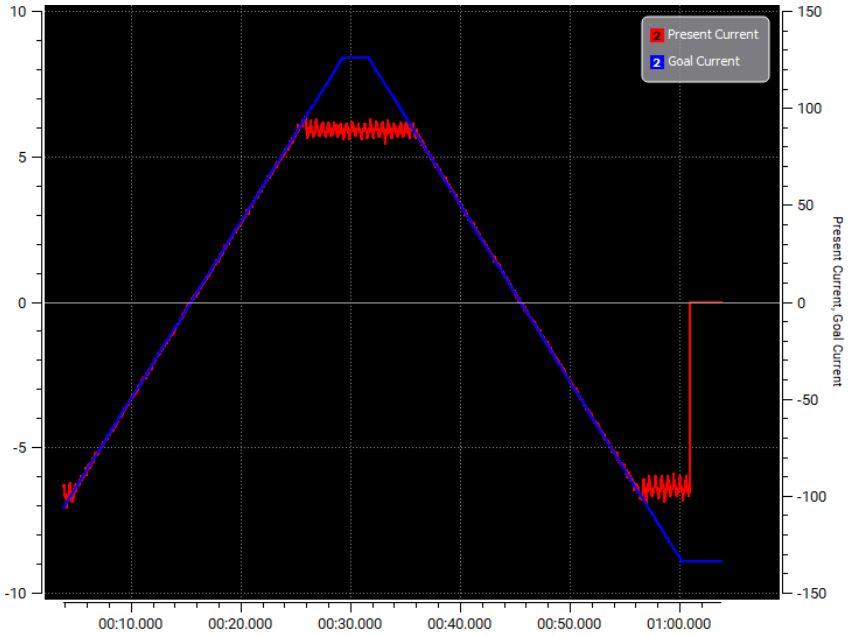


Figura 35. Ensayo vacío de la corriente

5. Cinemática del Robot

La cinemática del robot consiste en estudiar el movimiento de un robot, sin considerar el efecto de las fuerzas externas e internas. En el análisis cinemático del presente proyecto, se estudiará la posición, la velocidad y la aceleración de cada articulación del robot, como lo que ha dicho anteriormente, la realización del estudio de la cinemático solo dependerá de las propiedades geométricas del robot, no de las fuerzas, la relación entre la fuerza y el movimiento del robot se estudiará en el apartado de la dinámica del robot.

Uno de los fenómenos más importantes para el análisis de cinemática es la transformación de coordenadas. A continuación, se mostrará las matrices de rotación y matrices de traslación en el espacio cartesiano que se ha utilizado para realizar el estudio de la cinemática del robot.

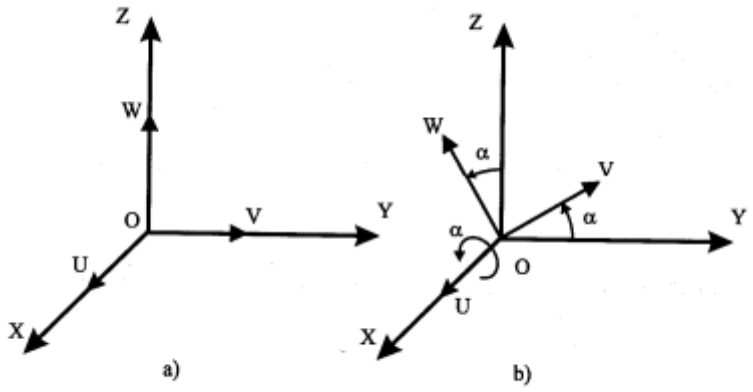


Figura 36. Rotación de sistemas de referencia

Matriz de rotación en el eje X:

$$Rot(x,\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.1)$$

Matriz de rotación en el eje Y:

$$Rot(y,\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.2)$$

Matriz de rotación en el eje Z:

$$Rot(z,\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.3)$$

Matriz de traslación:

$$T(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.4)$$

Siendo a,b,c el desplazamiento en el eje x,y,z respetivamente.

Cuando requiere varias rotaciones, solo hay que multiplicar las matrices de rotación en orden secuencial, por ejemplo, rotación en el eje x,y,z sucesivamente:

$$R_{xyz} = Rot(x,\alpha) \cdot Rot(y,\beta) \cdot Rot(z,\theta) \quad (5.5)$$

5.1. Cinemática Directa

La cinemática directa en robótica es el término que determina la posición y orientación final de un brazo robot en función del ángulo de sus diferentes articulaciones, también depende de la longitud de sus eslabones. Para realizar calcula de la cinemática directa en dicho proyecto, se ha utilizado el método de transformación de matrices, concretamente, mediante el método de Denavit-Hartenberg.

En la siguiente figura está representada el brazo robot con diferentes sistemas de referencia, dichas sistemas de referencia se ha identificado aplicando las reglas de Denavit-Hartenberg, partiendo de ahí se obtiene la tabla de Denavit-Hartenberg.

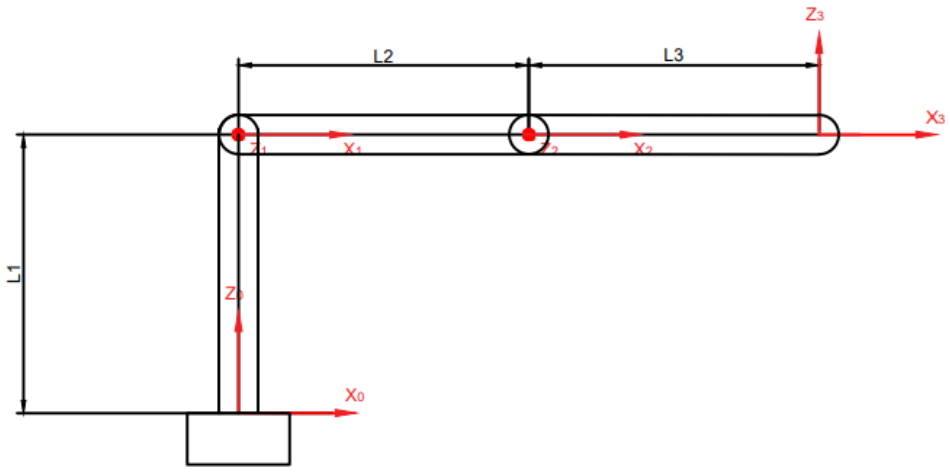


Figura 37. Representación del brazo robot en método de Denavit-Hartenberg

La tabla de D-H es la siguiente.

Articulación	ϑ	d	a	α
1	q1	L1	0	π/2
2	q2	0	L2	0
3	q3	0	L3	-π/2

Tabla 6. Tabla de Denavit-Hartenberg

Partiendo de la tabla anterior, aplicando la matriz de rotación y de traslación, se aplica la siguiente ecuación de DENAVIT-HARTENBERG para obtiene las matrices de transformación homogéneas correspondientes.

$$^{i-1}A_i = R(z,\theta_i)\cdot T(0,0,d_i)\cdot T(a_i,0,0)\cdot R(x,\alpha_i) \tag{5.6}$$

Aplicando la ecuación 5.6 se puede obtener las matrices de transformación de un sistema de referencia a otro sistema de referencia, como los siguientes:

$$^{0}A_1 = \begin{bmatrix} \cos q_1 & 0 & \sin q_1 & 0 \\ \sin q_1 & 0 & -\cos q_1 & 0 \\ 0 & 1 & 0 & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{1}A_2 = \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 & L2\cdot\cos q_2 \\ \sin q_2 & \cos q_2 & 0 & L2\cdot\sin q_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}A_3 = \begin{bmatrix} \cos q_3 & 0 & -\sin q_3 & L3\cdot\cos q_2 \\ \sin q_3 & 0 & \cos q_3 & L3\cdot\sin q_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz homogénea es la siguiente:

$$T = \begin{bmatrix} \cos q_1\cdot\cos(q_2+q_3) & -\cos q_1\cdot\sin(q_2+q_3) & -\sin q_3 & \cos q_1\cdot(L2\cdot\cos q_2 + L3\cdot\cos(q_2+q_3)) \\ \sin q_1\cdot\cos(q_2+q_3) & -\cos q_1 & \cos q_3 & \sin q_1\cdot(L2\cdot\cos q_2 + L3\cdot\cos(q_2+q_3) \\ \sin(q_2+q_3) & 0 & \cos(q_2+q_3) & L1 + L2\cdot\sin q_2 + L3\cdot\sin(q_2+q_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De la anterior matriz, los tres primeros elementos de la última columna es la posición extrema del brazo robot en coordenada cartesiana XYZ.

5.2. Cinemática inversa

La cinemática inversa en robótica es la determinación de los ángulos de cada articulación a partir de los valores de la posición del efecto final del robot. Existe varios métodos de resolver el problema de la cinemática inversa, en este proyecto, se ha resuelto el problema de la cinemática inversa por método geométrico.

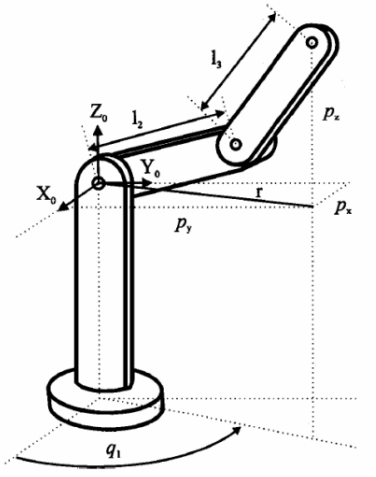


Figura 38. Brazo robot de 3 grado de libertad

La r es a distancia entre punto fina y el inicial desde vista planta, se calcula de siguiente forma:

$$r = \sqrt{{p_x}^2 + {p_y}^2} \tag{5.7}$$

Determinación del ángulo q_1:

$$q_1 = arc\ \tan\frac{p_y}{p_x} \tag{5.8}$$

Determinación del ángulo q_3:

$$h^2 = {l_2}^2 + {l_3}^2 - 2 \cdot l_2 \cdot l_3 \cdot \cos(\pi - q_3) \tag{5.9}$$

Donde la h es:

$$h^2 = r^2 + (p_z - l_1)^2 \tag{5.10}$$

Por otra parte, aplicando la relación trigonométrica, se simplifica la siguiente expresión:

$$\cos(\pi - q_3) = -\cos(q_3) \tag{5.11}$$

Al sustituir se queda de siguiente forma:

$$r^2 + (p_z - l_1)^2 = {l_2}^2 + {l_3}^2 + 2 \cdot l_2 \cdot l_3 \cdot \cos(q_3) \tag{5.12}$$

Por último, se despeja la q_3:

$$q_3 = arc\ \cos(\frac{r^2 + (p_z - l_1)^2 - {l_2}^2 - {l_3}^2}{2 \cdot l_2 \cdot l_3}) \tag{5.13}$$

Para la determinación del ángulo q2, dependerá del q3, según la ecuación 5.13, el resultado obtenido puede ser en codo arriba o codo abajo, como lo que está representado en la siguiente figura.

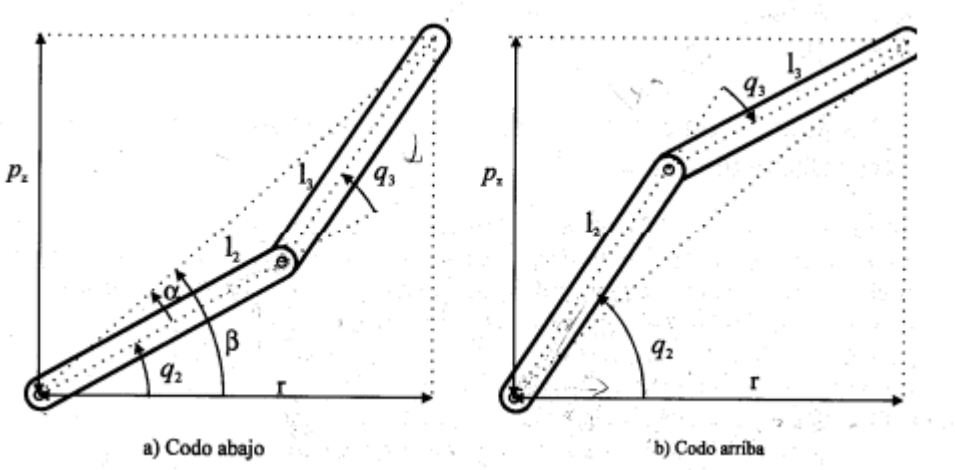


Figura 39. Configuración Codo arriba y codo abajo del brazo robot

En el caso de que quieres en codo abajo, se determina de siguiente forma:

$$\beta = arc\ \tan\frac{p_z - l_1}{r} \tag{5.14}$$

$$\alpha = arc\ \tan\frac{l_3 \cdot \sin(q_3)}{l_2 + l_3 \cdot \cos(q_3)} \tag{5.15}$$

$$q_2 = \beta - \alpha \tag{5.16}$$

Para configuración codo arriba, interpretando de la figura 39 y aplicando los conocimientos de trigonometría y geometría, los nuevos ángulos q2 y q3 tienen siguiente representación:

$$q_3 = -q3 \tag{5.17}$$

$$q_2 = q_2 + 2 \cdot \alpha \tag{5.18}$$

5.3. Cinemática de velocidad

En este apartado, se estudia la cinemática de velocidad del robot, en la cual consiste en determinar la velocidad necesaria de cada articulación para conseguir un control de velocidad de la herramienta final del robot. Dependiendo de la tarea que se realice, el robot necesita ir en diferentes velocidades. Existe diferentes métodos de determinar la velocidad, en este caso se ha utilizado la matriz jacobiana para la determinación, dicha matriz se puede obtener derivando parcialmente la posición del extremo del robot respecto en ángulo que gira cada articulación.

Siendo la posición en el extremo lo siguiente

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos q_1 \cdot (L2 \cdot \cos q_2 + L3 \cdot \cos(q_2 + q_3)) \\ \sin q_1 \cdot (L2 \cdot \cos q_2 + L3 \cdot \cos(q_2 + q_3) \\ L1 + L2 \cdot \sin q_2 + L3 \cdot \sin(q_2 + q_3) \end{pmatrix}$$

El jacobiano de velocidad se determina derivando la posición respecto los ángulos q.

$$J_v = \begin{pmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial z}{\partial q_3} \end{pmatrix} \tag{5.19}$$

Mediante la ecuación 5.17 se puede obtener el jacobiano del brazo robot:

$$\begin{pmatrix} -(\sin q_1 \cdot (L2 \cdot \cos q_2 + L3 \cdot \cos(q_2 + q_3))) & -\cos q_1 \cdot (L2 \cdot \cos q_2 + L3 \cdot \cos(q_2 + q_3)) & -L3 \cdot \cos q_1 \cdot \sin(q_2 + q_3)) \\ \cos q_1 \cdot (L2 \cdot \cos q_2 + L3 \cdot \cos(q_2 + q_3) & -\sin q_1 \cdot (L2 \cdot \sin q_2 + L3 \cdot \sin(q_2 + q_3) & -L3 \cdot \sin q_1 \cdot \sin(q_2 + q_3)) \\ 0 & L2 \cdot \cos q_2 + L3 \cdot \cos(q_2 + q_3) & L3 \cdot \cos(q_2 + q_3) \end{pmatrix}$$

Por otra parte, la matriz jacobiana de la velocidad angular del robot tiene siguiente expresión:

$$J_\omega = (Z_0 \quad Z_1 \quad Z_2) \tag{5.20}$$

Sustituyendo con los valores correspondiente se queda de siguiente manera:

$$J_\omega = \begin{pmatrix} 0 & \sin q_1 & \sin q_1 \\ 0 & -\cos q_1 & -\cos q_1 \\ 1 & 0 & 0 \end{pmatrix}$$

La matriz Jacobiana se puede determinar de distintas formas, también se ha realizado por siguiente método y el resultado obtenido es lo mismo.

$$J = \begin{pmatrix} Z_0 \times t_3 & Z_1 \times (t_3 - t_1) & Z_2 \times (t_3 - t_2) \\ Z_0 & Z_1 & Z_2 \end{pmatrix} \tag{5.21}$$

Donde la Z_i lo cocemos, que son los elementos de tercera columna de la matriz de transformación homogénea T_i , y t_i es los elementos de la cuarta columna (traslacion) de la matriz T_i.

Partiendo de la matriz jacobiana obtenido anteriormente, se puede calcular la velocidad lineal y velocidad angular en coordenadas XYZ.

$$\begin{pmatrix} V_x \\ V_y \\ V_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = J(q)_{6\times3} \cdot \begin{pmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{pmatrix} \tag{5.22}$$

Como se ha comentado antes, en la ecuación 5.20 se ha calculado la velocidad lineal y la velocidad angular mediante la velocidad de cada articulación dada, también se puede hacer de forma inversa, especificando la velocidad lineal y la velocidad angular deseada, y calcular los correspondientes valores de la velocidad de cada articulación.

$$\begin{pmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{pmatrix} = J(q)_{6\times3}{}^{-1} \cdot \begin{pmatrix} V_x \\ V_y \\ V_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{5.23}$$

Pero como la matriz jacobiana del robot no es cuadrática, sino una matriz de 6 x 3, la solución de la ecuación estará restringido, de manera que no se puede conseguir todas las velocidades cartesianas deseada mediante una velocidad articular dado. Por lo tante, en este proyecto se estimará los valores de cada articulación para aproximar la velocidad cartesiana deseada.

5.4. Cinemática de plataforma móvil

En este apartado, se estudiará la cinemática de la plataforma móvil del brazo robo, consiste en relacionar el movimiento de las dos ruedas con el movimiento de la plataforma móvil. En la figura siguiente se puede observar una plataforma móvil con dos ruedas en el plano XY.

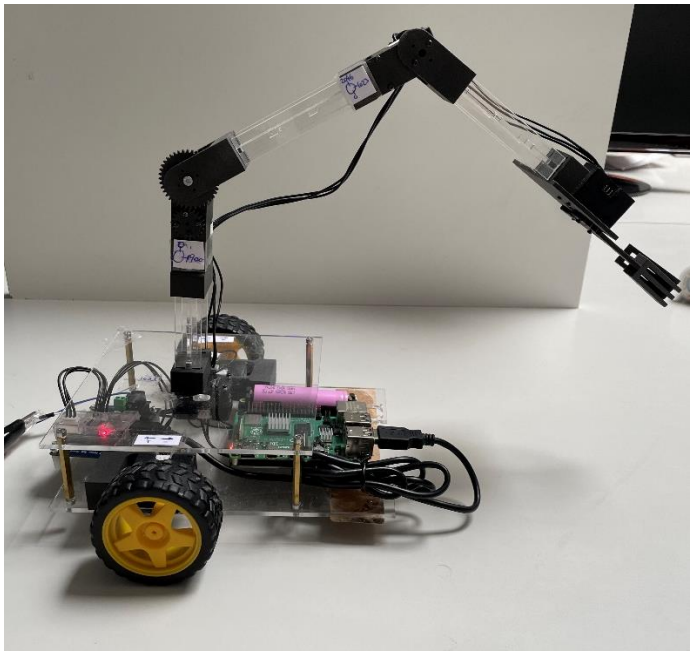


Figura 40. Manipulador móvil montado

El principal objetivo de estudiar el modelo cinemático de la plataforma móvil es para implementar el control del manipulador móvil en el plano 2D. Conociendo las variables como la velocidad de la rueda derecha, la velocidad de la rueda izquierda y la configuración geométrica del robot móvil, se puede calcular todos los valores interesados partiendo de las variables anteriores. A continuación, se explicará brevemente los procesos que se ha realizado para obtener las ecuaciones jacobianas del robot móvil.

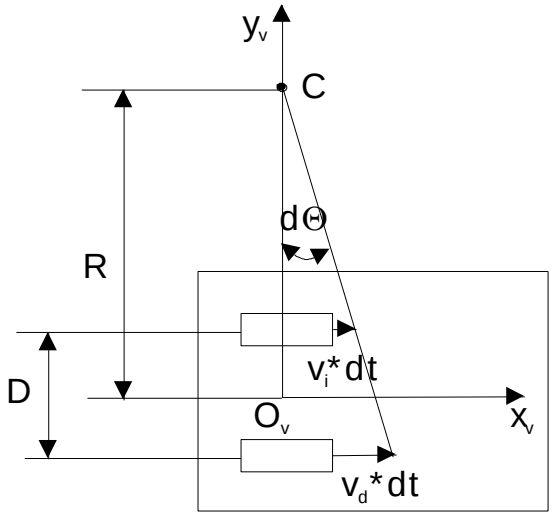


Figura 41. Base móvil con datos geométricos

En primer lugar, tenemos como dato(geométrico) de entrada el radio de las ruedas R_i y R_d, la distancia entre las dos ruedas D y también la velocidad angular de cada una de las ruedas que hemos asignado al principio.

De los anteriores datos se obtiene:

-Velocidad lineal de las ruedas:

$$V_d = R_d \cdot \omega_d \tag{5.24}$$

$$V_i = R_i \cdot \omega_i \tag{5.25}$$

$$V = \frac{V_d + V_i}{2} \tag{5.24}$$

-Velocidad angular de la plataforma móvil:

$$\omega = \frac{V_d - V_i}{D} \tag{5.25}$$

De la anterior expresión se puede deducir si ambas ruedas tienen la misma velocidad, la velocidad angular es 0 y entonces el movimiento que genera es una recta, mientras si es diferente, la plataforma móvil se girara hacia derecha o izquierda dependiendo de la velocidad angular de cada rueda.

El ángulo de gira se puede obtener mediante la siguiente expresión:

$$\theta = \omega \cdot t \tag{5.26}$$

La matriz jacobiana de la plataforma móvil se puede representar de siguiente manera:

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{D} & -\frac{R}{D} \end{bmatrix} \cdot \begin{bmatrix} \omega_d \\ \omega_i \end{bmatrix} \tag{5.27}$$

En cuando espacio de coordenadas, la nueva ubicación se representa de siguiente forma:

$$\begin{bmatrix} x(t+T) \\ y(t+T) \\ \theta(t+T) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} + \begin{bmatrix} V \cdot \cos(\omega t + \theta(t)) \\ V \cdot \sin(\omega t + \theta(t)) \\ \omega \cdot T \end{bmatrix} \tag{5.28}$$

Siendo T=incremento del tiempo.

De las anteriores matrices jacobianos (ecuación 5.17 y 5.27), tanto el jacobiano para el brazo robot y el jacobiano para la base móvil, se puede agrupar todo para obtener un matriz jacobiano total, quedándose de siguiente forma:

$$\begin{pmatrix} \frac{R}{2}.\cos\theta & \frac{R}{2}\cdot\cos\theta & -(\sin q_1\cdot(L2\cdot\cos q_2+L3\cdot\cos(q_2+q_3))) & -\cos q_1\cdot(L2\cdot\cos q_2+L3\cdot\cos(q_2+q_3)) & -L3\cdot\cos q_1\cdot\sin(q_2+q_3)) \\ \frac{R}{2}\cdot\sin\theta & \frac{R}{2}\cdot\sin\theta & \cos q_1\cdot(L2\cdot\cos q_2+L3\cdot\cos(q_2+q_3) & -\sin q_1\cdot(L2\cdot\sin q_2+L3\cdot\sin(q_2+q_3) & -L3\cdot\sin q_1\cdot\sin(q_2+q_3)) \\ 0 & 0 & 0 & L2\cdot\cos q_2+L3\cdot\cos(q_2+q_3) & L3\cdot\cos(q_2+q_3) \\ 0 & 0 & 0 & \sin q_1 & \sin q_1 \\ 0 & 0 & 0 & -\cos q_1 & -\cos q_1 \\ \frac{R}{D} & -\frac{R}{D} & 1 & 0 & 0 \end{pmatrix}\cdot\begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{pmatrix}$$

Se puede observar el tamaño de la matriz jacobiano es de 6x5, es decir, no es una matriz cuadrada, de manera que hay un grado de libertad que no la podemos controlar, además, tampoco se puede obtener la matriz jacobiano-inversa para el cálculo de velocidad reverso. Lo que se puede realizar es quitar una fila menos importante como la fila de la velocidad angular en x o la velocidad angular en y, de esta manera se puede obtener una matriz jacobino-cuadrada de 5x5.

6. Trayectoria

Para que el robot no vaya de forma brusca, es necesaria generar una trayectoria y seguirla, una de las trayectorias más utilizada es la trayectoria trapezoidal, donde el motor acelera bruscamente para alcanzar a la velocidad deseada, una vez llegado la velocidad se mantiene y cuando llega el casi final del tramo, se decelera bruscamente con la misma aceleración hasta pararse.

En la siguiente imagen muestra el perfil de velocidad del motor XL330-M288-T, al mandar el orden de ir a una posición con una velocidad deseada (Profile Velocity), la trayectoria de una articulación se comporta como lo que muestra en la figura.

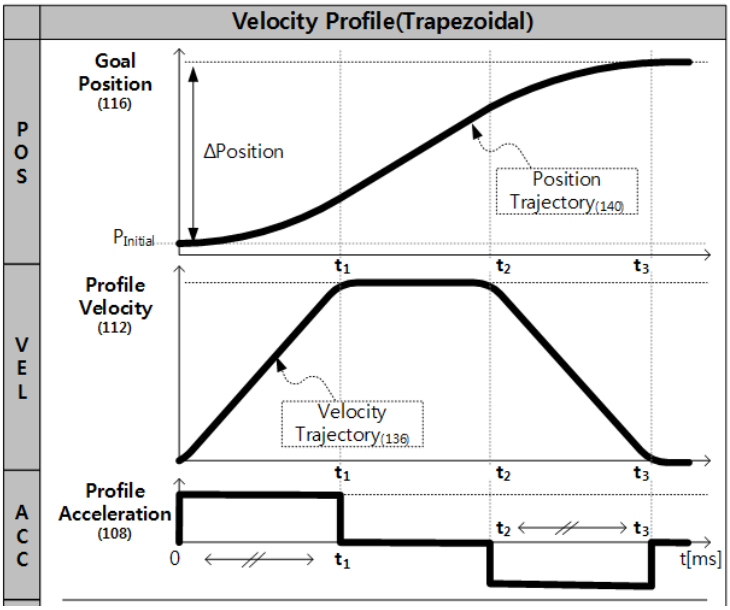


Figura 42. Perfil de velocidad del servo motor

En cuando la trayectoria del brazo robot, el movimiento es de trayectoria cubica(moveJ), se puede utilizar la siguiente función cubica para la generación de la trayectoria.

$$q(t) = a \cdot t^3 + b \cdot t^2 + c \cdot t + d \tag{6.1}$$

$$\dot{q}(t) = 3 \cdot a \cdot t^2 + 2 \cdot b \cdot t + c \tag{6.2}$$

Para despejar las incógnitas a,b,c y d de la ecuación 6.1 y 6.2 , se puede hacer un análisis en el instante inicial del movimiento,

$$t = 0 \rightarrow q_i = d$$

$$t = 0 \rightarrow \dot{q}_i = c$$

Como en el instante inicial, el brazo robot estará en su posición inicial y la velocidad es nulo, entonces se deduce:

$$d = q_i$$

$$c = 0$$

De esta manera, se queda en las dos incógnitas a y b, como sabemos en el instante final, la velocidad tiende al 0 y la posición es q_f, se deduce:

$$\begin{cases} a \cdot t^3 + b \cdot t^2 + q_i = q_f \\ 3 \cdot a \cdot t^2 + 2 \cdot b \cdot t = 0 \end{cases} \rightarrow \begin{matrix} a = \dfrac{-2 \cdot (q_f - q_i)}{t^3} \\ b = \dfrac{3 \cdot (q_f - q_i)}{t^2} \end{matrix} \tag{6.3}$$

En la siguiente figura muestra un ejemplo realizado por Excel que representa la trayectoria cubica con un muestreo de tiempo 10ms.

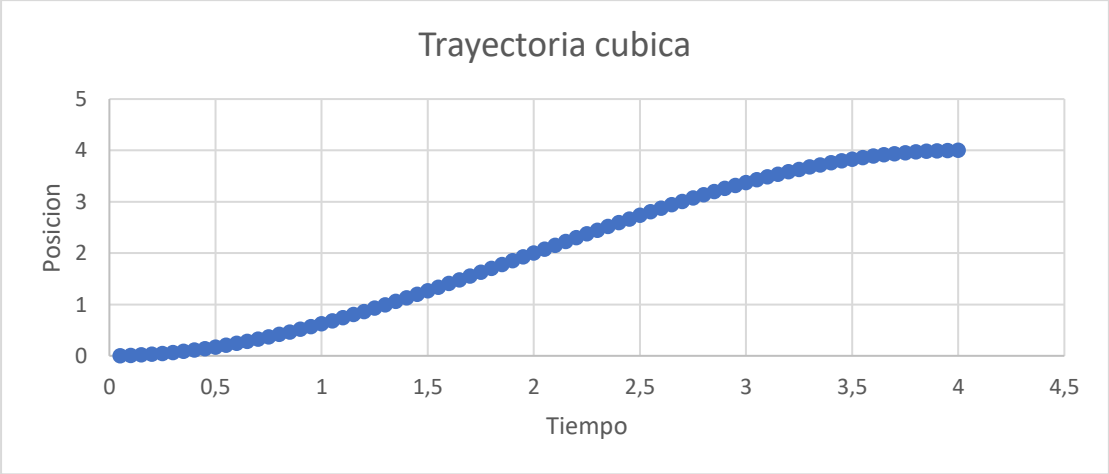


Figura 43. Trayectoria cubica generado mediante Excel

7. Dinámica del Robot

La dinámica del robot es la relación entre el movimiento del brazo robot y las fuerzas que se aplica, se utiliza las ecuaciones matemáticas para expresar el comportamiento dinámico del brazo robot, mediante el conjunto de ecuación, se puede modelar en las computadoras el comportamiento real de un brazo robot. El modelo dinámico real se puede obtener a partir de las leyes fundamentales de física, como las leyes mecánicas newtoniana o la mecánica lagrangiana. Esto lleva al desarrollo de las ecuaciones dinámicas del movimiento para las diversas uniones articuladas del robot, en termino de parámetros geométricos e inerciales.

En este apartado, se pretende obtener las ecuaciones dinámicas del brazo robot diseñado para posteriormente aplicar el control correspondiente con finalidad de obtener una acción de control correcta y eficiente. Se aplica las ecuaciones de Lagrange-Euler para desarrollar el modelo dinámico del brazo robot, este método comparando con el método de Newton-Euler es más sistemático y es conveniente para aplicar en los sistemas mecánicos complejos, como el caso de brazo robot con varias articulaciones.

La ecuación de Lagrange-Euler tiene siguiente expresión:

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}_\iota}\right] - \frac{\partial L}{\partial q_i} = \tau_i \qquad i = 1,2\,,\ldots,n \tag{7.1}$$

Mediante el método de Lagrange-Euler, realizando las conversiones correspondientes de cada articulación, se obtiene la siguiente ecuación que describe la dinámica del robot.

$$\tau = D(q)\cdot \ddot{q}(t) + C(q,\dot{q})\cdot \dot{q} + G(q) \tag{7.2}$$

Donde:

$D(q)$ es la matriz de inercia del robot, dependerá de la masa del cuerpo y también la posición del centro de masa.

$C(q,\dot{q})$ es el termino de Coriolis y centrípeto.

$G(q)$ es el efecto de la gravedad que se aplica en el brazo robot.

Al tratarse de un sistema complejo, con múltiples ecuaciones, se ha utilizado la herramienta software, Wolfram Mathematica para realizar los cálculos necesarios para obtener los parámetros de la ecuación de la dinámica del robot.

En el presente proyecto, solo se explicará brevemente como se ha obtenido la ecuación de la dinámica del robot, para más detalles y especificaciones de los pasos y métodos que se ha utilizado para llegar a obtener la ecuación dinámica, pueden consultar los libros que están citado en la bibliografía.

Lo que se ha realizado primero es obtener la matriz U_{ij} que interpreta el efecto de movimiento del nudo j en todos los puntos del vínculo i.

$$U_{ij} = \begin{cases} 0_{A_{j-1}}.Q_j.{}^{j-1}A_i & for\ j \le i \\ 0 & for\ j > i \end{cases} \tag{7.3}$$

Donde la matriz A es la matriz de transformación y Q una matriz constante que esta adjuntado en el anexo.

El siguiente paso es obtener la matriz de inercia de enlace (o pseudo-inertia matrix) J_i.

$$J_i = \begin{bmatrix} \frac{-I_{xx}+I_{yy}+I_{zz}}{2} & I_{xy} & I_{xz} & m_i.\overline{x_\iota} \\ I_{xy} & \frac{I_{xx}-I_{yy}+I_{zz}}{2} & I_{yz} & m_i.\overline{y_\iota} \\ I_{xz} & I_{yz} & \frac{I_{xx}+I_{yy}-I_{zz}}{2} & m_i.\overline{z_\iota} \\ m_i.\overline{x_\iota} & m_i.\overline{y_\iota} & m_i.\overline{z_\iota} & m_i \end{bmatrix} \quad (7.4)$$

Con los anteriores componentes obtenido, se puede construir la matriz D de siguiente forma:

$$D_{ik} = \sum_{j=\max(i,k)}^{n} Traza\left(U_{jk}\, J_i\, {U_{ji}}^T\right) \qquad i,k = 1,2,\dots,n \quad (7.5)$$

Por otra parte, para obtener la matriz de Coriolis y centrípeto, se ha utilizado símbolo de Christoffel, que tiene la siguiente expresión:

$$C_{ijk} = \frac{1}{2}\left\{\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right\} \qquad i,j,k = 1,2,\dots,n \quad (7.6)$$

$$C_{kj} = \sum_{i=1}^{n} C_{ijk}(q).\, q_i \qquad j,k = 1,2,\dots,n \quad (7.7)$$

Por última parte, para la matriz que describe el efecto de gravedad, es la siguiente:

$$G_i = \sum_{j=1}^{n} (-m_j.\, g.\, U_{ji}.\ {}^{j}\overline{r_j}) \qquad j,k = 1,2,\dots,n \quad (7.8)$$

Al final, la ecuación de la dinámica del robot se queda de siguiente forma, debido a la longitud de la matriz M y la matriz C, se ha representado solo la letra en la siguiente ecuación, los valores esta adjuntado en el anexo.

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = M \cdot \begin{bmatrix} \ddot{\vartheta}_1 \\ \ddot{\vartheta}_2 \\ \ddot{\vartheta}_3 \end{bmatrix} + C +$$

$$+ \begin{bmatrix} 0 \\ -m_2 \cdot \left(-\frac{1}{2} \cdot g \cdot L_2 \cdot \cos\vartheta_2\right) - m3 \cdot \left(-\frac{1}{2} \cdot g \cdot (2 \cdot L_2 \cdot \cos\vartheta_2 + L_3 \cdot \cos(\vartheta_2 + \vartheta_3))\right) \\ -m3 \cdot \left(-\frac{1}{2} \cdot g \cdot L_3 \cdot \cos(\vartheta_2 + \vartheta_3)\right) \end{bmatrix}$$

8. Control de las articulaciones

Con la configuración inicial realizado anteriormente, el servo llega a la posición de referencia sin error de posición, tras realizar el montaje del brazo robot completo, a la hora de simularlo, presenta un error de posición en las articulaciones debido a la fuerza gravitatoria. Entonces, en este apartado se va a diseñar el controlador basando en el controlador PD anterior, pero añadiendo la compensación de la gravedad, el esquema de control de PD+ compensación de gravedad es como el siguiente.

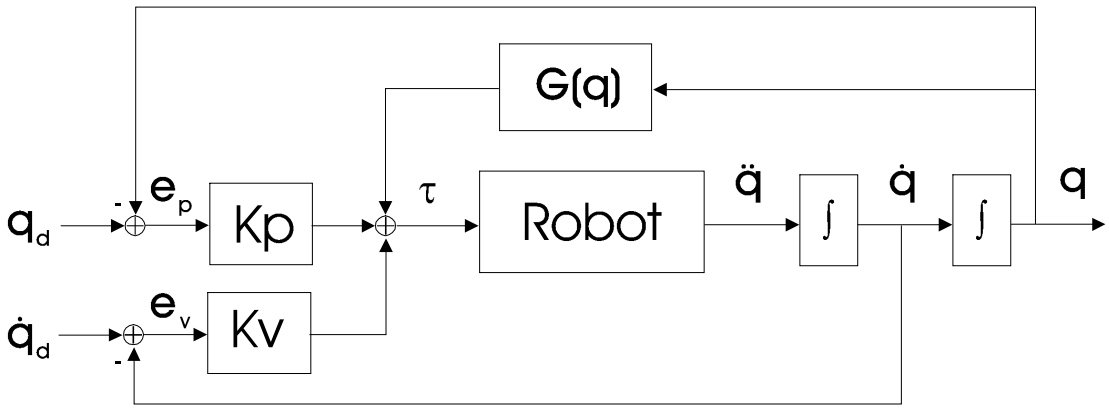


Figura 44. Control PD + compensación de la gravedad

La idea principal es que el motor genere un par adicional que pueda anular el efecto de la gravedad, el efecto de gravedad es variable y que depende del ángulo q, entonces para diferentes posiciones y orientación del brazo robot, los parámetros del control cambia, a parte, como el par es directamente proporcional a la corriente, se ha utilizado el modo de control "Current Based Position Controller".

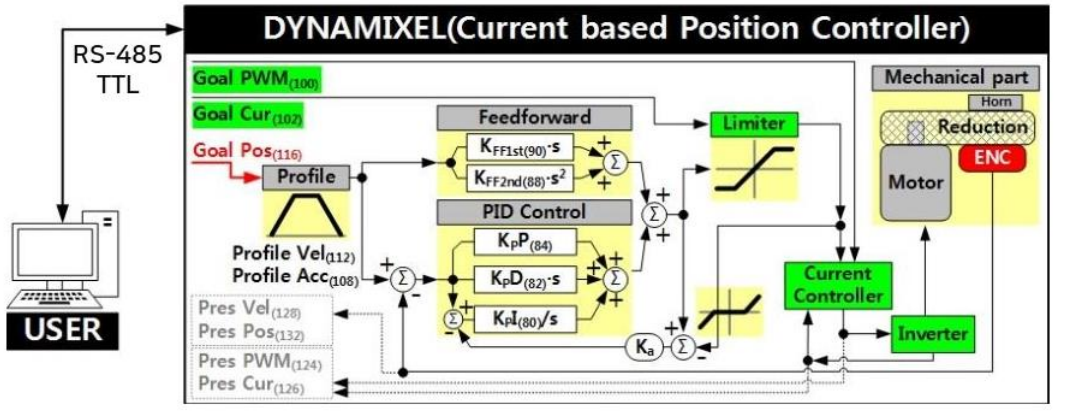


Figura 45. Esquema de control de posición basado en corriente

Los parámetros principales del control que se va a modificar son Goal Current, Kp y Kd. En cuando el brazo robot está elevando (en contra de la fuerza de gravedad), hay que generar más par y el valor de las Goal Current tiene que ser mayor, en el caso contrario, cuando está bajando, el par necesario y el Goal Current necesario es menor.

Según el control de PD más compensación de gravedad, la acción de control se puede deducir de la figura 41, que tiene siguiente expresión:

$$\tau = K_p \cdot (q_d - q) + K_v \cdot (\dot{q_d} - \dot{q}) + G(q) \quad (8.1)$$

τ: Acción de control(par)

K_p: Ganancia proporcional

q_d: Posición de referencia

q: Posición de salida

K_v: Ganancia derivativa

$\dot{q_d}$: Velocidad de referencia

$\dot{q}$: Velocidad de salida

G: Efecto gravedad

De las anteriores variables, la más importante es la G, el efecto de gravedad sobre la articulación, ya que, para realizar movimiento vertical, dependiendo del ángulo q, la acción de control o el par necesarios para realizar el movimiento es mayor o menor. Se puede obtener el efecto de gravedad partiendo de la dinámica del robot que ya está resuelto en el apartado anterior.

Debido a que el servomotor que se ha utilizado para diseñar el manipulador móvil es un tipo de servo de bajo coste, de manera que el fabricante no dispone los métodos de control avanzado, por ejemplo, el controlador PD más compensación de gravedad.

Lo que se planteo es utilizar el método de control de posición basado en corriente, ya que el corriente es directamente proporcional al par. Para ello, es necesario tener es constante de par, en la ficha técnico no se especificó dicho valor, pero se puede estimar mediante el par de parada (Stall torque) y la corriente.

$$K_t = \frac{0{,}52\ Nm}{1{,}47\ A} = 0{,}353\ Nm/A$$

Mediante el constante de par obtenido anteriormente, se puede estimar la corriente necesaria para generar dicho par. Entonces el Goal Current se obtiene de la siguiente expresión:

$$Goal\ Current = \frac{Par\ necesario}{K_t} \quad (8.2)$$

Por último, en el control de posición basado en corriente, el servomotor ira a la posición objetiva generando dicho corriente objetiva como acción de control.

9. Diagrama de flujo

En este apartado, consiste en realizar un diagrama de flujo, es la representación gráfica sobre el algoritmo de control del sistema, para la representación se utiliza los símbolos normalizados por el Instituto Norteamericano de Normalización (ANSI) para realizar el diagrama de flujo, en este proyecto su principal función es para entender mejor el funcionamiento del manipulador móvil.

Los símbolos normalizados que se ha utilizado en el diagrama de flujo son los siguientes:

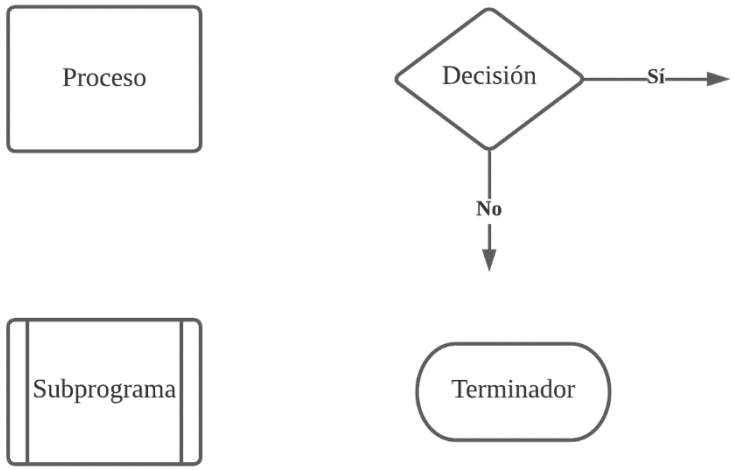


Figura 46. Símbolos de diagrama de flujo

En la figura 47 está representado el diagrama de flujo del funcionamiento principal del manipulador móvil, Lo primero es incluir las librerías necesarias y definir todos los variables que se ha utilizado en el Código. Después se pedirá al usuario que introduzca la coordenada cartesiana del objeto que quiera coger, una vez introducido las coordenadas, el algoritmo calculara la distancia y decide si el brazo robot es alcanzable o no. En el caso afirmativo, se ejecuta directamente el algoritmo de la cinemática inversa para obtener los ángulos que debe girar cada articulación para alcanzar el objeto. En cuando el brazo robot no alcanza el objeto, se ejecutará el algoritmo de la cinemática móvil para acercar al objeto, dicho algoritmo tiene un diagrama de flujo extra que está representado en la figura 48, posteriormente se ejecutara el algoritmo de la cinemática inversa con las nuevas coordenadas que son alcanzable para el brazo robot.

Una vez terminado el cálculo del algoritmo, se mandará las soluciones obtenidas a los servomotores que compone el manipulador móvil para realizar el movimiento de coger el objeto, una vez cogido, se volverá a su posición inicial. Al final se preguntará al usuario si desea salir del programa pulsando ESC o iniciar de nuevo el algoritmo pulsando cualquiera teclado que no sea ESC.

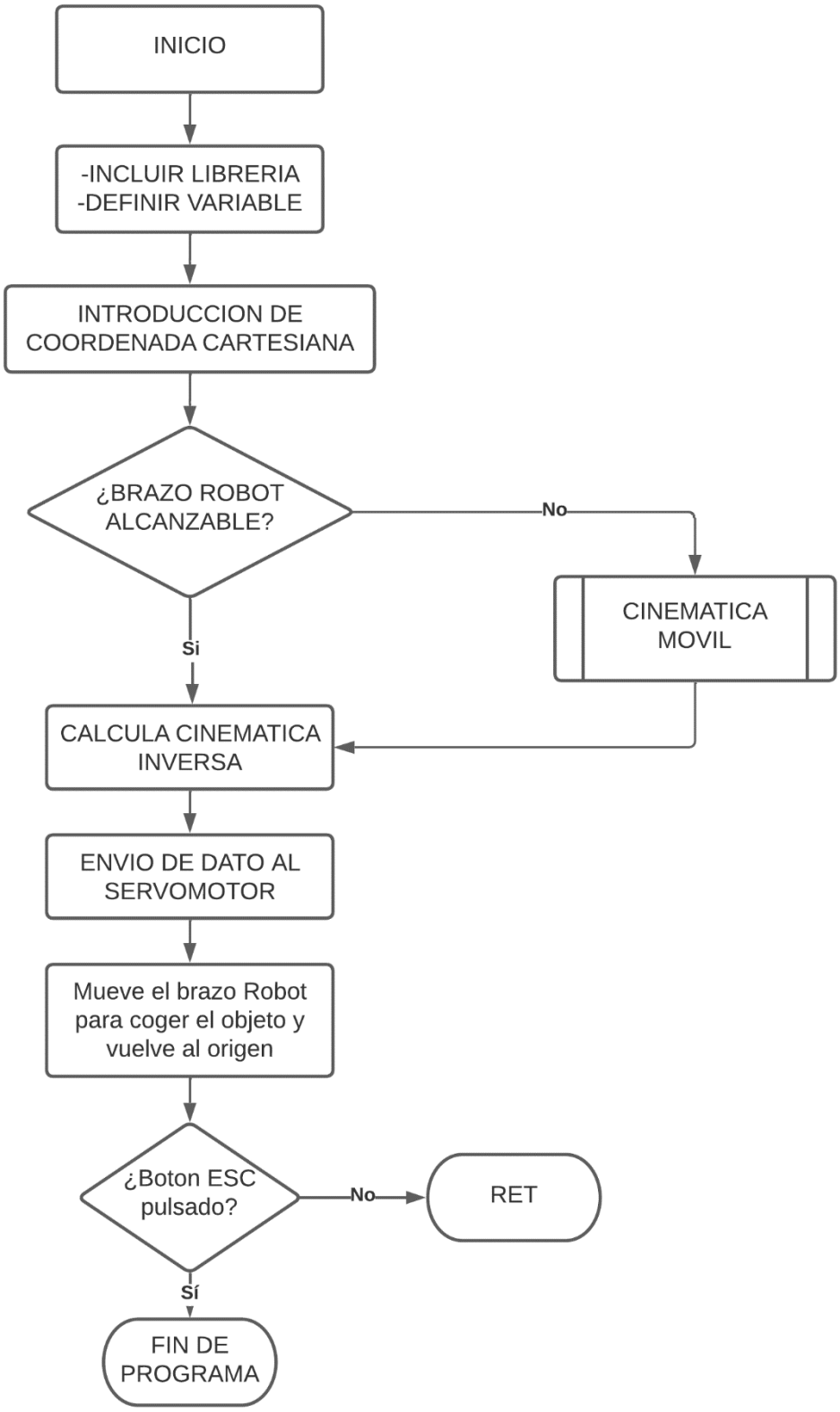


Figura 47. Diagrama de flujo principal del manipulador móvil

El algoritmo de la gestión de la plataforma móvil está representado en el siguiente diagrama de flujo, lo primero es introducir dados de entrada como radio de las ruedas, la distancia entre las dos ruedas, la velocidad angular de las ruedas, etc. Después se generará una trayectoria para el movimiento de la base móvil y entrara a un bucle “for” que hará la comparación en cada muestreo de tiempo entre la posición objetiva y la posición actual, cuando el resultado de la comparación es muy pequeño, el robot se queda en estado “STOP”, en caso contrario se realizara el movimiento para acercar al punto objetivo, como la trayectoria generada no es una trayectoria lineal pura, con la diferencia de la velocidad angular de cada rueda, el manipulador móvil realizara movimiento de gira izquierdo y giro derecho para llegar al punto objetivo. El resultado del algoritmo mandara señal de control al servomotor para realizar el movimiento, así sucesivamente hasta llegar al punto objetivo final.

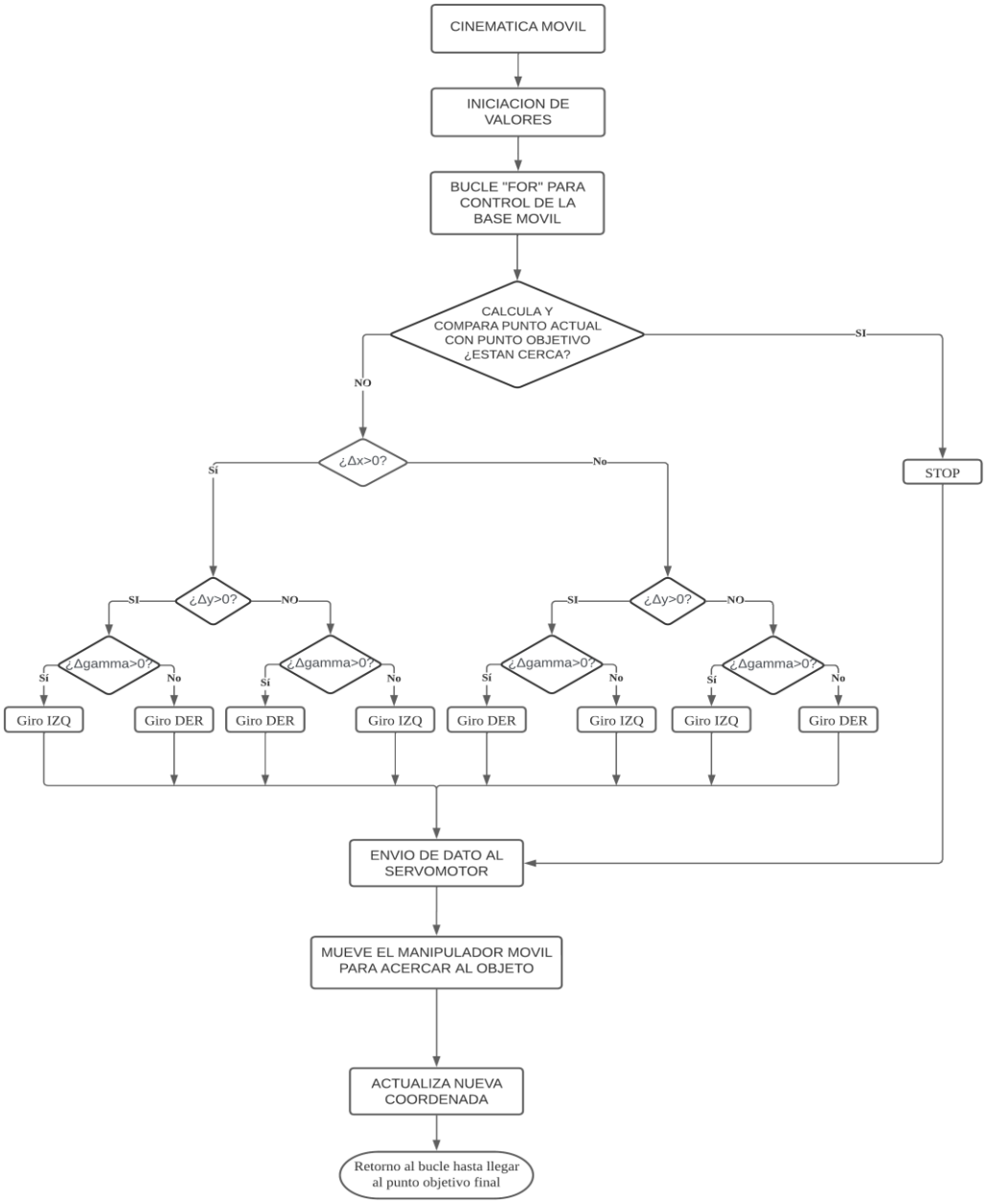


Figura 48. Diagrama de flujo del bloque Cinemática móvil

10. Código en C

En este apartado, se explicará en palabras las principales funciones del algoritmo de control del manipulador móvil. El código fuente esta adjuntado en el anexo.

En primer lugar, se incluirá todas las librerías necesarias, sobre todo la siguiente librería de Dynamixel, dicha librería incluye los diferentes comandos de funciones para controlar el servomotor.

```
#include "dynamixel_sdk.h"
```

Una vez incluido las librerías hay que definir las direcciones de cada variable, ya que, para mandar el dato al servo, hay que escribir en dirección correspondiente. En la siguiente figura muestra algunas definiciones de las variables principales a modificar, como el posición objetiva, corriente objetiva,etc. Tener en cuenta que los diferentes modelos de servo tienen diferentes direcciones.

```
// Control table address
#define TORQUE_ENABLE           64
#define GOAL_POSITION           116
#define PRESENT_POSITION        132
#define GOAL_CURRENT            102
#define GOAL_VELOCITY           104
#define PROFILE_VELOCITY        112
#define PROFILE_ACCELERATION    108
#define Kp      84
#define Kd      80
```

Por otra parte, se define los variables internos que se va a utilizar, y también los variables auxiliares necesarios para realizar el cálculo de la cinemática y la dinámica del robot.

```
//Definir datos geometricos
#define L1    140  //Longitud barra 1 en mm
#define L2    150
#define L3    150
#define R      30  // Radio de la rueda en mm
#define M      50  // Masa del cuerpo, como es simetrico, pesa lo mismo m1=m2=m3, en gramo.
#define pi   3.1415927
//Declarar variable globales
float q1,q2,q3; //angulos que deben calcular para cada articulacion
float r,beta,alfa,X,Y,Z; //Coordenada cartesiana del punto final
int q1f,q2f,q3f,conv; //numeros enteros porque son bit de 0 a 4095

//Variables para calculo de base movil
float vd,vi,wd,wi,w,D,gama,V,dist,xtemp,ytemp;
int t;
```

En el bloque main, la primera función es establecer la conexión por puerto seria entre Raspberry y el driver de servo motor, una vez establecido la conexión, en el programa se puede mandar ordenes de escritura al servomotor mediante siguiente función. Los datos pueden ser de 1 byte,2byte o 4 byte dependiente de la dirección de la variable.

De la siguiente expresión, las variables importantes es el ID, que corresponde al número del motor, la dirección, que está definido al principio del todo y el valor deseado, hay que tener en cuente que el value tiene que ser un numero entero.

```
/*write1ByteTxRx(int port_num, int protocol_version, int id, int address, int value);*/
```

También se ha creado algunas funciones propias para facilitar la implementación del algoritmo de control, por ejemplo, la función de generar trayectoria que te devuelve la trayectoria cubica mediante la introducción del origen, destino, el tiempo de muestreo y el periodo.

```
float gentra(float qi,float qf,float t,float Periodo){
    float b,a,d,c,x;
    b=3*(qf-qi)/(Periodo*Periodo);
    a=(-2*(qf-qi))/(Periodo*Periodo*Periodo);
    d = qi;
    c=0;
    x=a*pow(t,3)+b*pow(t,2)+c*t+d;
    return x;
}
```

Los siguientes códigos son funciones que realiza el movimiento de la base móvil.

```
void giro_izq_dir(){
      //Configuracion inicial motor 5 y 6
    write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, -w_r); //motor derecha va mas rapido
    write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY, w_l);//motor izquierda
}

void giro_der_dir(){
      //Configuracion inicial motor 5 y 6
    write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, -w_l); //motor derecha
    write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY, w_r);//motor izquierda va mas rapido
}

void giro_izq_inv(){
      //Configuracion inicial motor 5 y 6
    write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, w_r); //motor derecha va mas rapido
    write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY, -w_l);//motor izquierda
}

void giro_der_inv(){
      //Configuracion inicial motor 5 y 6
    write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, w_l); //motor derecha
    write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY, -w_r);//motor izquierda va mas rapido
}
```

Los algoritmos de la cinemática inversa y la cinemática móvil están adjuntado en el anexo, con los valores calculado anteriormente, se utiliza la función principal de WriteXbyteTxRx para ejercer el movimiento de cada articulación para llegar al punto objetiva. Y también se mostrará por la pantalla los mensajes informativos y los valores de la variable como velocidad lineal y velocidad angular.

11. Resultado

11.1. Cinemática directa en Matlab

Se ha construido el brazo robot en el entorno de Matlab y también se realizó el ensayado de la función de Pick and Place mediante cinemática directa del brazo robot, consiste en asignar valores a los ángulos q1, q2 y q3 para calcular la posición del efecto final, de manera que pueda mover la pieza que este situado en la mesa interior a la mesa que está situado exterior. La trayectoria que sigue el brazo robot se trata de una trayectoria cubica, esta explicado en apartado posterior. En las dos figuras siguientes muestran el resultado del pick and place.

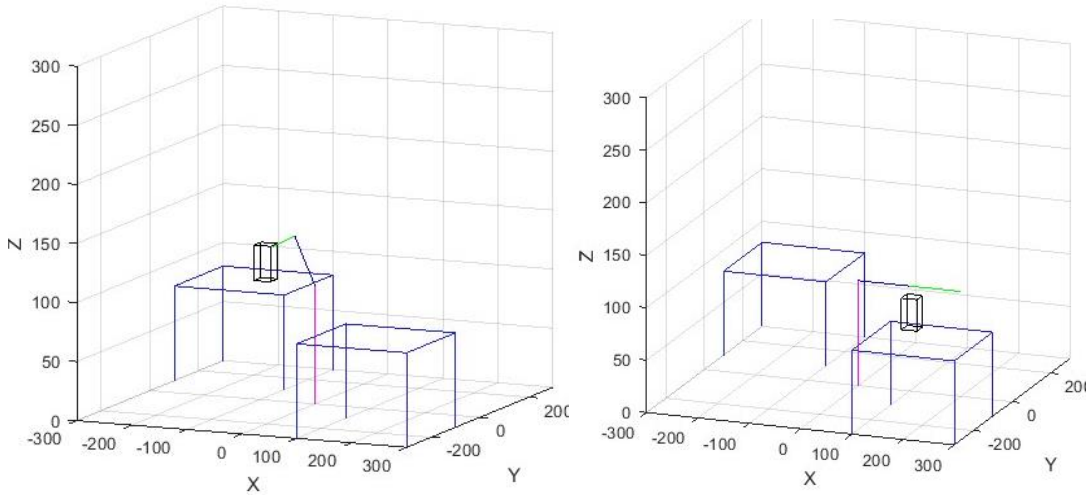


Figura 49. Pick and place de un brazo robot en contorno de Matlab

11.2. Cinemática inversa en Matlab

Como se ha dicho al principio, la cinemática inversa tiene diversos resultados, en este caso se ha obtenido dos resultados, uno es de codo arriba y el otro es de codo abajo, cuando más grado de libertad tenga, más resultado tendrá la cinemática inversa.

Se ha comprobado en el programa Matlab la cinemática inversa, consiste en dar una coordenada en XYZ y el algoritmo calcula los ángulos que deben girar cada articulación para llegar dicho punto. A lo largo del proyecto, se va a utilizar el método de la cinemática inversa con codo arriba para la función de pick and place.

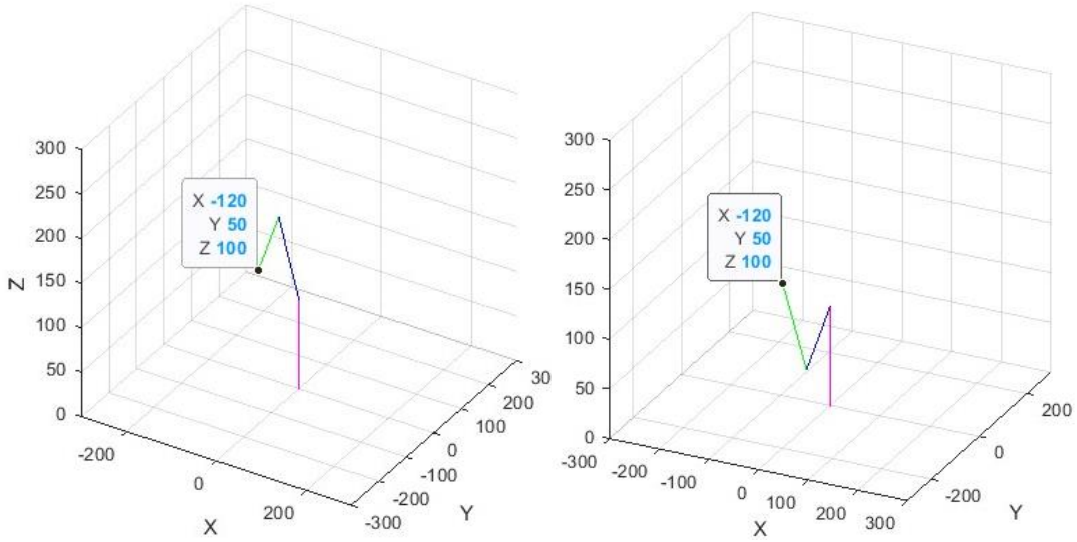


Figura 50. Simulación de la cinemática inversa en Matlab

11.3. Movimiento de la base móvil

Según el estudio en el apartado de la cinemática móvil, se ha ensayado con el movimiento de la base móvil mediante la hoja de cálculo, Excel. Donde se ha generado una trayectoria al inicio, y la plataforma se realiza el movimiento para llegar a dicho puno objetivo.

Durante el movimiento, se puede observar que la base móvil partiendo desde el origen de coordenada e intenta realizar movimiento de giro izquierda para acercar a la trayectoria objetivo. Una vez llegado al punto objetivo, se para el manipulador móvil.

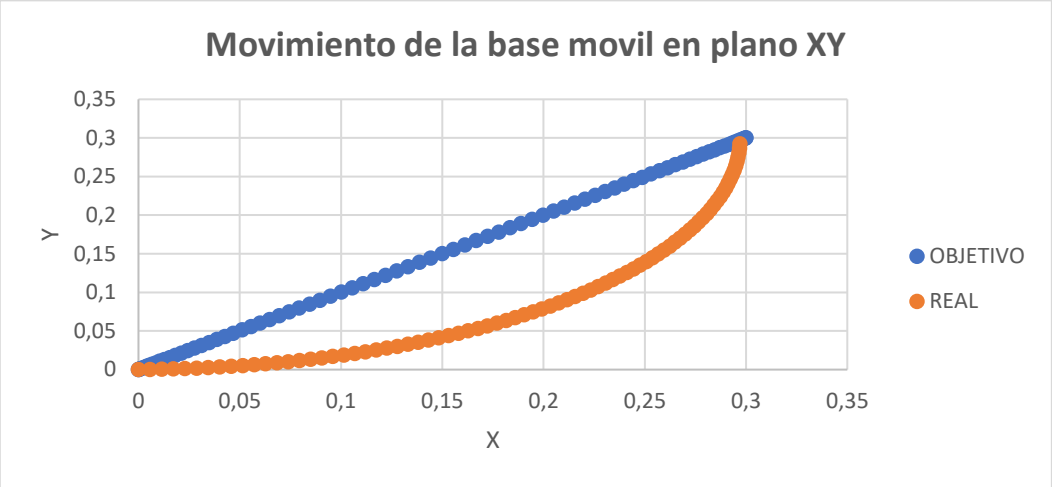


Figura 51.Grafica de la simulación del seguimiento de trayectoria de la base móvil.

En la gráfica siguiente, se ha simulado el movimiento de la base móvil, pero con una velocidad diferente, concretamente una velocidad mayor que el de la gráfica anterior, se observa que el movimiento que se ha realizado la base móvil es partir de giro hacia izquierda, y luego gira hacia derecha para alcanzar el objetivo.

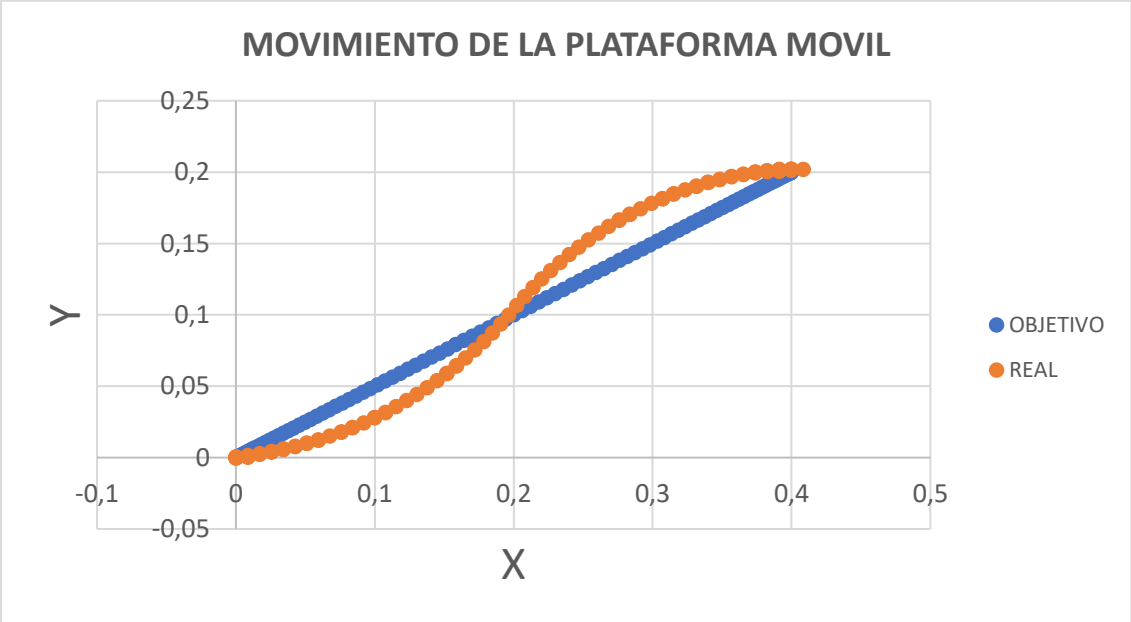


Figura 52. Grafica de la simulación del seguimiento de trayectoria de la base móvil.

Por otra parte, se ha generado el algoritmo de control y se puede observar el resultado obtenido es similar, en cada muestreo de tiempo calcula la posición actual y posición objetivo, realiza la comparación y el resultado de la comparación decide como se va a realizar el movimiento en el siguiente muestreo de tiempo. Además, solo ha tardado 0,3s para recorrer el algoritmo de control de 200 ciclos.

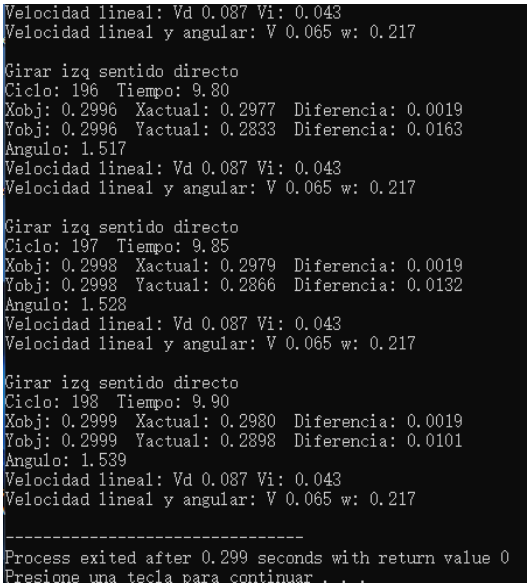


Figura 53. Resultado obtenido con lenguaje C

11.4. Comportamiento de motores

En este apartado, se mostrarán los resultados sobre comportamiento del motor tras el montaje. Cada motor contiene dos gráficas, una gráfica es la comparación de la posición de referencia con la posición de salida y la otra grafica está representada la acción de control y la velocidad del motor.

Motor 1

En las siguientes figuras muestran el comportamiento del motor 1 del brazo robot, dicho motor está situado en la base del brazo robot. En la primera figura, está representado la gráfica de la posición de referencia y la posición de salida, se puede observar el motor sigue la referencia perfectamente, con error de posición nulo y un tiempo de establecimiento alrededor de 1 segundo, además no tiene ningún sobre oscilación.

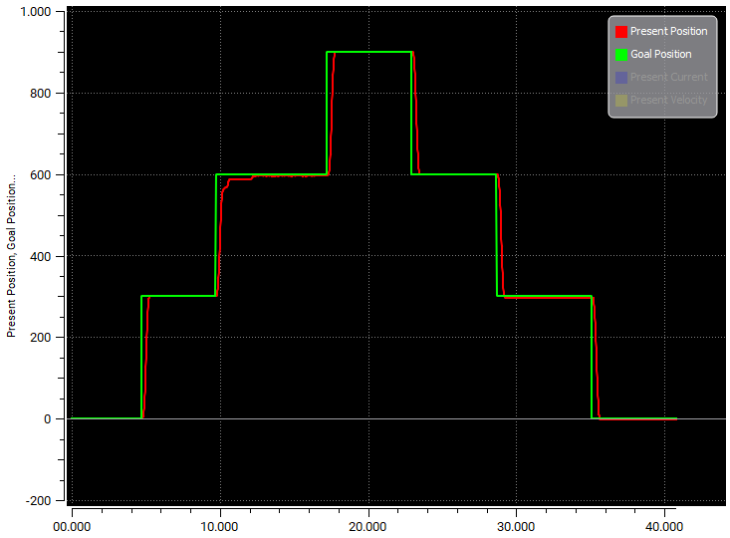


Figura 54. Posición del motor 1

En la figura 55 se puede observar la acción de control de robot, la corriente máxima alcanza a 0.16 A y la velocidad tiene un valor que de 70 unidades (16rpm), mientras que la velocidad de la referencia es de 80 unidades, pero al tratarse de un servomotor de bajo coste, dicho resultado obtenido se puede decir que es satisfecho.

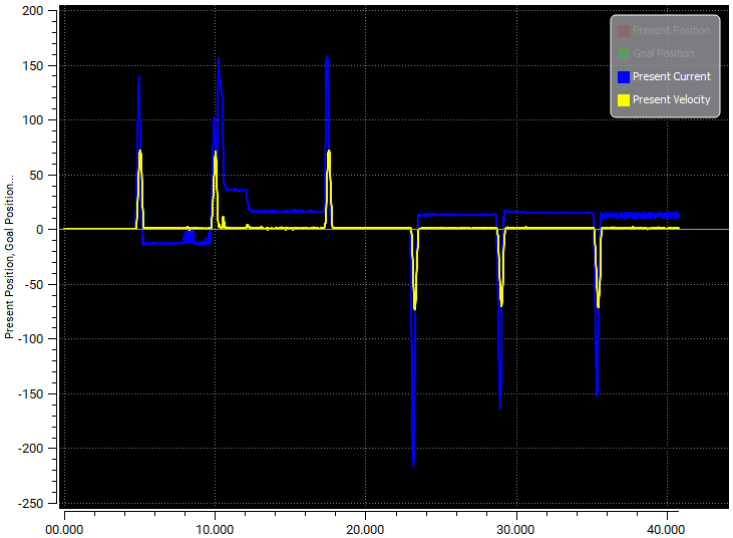


Figura 55. Acción de control del motor 1

Motor 2

El motor 2 es el motor que este acoplado con un engranaje, ya que, según el estudio realizado en el apartado anterior, dicho motor no tiene suficiente par en el caso extremo. Se ha observado que el motor tiene un poco de error de posición, según la observación, es debido al causa del engranaje acoplado, pero el resultado es admisible.

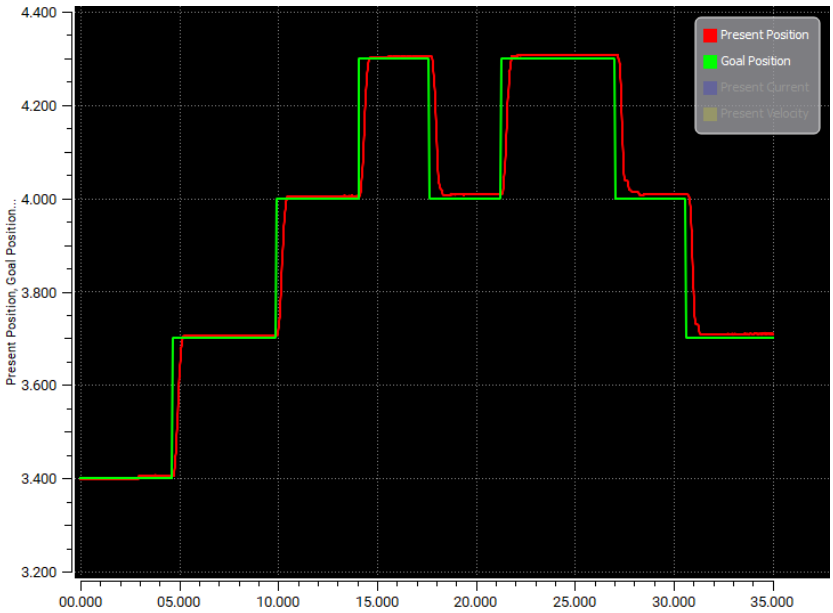


Figura 56. Posición del motor 2

Por otra parte, se puede observar la acción de control es mucho mayor que el caso del motor1 ya que dicho motor es el motor que genera más carga debido a su posición en el brazo robot. En cuando la velocidad, ocurre lo mismo que el motor uno, pero el error no es mucho, entonces no afecta al comportamiento global del manipulador móvil.

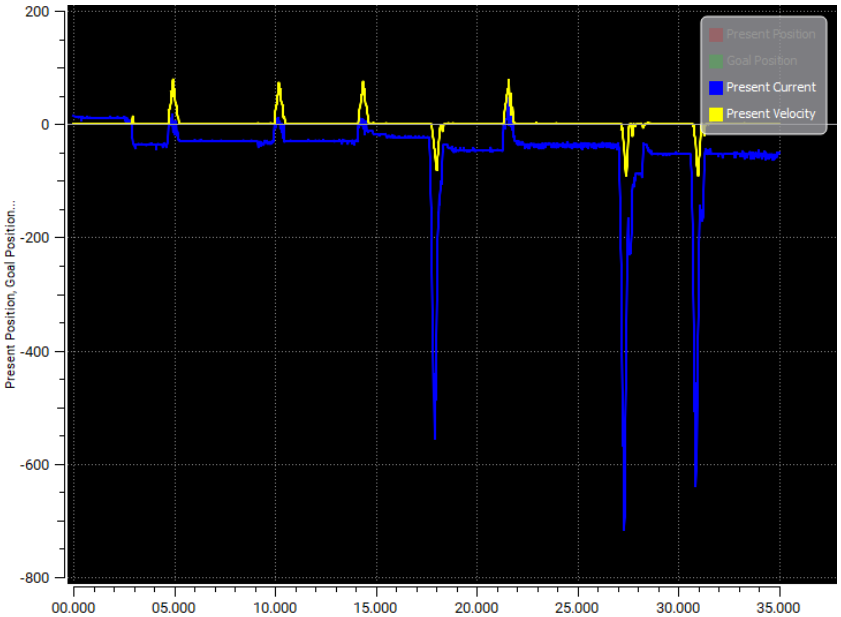


Figura 57. Acción de control del motor 2

Motor 3

El motor 3 es el motor que está situado en la articulación extrema del brazo robot, dicho motor no tiene acoplado ningún engranaje ya que con el par que genera es suficiente levantar o bajar el eslabón sin ningún problema, como se puede observar en la figura siguiente, la posición seguir referencia perfectamente.

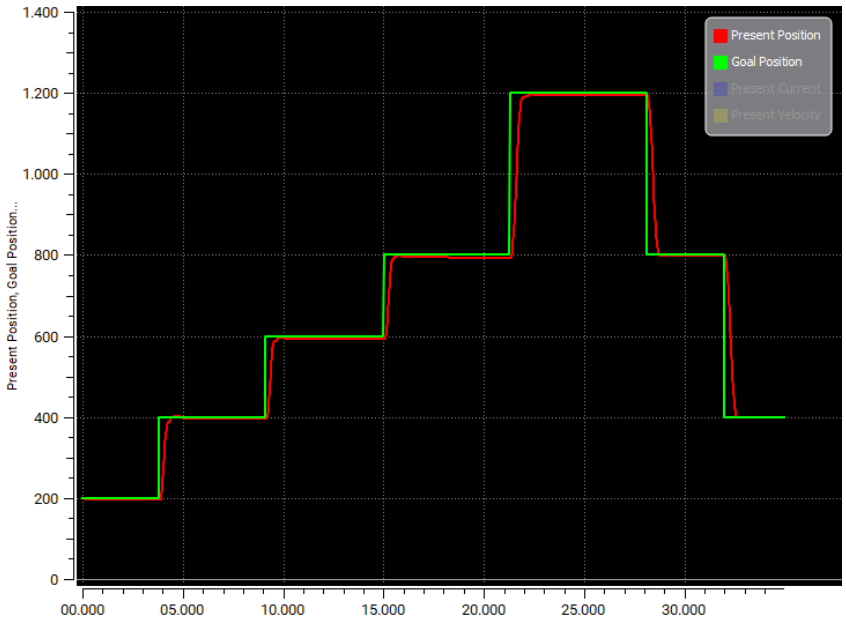


Figura 58. Posición del motor 3

En cuando la acción de control, la corriente alcanzaba hasta 0.18 A y la velocidad esta alrededor de 70 unidades. No se ha observado ningún comportamiento extraño, se puede decir que el resultado es satisfecho para el control.

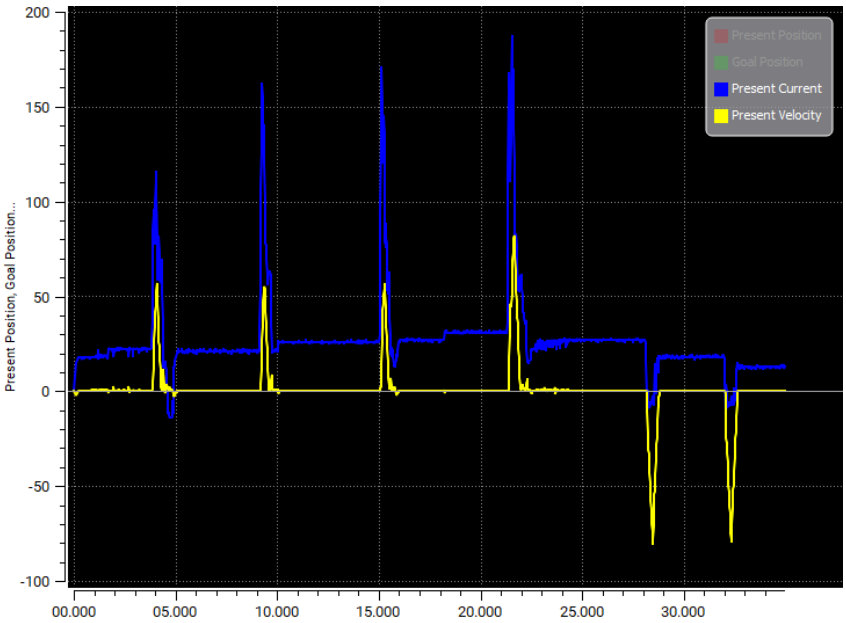


Figura 59. Acción de control del motor 3

Motor 4

Dicho motor es el motor que este acoplado con la pinza del brazo robot, de manera que el tipo de control que se ha implementado para este motor es control por corriente. En la figura siguiente se puede observar mediante una referencia dado, el motor abre la pinza o se cierra la pinza, se puede observar cuando llega al extremo se ha producido un aumento de corriente debido que la pinza se ha chocado con algo, pero enseguida vuelve a la señal de referencia, sin ningún error de posición.

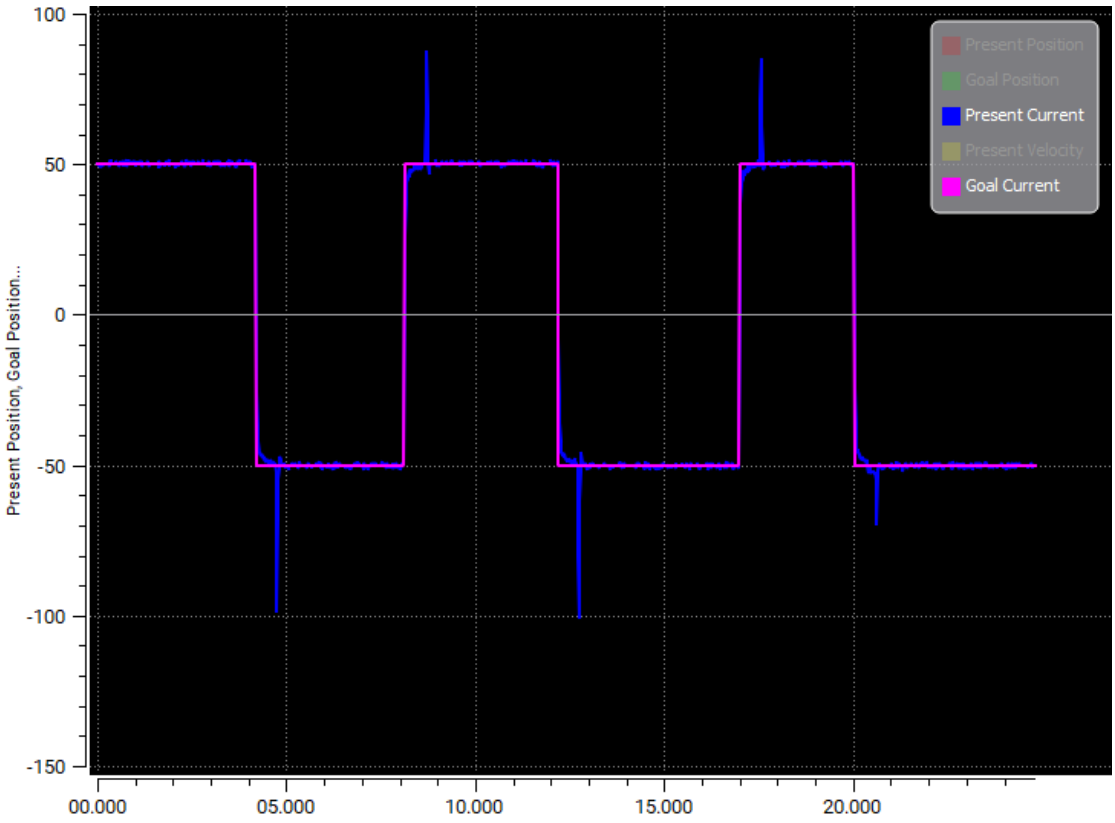


Figura 60. Corriente del motor 4

Motor 5 y motor 6

El motor 5 y el motor 6 son los dos motores que controla la rueda, el tipo de control es control por velocidad, en la figura siguiente, dada una velocidad, el servo motor seguir dicha velocidad perfectamente con un poco de rizado, dichos rizados es debido al ruido del sensor de velocidad y también al rozamiento que existe entre el motor y la rueda.

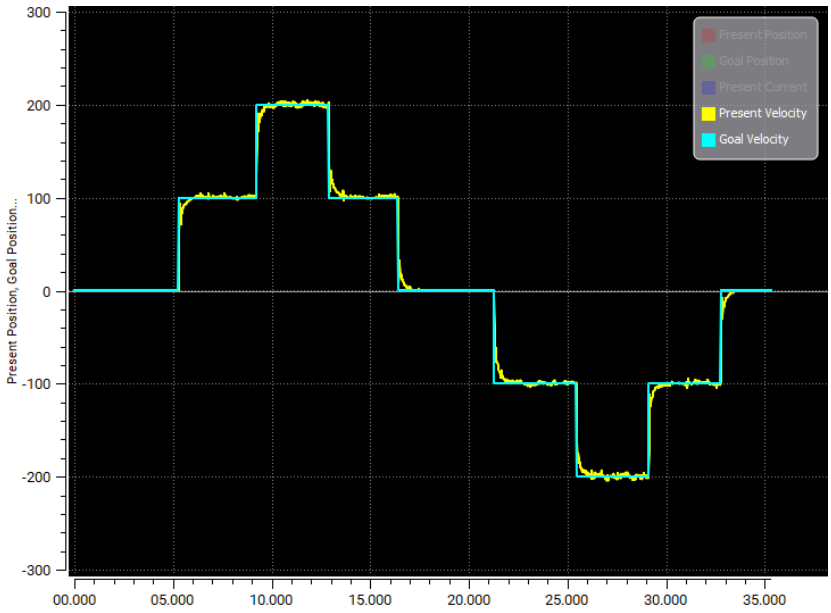


Figura 61. Velocidad del motor 5 y 6

En cuando la acción de control se ha observa que la relación entre la acción de control y la velocidad es proporcional, cuando aumenta la velocidad, la acción de control es mayor, y cuando disminuye la velocidad, la corriente también disminuye.

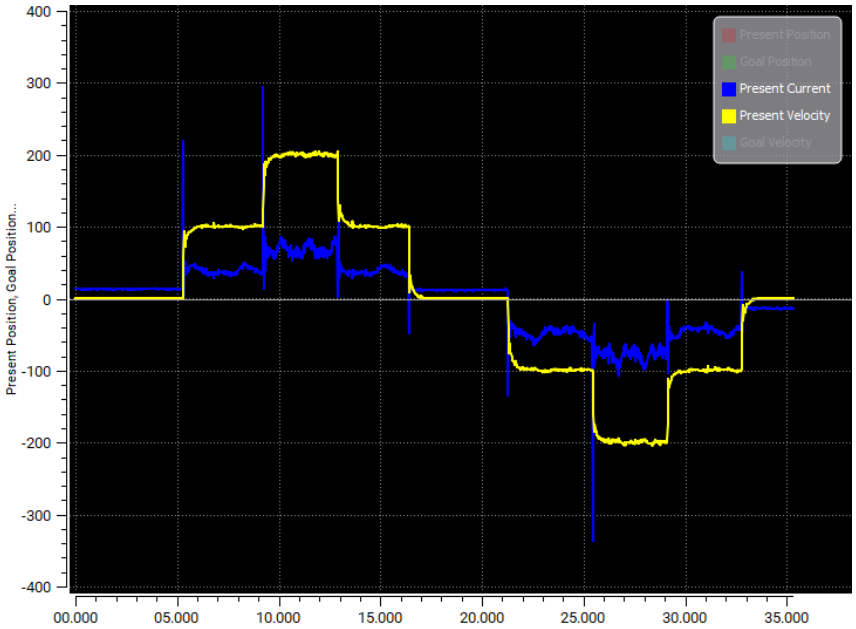


Figura 62. Acción de control del motor 5 y 6

12. Conclusión

Mediante la realización del manipulador móvil, se puede concluir que se ha logrado los propósitos principales del trabajo, que son los siguientes objetivos:

-Se ha diseñado las piezas necesarias del manipulador móvil mediante programa da CAD y posteriormente se imprimió las piezas con la tecnología de impresión 3D. Los montajes y las conexiones realizados

-Se ha seleccionado los componentes necesarios mediante los conocimientos adquirido en la vida académica y se logra buen funcionamiento del manipulador móvil.

-Con las piezas imprimidas y los componentes comprados se realizó el montaje del manipulador móvil.

-El sistema de control diseñado tanto para el brazo robot y para el robot móvil mediante la cinética inversa y la cinemática móvil ha resultado satisfecho en los programas de simulación.

-Se genero los algoritmos de control para implementarlo en la realidad y los resultados obtenidos también satisface con los propósitos principales.

En conclusión, con todos los conocimientos aprendido y con los apoyos de los profesores durante la vida universitaria en la Universidad Politécnica de Valencia se ha logrado diseñar partiendo desde cero hasta llegar tener el manipulador móvil en presente, a parte, el KIT cuenta con todas las funciones deseadas que se ha citado en el objetivo del dicho trabajo. Además, también se ha propuesto algunas mejoras en el siguiente apartado.

13. Propuestas mejoras

Durante la realización del manipulador móvil del dicho trabajo, debido al tema del presupuesto, no se ha implementado, pero sí que se ha sugerido algunas mejoras para optimizar el funcionamiento del robot,

Actualmente, las coordenadas del objeto han sido introducido por el usuario, lo que se ha planteado es implementar la visión artificial para que el robot pueda alcanzar automáticamente al objeto que queremos coger.

Implementar los sensores o módulos como GPS que pueda devolver la posición actual en el tiempo real para cooperar con la visión artificial que se ha citado antes, de manera que el control de la cinemática móvil sea más preciso.

Para familiarizar a los usuarios como gentes mayores o gentes que no domina los dispositivos electrónicos, lo que se propuso es añadir dispositivos de control por voz.

Para alcanzar las propuestas anteriores es necesario que el controlador tengo una capacidad de resolución alta, por una parte, se puede optimizar el algoritmo de control, y por otra parte cambiar por un procesador de alto rendimiento.

14. Bibliografía

[1] Fu, K., Gonzalez, R. & Lee, C., 1987. Robotics: Control, Sensing, Vision and Intelligence. Tercera ed. United States: McGraw Hill

[2] ROBOTIS-GIT/emanual. docs/en/dxl/x/xl330-m288.md

[3] Ficha técnica Raspberry, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[4] Barrientos, Antonio | Peñin, Luis Felipe | Balaguer, Carlos | Aracil, Rafael | Aravaca, Madrid: McGraw-Hill/Interamericana de España | 2ª ed. | D.L. 2007, Fundamentos de robótica

[5] Amirhossein Dadbin; Ahmad Kalhor; Mehdi Tale Masouleh, A comparison study on the dynamic control of OpenMANIPULATOR-X by PD with gravity compensation tuned by oscillation damping based on the phase-trajectory-length concept

[6] Zotovic Stanisic, R., 2022. Apuntes asignatura 32919: Robótica (Máster Mecatrónica). Valencia: Universidad Politécnica de Valencia (UPV)

[7] Martin Mellado, Video tutorial: ROBOTS MOVILES modelado cinemático.

[8] Rubio Montoya, Francisco José, Llopis Albert, Carlos, 2022. Apuntes asignatura 32911: Dinámica de Sistemas Mecánicos (Máster Mecatrónica). Valencia: Universidad Politécnica de Valencia (UPV)

[9] Denia Guzmán, Francisco David, 2021. Apuntes asignatura 32906: Diseño de Máquinas (Máster Mecatrónica). Valencia: Universidad Politécnica de Valencia (UPV)

[10] Kelly, R., Santibáñez, V. & Loría, A., 2005. Control of Robot Manipulators in Joint Space. s.l.:Springer

[11] Wikipedia, Convertidor Buck, https://es.wikipedia.org/wiki/Convertidor_reductor

Presupuesto

El presupuesto para realizar el manipulador móvil hay tres partes, el principal coste es el coste de los materiales, que es un coste directo, por otra parte, son los costes del Software y coste personal.

Coste del material

El coste de los materiales es un coste imprescindible e inevitable, la mayoría de los materiales de la siguiente tabla se ha comprado por internet.

Material	Unidad	Precio (€)	Total (€)
Dynamixel XL330-M288-T	7	35	245
U2D2	1	35	35
U2D2 Power Hub	1	35	35
Raspberry	1	150	150
Batería Raspberry	1	40	40
Rueda	1	6	6
Batería LiPo	1	20	20
Step-Down	1	10	10
Piezas a impresión 3D	20	5	100
Aduana	1	60	60
Fuente de alimentación	1	60	60
		Total	€ 761

Coste Software

Siendo un trabajo académico, los programas que se ha utilizado por el estudiante son los programas que se ofrece por la universidad, de manera que no se ha cotizado ningún coste. Sin embargo, en caso de que no tuviera acceso a dichos programas, el precio para obtener las licencias son los siguientes:

Programa	Unidad	Precio (€)	Total (€)
Matlab	1	262	262
Wolfram Mathematica	1	194	194
AutoCad	1	140	140
SolidWorks	1	947	947
Pack Office	1	50	50
		Total	€ 1593

Coste personal

El coste personal se dedica al coste que se ha facturado para los ingenieros que se ha trabajado y contribuido en el presente trabajo, como el propio alumno, el tutor del TFM y el técnico de laboratorio de impresión 3D.

Descripción	Unidad(h)	Precio(€/h)	Total (€)
Coste ingeniero	200	8	1600
Coste tutoría	5	40	200
Coste técnico laboratorio	10	20	200
		Total	€ 2050

Presupuesto final

Considerando todos los factores anteriores, el coste final para diseñar y montar el KIT de manipulador móvil es de cuatro mil trescientos cincuenta y cuatro euros (4354 €) con IVA incluido.

Descripción	Precio (€)
Coste material	761
Coste software	2971
Coste personal	2000
Total	€ 4354

Sin embargo, el coste real que se ha contribuido por el alumno solo es el coste de los materiales, que son los setecientos sesenta y uno euros (761 €).

Pliego de condiciones

En este documento se detalla todas las informaciones necesarias para realizar el proyecto, todos los intervinientes a la hora de realizar este proyecto deben cumplir los requisitos del dicho apartado.

Condiciones generales

-Al tratarse de un prototipo, hay algunos componentes que este cableado, pero sin aislante, por el tema de seguridad, es recomendable utilizar protección de aislamiento como guantes de goma. La manipulación de las conexiones siempre requiere desconectar la fuente de alimentación.

-En el prototipo existe algunos componentes electrónicos de pequeño tamaño y también los tornillos, se debe aislar de los niños de menores de 8 años para evitar riesgo de tragarse.

-Cualquier modificación tanto como cableado o como la parte del control, se debe estar supervisado por un profesional.

Condiciones técnicas

-Según la ficha técnica de los motores, la tensión de manipulación es recomendable utilizar fuentes que genera 5V, y en ningún caso debe superar a los 6V. Cualquier consecuencia que produce debido al motivo de no respetar la tensión de maniobra, serán responsable los propios usuarios.

-La batería de LiPo debe cargarse exclusivamente con los cargadores de LiPo, cualquier otro cargador puede provocar el mal funcionamiento de la batería incluso en el peor caso, se provocará la explosión de la batería.

-La placa Raspberry Pi deben trabajarse un ambiente fresco, en el caso de que necesita trabajar a largo tiempo es recomendable enfriarse con ventiladores o disipadores.

Anexo

Software

DYNAMIXEL Wizard 2.0

Es un programa que se ha desarrollado por la compañía de DYNAMIXEL para controlar los servomotores de su propia marca. Es compatible con los diferentes sistemas de operativos, como Windows, Linux y Mac. Las características destacado son los siguientes:

-Actualización de firmware de DYNAMIXEL

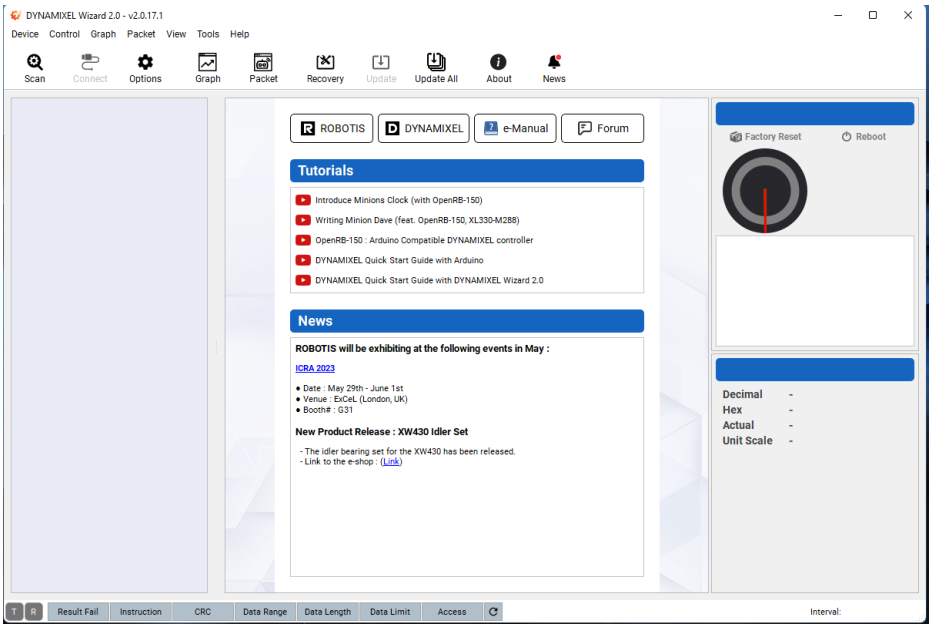
-Diagnóstico DYNAMIXEL

-Configuración y prueba de DYNAMIXEL

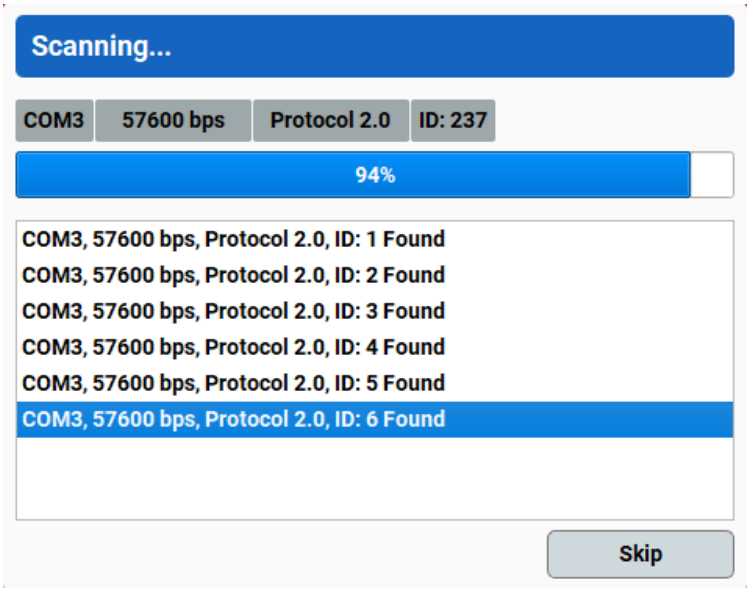
-Trazado de datos DYNAMIXEL en tiempo real

-Generar y monitorear paquetes DYNAMIXEL

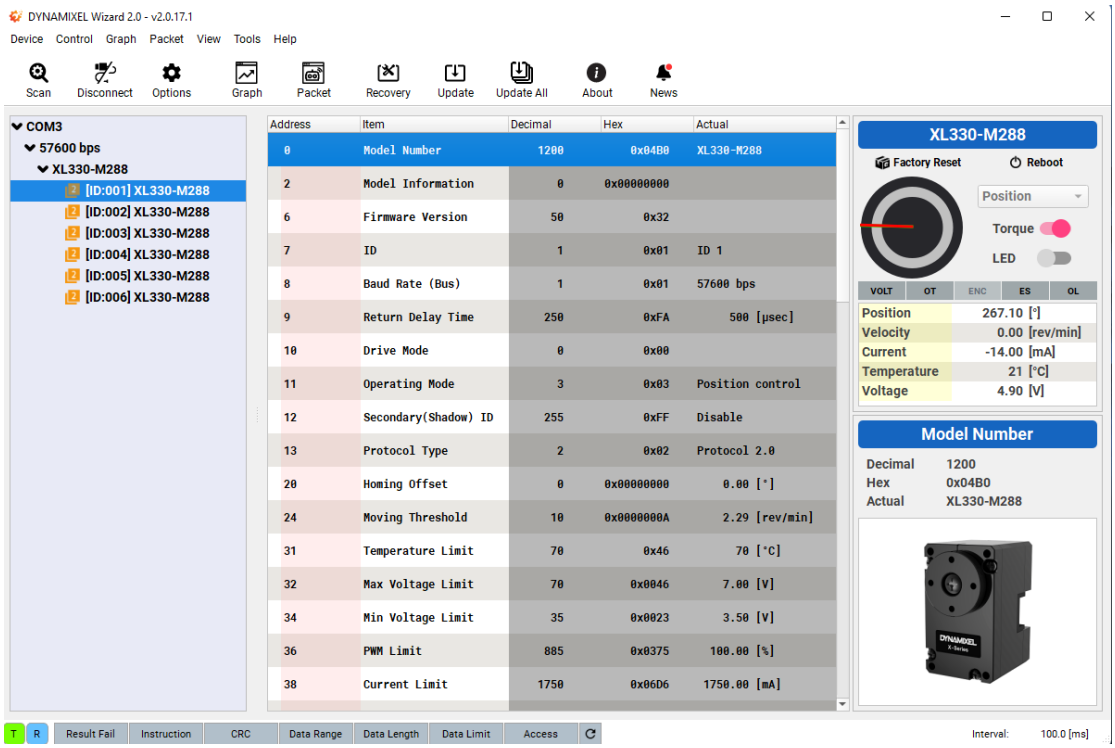
La siguiente figura muestra la pestaña de entrada del programa, para realizar la conexión entre el servo motor y el programa hay que escanear pulsando “scan”, pero antes debe configurar el protocolo de comunicación, el baudio correspondiente y también el numero de puerto seria.



Si la conexión se realizó con éxito, se encontrará todos los motores conectados como lo que muestra en la siguiente figura.

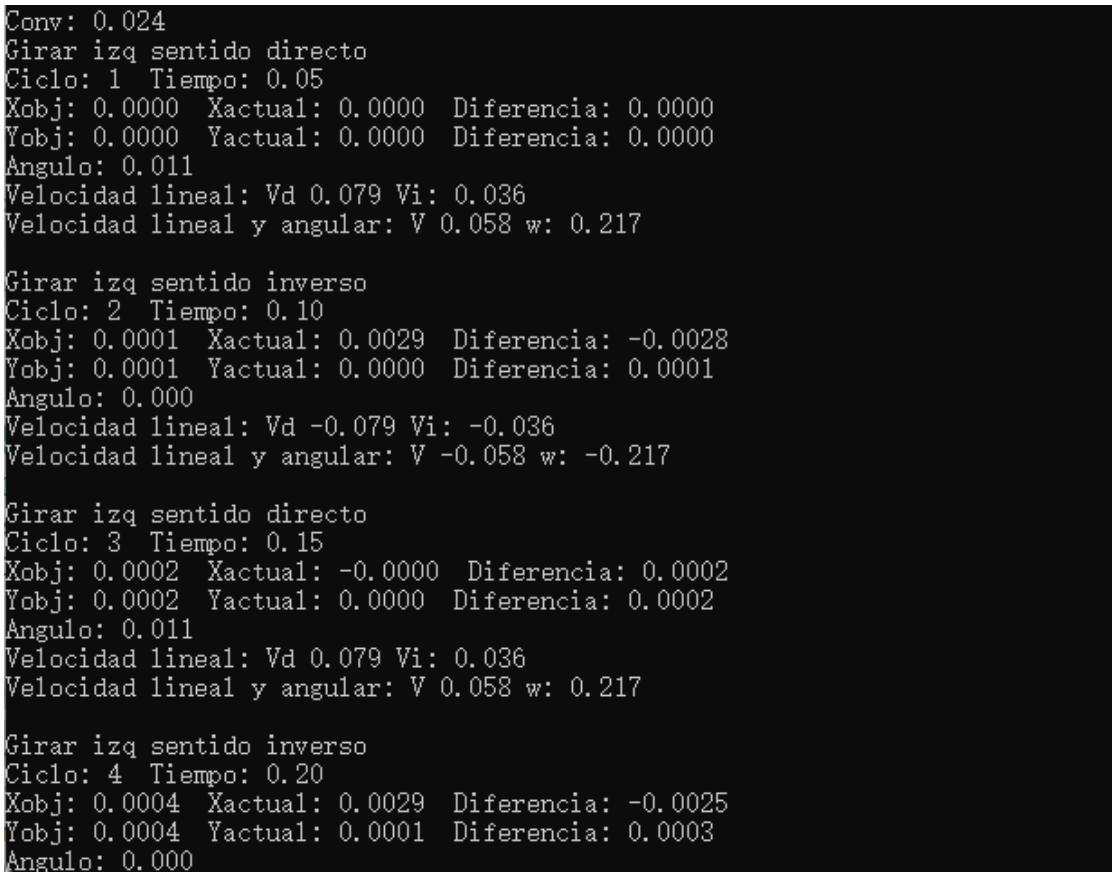


A continuación, se mostrará todos los parámetros del motor, tanto la configuración inicial del motor como los parámetros de control del motor. Además, dispone la gráfica en tiempo real que se puede leer las variables como la posición actual, velocidad actual, corriente actual, etc.



Dev C++

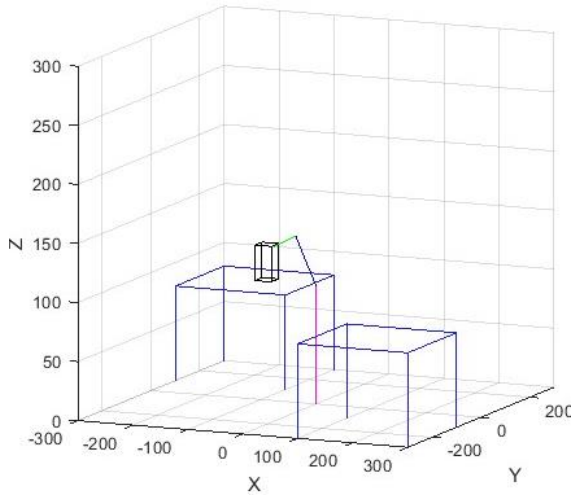
Es un entorno de desarrollo integrado para programar en lenguaje C/C++, Se utilizo para programar los códigos de fuente para controlar el manipulador móvil, principalmente para testear y analizar los algoritmos de control antes de aplicar en el manipulador móvil. Una vez que se ha obtenido los resultados deseados, se implementa al manipulador móvil mediante conexión en serie entre el manipulador y el Raspberry Pi.



Matlab

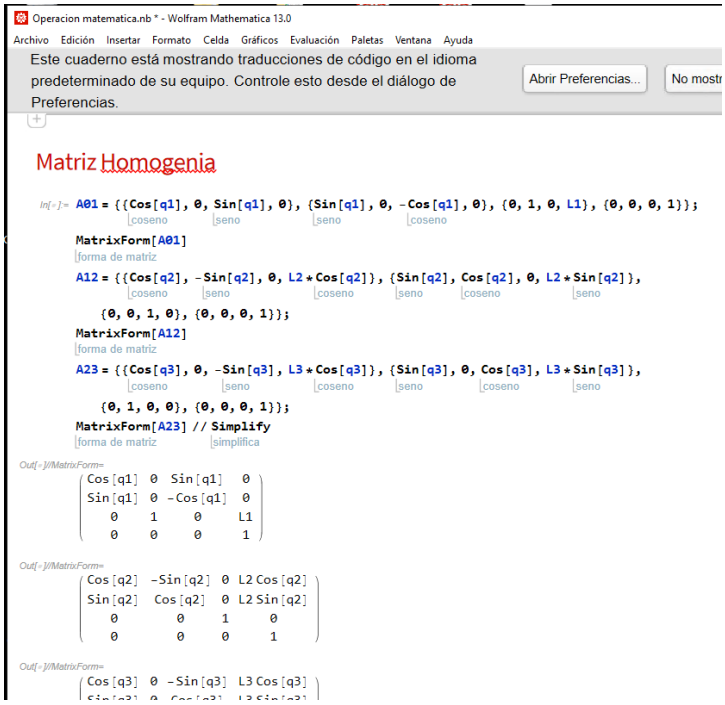
MATLAB es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio.

En este proyecto se utilizó Matlab para realizar cálculos matriciales y construir el modelo del brazo robot en espacio 3D, como lo que muestra en la figura siguiente.



Wolfram Mathematica

Wólfram Mathematica es un sistema de algebra computacional, programación y visualización desarrollado por Wolfram Reserch. En el presente proyecto se ha utilizado para realizar operaciones matemáticas con finalidad de obtener el modelo dinámico del brazo robot. Como lo que muestra en la siguiente figura, se puede hacer operaciones matemáticas utilizando diferentes variables nombrado por el usuario sin asignarle ningún valor.



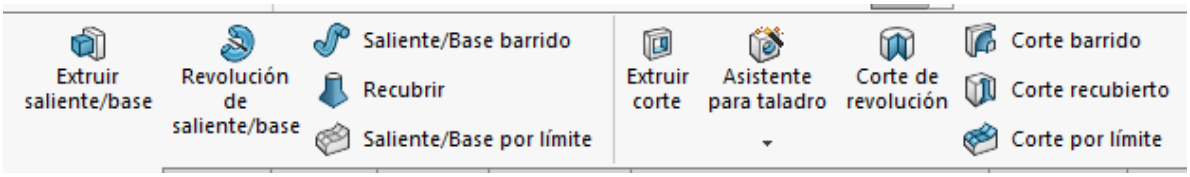
SolidWorks

Para modelar la pieza de acople del servomotor, se ha adquirido los conocimientos aprendidos en las secciones de practica Solidworks y los correspondientes tutoriales, el programa “SOLIDWORKS” es un software CAD para modelado mecánico tanto 2D como 3D.

Una de las operaciones más importante es “Croquis”, se crea un croquis en un plano cualquiera y se modela el mecanismo en un plano 2D, en la siguiente figura representa las herramientas más utilizadas, como “línea”,” rectángulo”,” Circulo”, etc.

Barra de herramienta croquis.

Las otras operaciones más importantes son “Extruir saliente” y “Extruir corte”, esas dos operaciones nos permiten representar el Croquis (2D) en un mecanismo de 3D.



Barra de herramienta operaciones.

Código fuente

```
/*******************************************************************************

/* Xiangxiang SHAN */

/* Manipulador movil TFM*/



#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <errno.h>
#include <time.h>              // for time()
#include <unistd.h>      // for sleep()
#include "dynamixel_sdk.h"

// Control table address
#define TORQUE_ENABLE        64        // Control table address is different in Dynamixel model
#define GOAL_POSITION        116
#define PRESENT_POSITION      132
#define GOAL_CURRENT                  102
#define GOAL_VELOCITY                 104
#define PROFILE_VELOCITY         112
#define PROFILE_ACCELERATION   108
#define Kp     84
#define Kd     80
#define Kvp     78
#define Kvi     76

// Protocol version
#define PROTOCOL_VERSION        2.0   // See which protocol version is used in the Dynamixel

// Definir motores
#define MOTOR1                  1               // Dynamixel ID: 1
#define MOTOR2                  2
#define MOTOR3                  3
#define MOTOR4                  4
#define MOTOR5                  5
#define MOTOR6                  6

#define BAUDRATE                57600
#define DEVICENAME      "/dev/ttyUSB0"  // Check which port is being used on your controller
                        // ex) Windows: "COM1"   Linux: "/dev/ttyUSB0" Mac: "/dev/tty.usbserial-*"
```

```
#define ACTIVAR                 1               // Value for enabling the torque
#define DESACTIVAR              0               // Value for disabling the torque
#define ESC_ASCII_VALUE             0x1b

//Definir datos geometricos
#define L1    0.14  //Longitud barra 1 en m
#define L2    0.15
#define L3    0.15
#define R     0.03 // Radio de la rueda en m
#define m2     0.250  // Masa del cuerpo, como es simétrico, pesa lo mismo m1=m2=m3, en
kilogramo.
#define m3    0.250
#define pi   3.1415927
#define D          0.2
#define g          9.81

//Definir variables interno
#define T 5
#define conv_vel 0.024085544 //Conversion de unidad de velocidad dynamixel a rad/s
#define N 1000
#define w_r 250            //velocidad angular rueda rapida
#define w_l 100            //velocidad angular rueda lenta
#define tm 0.05//Tiempo de muestreo en 50ms

//Variable global
float dist,X,Y,Z,Vx,Vy,Vz;;
int i;

//varibles para calculo de cinematica movil
float t,xi,yi,wd,wi;
float x_ori,x_fin,y_ori,y_fin,xactual,yactual,gama_actual;
float vd,vi,V,w,muestreo;


//Variable para cinematica inversa
float q1,q2,q3,q1p,q2p,q3p; //angulos que deben calcular para cada articulacion y la velocidad
float r,beta,alfa,G2; //Coordenada cartesiana del punto final
int q1f,q2f,q3f,conv,corr; //numeros enteros porque son bit de 0 a 4095


//Inicializacion de Dynamixel
int getch()
{
#if defined(__linux__) || defined(__APPLE__)
  struct termios oldt, newt;
  int ch;
  tcgetattr(STDIN_FILENO, &oldt);
  newt = oldt;
```

```
  newt.c_lflag &= ~(ICANON | ECHO);
  tcsetattr(STDIN_FILENO, TCSANOW, &newt);
  ch = getchar();
  tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
  return ch;
#elif defined(_WIN32) || defined(_WIN64)
  return _getch();
#endif
}

int kbhit(void)
{
#if defined(__linux__) || defined(__APPLE__)
  struct termios oldt, newt;
  int ch;
  int oldf;

  tcgetattr(STDIN_FILENO, &oldt);
  newt = oldt;
  newt.c_lflag &= ~(ICANON | ECHO);
  tcsetattr(STDIN_FILENO, TCSANOW, &newt);
  oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
  fcntl(STDIN_FILENO, F_SETFL, oldf | O_NONBLOCK);

  ch = getchar();

  tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
  fcntl(STDIN_FILENO, F_SETFL, oldf);

  if (ch != EOF)
  {
    ungetc(ch, stdin);
    return 1;
  }

  return 0;
#elif defined(_WIN32) || defined(_WIN64)
  return _kbhit();
#endif
}

// Funciones creadas
float gentra(float qi,float qf,float t,float Periodo){
        float b,a,d,c,x;
        b=3*(qf-qi)/(Periodo*Periodo);
        a=(-2*(qf-qi))/(Periodo*Periodo*Periodo);
        d = qi;
        c=0;
```

```
   x=a*pow(t,3)+b*pow(t,2)+c*t+d;
   return x;
}


int main()
{
  // Initialize PortHandler Structs
  // Set the port path
  // Get methods and members of PortHandlerLinux or PortHandlerWindows
  int port_num = portHandler(DEVICENAME);
  // Initialize PacketHandler Structs
  packetHandler();
//  int dxl_comm_result = COMM_TX_FAIL;           // Communication result
//  uint8_t dxl_error = 0;                        // Dynamixel error
//  int32_t dxl_present_position = 0;             // Present position
  // Open port
  if (openPort(port_num))
  {
    printf("Succeeded to open the port!\n");
  }
  else
  {
    printf("Failed to open the port!\n");
    printf("Press any key to terminate...\n");
    getch();
    return 0;
  }
  // Set port baudrate
  if (setBaudRate(port_num, BAUDRATE))
  {
    printf("Succeeded to change the baudrate!\n");
  }
  else
  {
    printf("Failed to change the baudrate!\n");
    printf("Press any key to terminate...\n");
    getch();
    return 0;
  }


void giro_izq_dir(){
                    //Configuracion inicial motor 5 y 6
          write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, Kvp, 800);
          write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, Kvi, 1500);
          write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, Kvp, 800);
          write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, Kvi, 1500);
```

```
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, -w_r);
//motor derecha va mas rapido
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY, w_l);
//motor izquierda
}

void giro_der_dir(){
                //Configuracion inicial motor 5 y 6
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, Kvp, 800);
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, Kvi, 1500);
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, Kvp, 800);
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, Kvi, 1500);
        write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, -w_l);
//motor derecha
        write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY,
w_r);//motor izquierda va mas rapido
}

void giro_izq_inv(){
                //Configuracion inicial motor 5 y 6
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, Kvp, 800);
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, Kvi, 1500);
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, Kvp, 800);
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, Kvi, 1500);
        write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, w_r);
//motor derecha va mas rapido
        write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY, -
w_l);//motor izquierda
}

void giro_der_inv(){
                //Configuracion inicial motor 5 y 6
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, Kvp, 800);
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, Kvi, 1500);
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, Kvp, 800);
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, Kvi, 1500);
        write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, w_l);
//motor derecha
        write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY, -
w_r);//motor izquierda va mas rapido
}

void stop(){
        write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, 0);
        write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY, 0);
}
void estado_reposo(){
        write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, Kp, 1700); //
```

```
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, Kd, 200);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, PROFILE_VELOCITY, 80);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, PROFILE_ACCELERATION, 16);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, GOAL_POSITION, 3071);
//Escribir valores de goal position en el servo

//Configuracion inicial motor 2
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, Kp, 1500);
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, Kd, 500);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, GOAL_CURRENT, 1500);//corr
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, PROFILE_VELOCITY, 180);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, PROFILE_ACCELERATION, 16);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, GOAL_POSITION, q2f);

//Configuracion inicial motor 3
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, Kp, 1700);
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, Kd, 200);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, PROFILE_VELOCITY, 80);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, PROFILE_ACCELERATION, 16);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, GOAL_POSITION, 0);
}

void gira_angulo(float gama_act){
        float angulo;
        int t_giro;
        angulo=pi-gama_act;
        if(angulo<0){
                angulo=angulo+2*pi;
        }
        t_giro=(angulo/0.39525)*1000000; // Uds en micro segundo
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR5, GOAL_VELOCITY, -50); //motor
derecha
 write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR6, GOAL_VELOCITY, -50);//motor
izquierda va mas rapido
   usleep(t_giro);
}
/*write1ByteTxRx(int port_num, int protocol_version, int id, int address, int value);*/
 while (1)
 {

        printf("\nDame las coordenadas en XYZ: ");
        scanf("%f %f %f",&X,&Y,&Z);
        dist=sqrt((X*X)+(Y*Y));

        if(dist>0.250){ // Cuando esta fuera de area de trabajo del brazo robot, se procede
mover la base movil para acercar al objeto

                muestreo=T/tm;
```

```
        x_fin=X-0.2;
        y_fin=Y-0.2;
        //resetea los valores iniciales
        x_ori=0;
        y_ori=0;
        xactual=0;
        yactual=0;
        gama_actual=0;
        t=0;
for (i=1;i<muestreo;i++){
        t=t+tm;
        xi=gentra(x_ori,x_fin,t,T);
        yi=gentra(y_ori,y_fin,t,T);

        if((y_fin-yactual<=0.01)&&(x_fin-xactual<=0.01)){ // Cuando este muy cerca, se para
                                stop();
        }else{
                if(yi>=yactual&&xi>=xactual){     //Giro izquierda sentido directo
                        giro_izq_dir();
                        wd=w_r;
                        wi=w_l;
                        printf("\nGirar izq sentido directo");
                        usleep(tm*1000000); // 50000 microsegundos = 50ms
        }
                if(yi>=yactual&&xi<xactual){     //Giro izquierda sentido inverso
                        giro_izq_inv();
                        wd=-w_r;
                        wi=-w_l;
                        printf("\nGirar izq sentido inverso");
                        usleep(tm*1000000);
                }
                if(yi<yactual&&xi>=xactual){     //Giro izquierda sentido directo
                        giro_der_dir();
                        wd=w_l;
                        wi=w_r;
                        printf("\nGirar derecha sentido directo");
                        usleep(tm*1000000);
                }
                if(yi<yactual&&xi<xactual){     //Giro izquierda sentido inverso
                        giro_der_inv();
                        wd=-w_l;
                        wi=-w_r;
                        printf("\nGirar derecha sentido inverso");
                        usleep(tm*1000000);
                }
                vd=wd*R*conv_vel;  //velocidad lineal rueda derecha
                vi=wi*R*conv_vel; //velocidad lineal rueda izquierda
                V=(vd+vi)/2;  //velocidad lineal
```

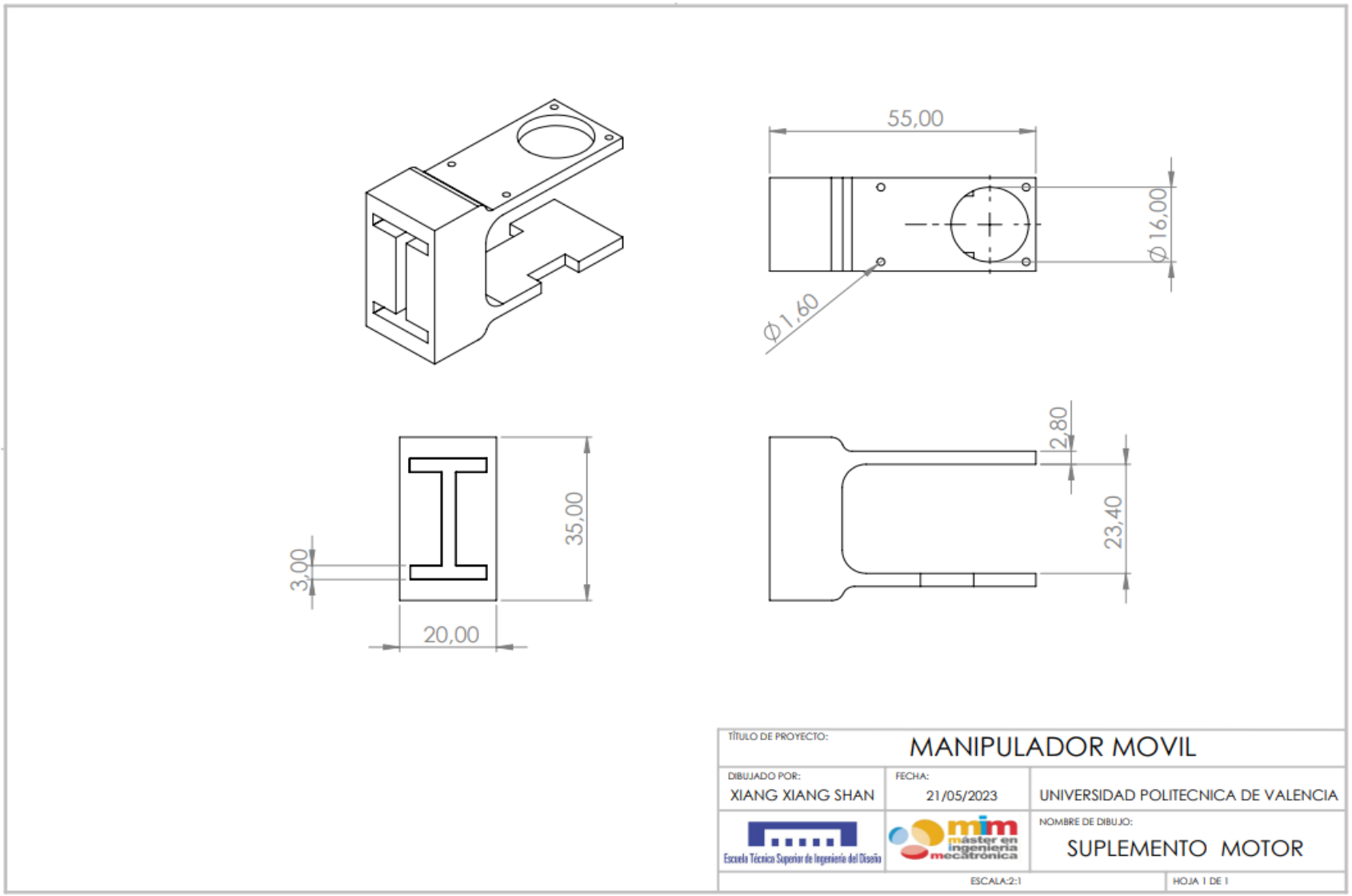
```
                w=(vd-vi)/D;  //velocidad angular
                printf("\nCiclo: %d  Tiempo: %.2f",i,t);
                printf("\nXobj: %.4f  Xactual: %.4f  Diferencia: %.4f ",xi,xactual,xi-xactual);
                printf("\nYobj: %.4f  Yactual: %.4f  Diferencia: %.4f ",yi,yactual,yi-yactual);
                //Acumulando los valores de gama, x e y.
                gama_actual=gama_actual+w*tm;
                xactual=xactual+V*cos(gama_actual)*tm;
                yactual=yactual+V*sin(gama_actual)*tm;
                printf("\nAngulo: %.3f  ",gama_actual);
                printf("\nVelocidad lineal: Vd %.3f Vi: %.3f",vd,vi);
                printf("\nVelocidad lineal y angular: V %.3f w: %.3f\n",V,w);
                }
            if(i==muestreo){
                  stop();
                }
          }
          }
X=X-xactual;
Y=Y-yactual;
dist=sqrt((X*X)+(Y*Y));
if(dist>0.250){
  stop();
}else{
printf("\nNueva X Y Z: %.2f %.2f %.2f",X,Y,Z);
  //Cinematica inversa partiendo del posicion dato XYZ
 q1=atan2(Y,X)-gama_actual;
r=sqrt((X*X)+(Y*Y));
q3=(acos((pow(r,2)+pow((Z-L1),2)-pow(L2,2)-pow(L3,2))/(2*L2*L3)));
beta=atan((Z-L1)/r);
alfa=atan((L3*sin(q3))/(L2+L3*cos(q3)));
q2 =(beta-alfa);
printf("\nCodo abajo: q1 q2 q3 :%.3f , %.3f , %.3f",q1*57.2957,q2*57.2957,q3*57.2957);
conv=4095/(2*pi);                              //Convertir radianes en numero bit Dynamixel
//Se ha cambiado para el modo de codo arriba
q2=q2+2*alfa;
q3=-q3;
printf("\ncodo arriba:   q1 q2 q3 :%.3f , %.3f , %.3f",q1*57.2957,q2*57.2957,q3*57.2957);
// como el rango de escritura es de 0 a 4095, en caso de numero negativo, hay que sumarle
4095
q1f=q1*conv+3071; if(q1f<0){q1f=q1f+4095;}
//if(q2<0){q2f=-(q2+2*alfa)*conv*2.15+4095;}
q2f=-(q2)*conv*1.857+1900; //if(q2f<0){q2f=q2f+4095;}
q3f=(q3)*conv+1023; if(q3f<0){q3f=q3f+4095;}
printf("\nq1:%d",q1f);
printf("\nq2:%d",q2f);
printf("\nq3:%d",q3f);
```

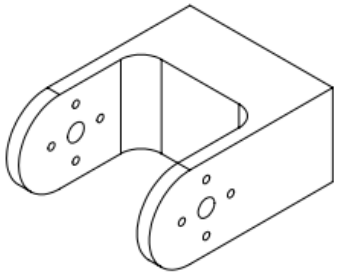
```
q1p=80*conv_vel;
q2p=180*conv_vel;
q3p=80*conv_vel;
Vx=-(sin(q1)*(L2*cos(q2)+L3*cos(q2+q3)))*q1p-cos(q1)*(L2*cos(q2)+L3*cos(q2+q3))*q2p-
L3*cos(q1)*sin(q2+q3)*q3p; //Velocidad lineal en x
Vy=-(cos(q1)*(L2*cos(q2)+L3*cos(q2+q3)))*q1p-sin(q1)*(L2*sin(q2)+L3*sin(q2+q3))*q2p-
L3*sin(q1)*sin(q2+q3)*q3p; //Velocidad lineal en x
Vz=(L2*cos(q2)+L3*cos(q2+q3))*q2p+L3*cos(q2+q3)*q3p;
G2=-m2*(0.5*g*L2*cos(q2))-m3*(0.5*g*(2*L2*cos(q2)+0.001*L3*cos(q2+q3)));
corr=(-G2/1.818)*1000/0.353;//Conversion de par a goal current.
if(q2f<0||q2f>2200||q3f>2046){//Cuando alguno de los valores esta fuera de rango de
trabajo
printf("\nZona inalcanzable, vuelve a seleccionar.");
}else{
        // Write goal position
        //Configuracion inicial motor 1
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, Kp, 1700); //
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, Kd, 200);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, PROFILE_VELOCITY, 80);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, PROFILE_ACCELERATION, 16);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR1, GOAL_POSITION, q1f); //Escribir
valores de goal position en el servo
        //Configuracion inicial motor 2
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, Kp, 1500);
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, Kd, 500);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, GOAL_CURRENT, 1500);//corr
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, PROFILE_VELOCITY, 180);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, PROFILE_ACCELERATION, 16);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR2, GOAL_POSITION, q2f);
        //Configuracion inicial motor 3
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, Kp, 1700);
write2ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, Kd, 200);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, PROFILE_VELOCITY, 80);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, PROFILE_ACCELERATION, 16);
write4ByteTxRx(port_num, PROTOCOL_VERSION, MOTOR3, GOAL_POSITION, q3f);

        printf("\nLa velocidad lineal es siguiente:");
        printf("\nVx:%f",Vx);
        printf("\nVy:%f",Vy);
        printf("\nVz:%f",Vz);
        }
  }
estado_reposo();
gira_angulo(gama_actual);
                xactual=0;
                yactual=0;
                gama_actual=0;
                t=0;
```

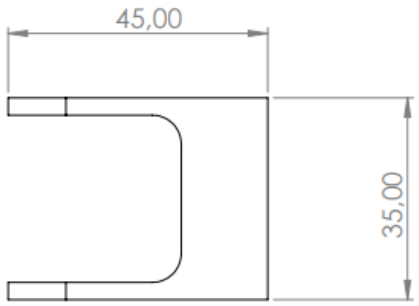
```
for (i=1;i<muestreo;i++){
t=t+tm;
xi=gentra(x_ori,x_fin,t,T);
yi=gentra(y_ori,y_fin,t,T);

if((y_fin-yactual<=0.01)&&(x_fin-xactual<=0.01)){ // Cuando este muy cerca, se para
        stop();
}else{
        if(yi>=yactual&&xi>=xactual){     //Giro izquierda sentido directo
                giro_izq_dir();
                wd=w_r;
                wi=w_l;
                printf("\nGirar izq sentido directo");
                usleep(tm*1000000); // 50000 microsegundos = 50ms
                                        }
        if(yi>=yactual&&xi<xactual){     //Giro izquierda sentido inverso
                giro_izq_inv();
                wd=-w_r;
                wi=-w_l;
                printf("\nGirar izq sentido inverso");
                usleep(tm*1000000);
        }
        if(yi<yactual&&xi>=xactual){     //Giro izquierda sentido directo
                giro_der_dir();
                wd=w_l;
                wi=w_r;
                printf("\nGirar derecha sentido directo");
                usleep(tm*1000000);
        }
        if(yi<yactual&&xi<xactual){     //Giro izquierda sentido inverso
                giro_der_inv();
                wd=-w_l;
                wi=-w_r;
                printf("\nGirar derecha sentido inverso");
                usleep(tm*1000000);
        }
        vd=wd*R*conv_vel;  //velocidad lineal rueda derecha
        vi=wi*R*conv_vel; //velocidad lineal rueda izquierda
        V=(vd+vi)/2;  //velocidad lineal
        w=(vd-vi)/D;  //velocidad angular
        printf("\nCiclo: %d  Tiempo: %.2f",i,t);
        printf("\nXobj: %.4f  Xactual: %.4f  Diferencia: %.4f ",xi,xactual,xi-xactual);
        printf("\nYobj: %.4f  Yactual: %.4f  Diferencia: %.4f ",yi,yactual,yi-yactual);
        //Acumulando los valores de gama, x e y.
        gama_actual=gama_actual+w*tm;
        xactual=xactual+V*cos(gama_actual)*tm;
        yactual=yactual+V*sin(gama_actual)*tm;
        printf("\nAngulo: %.3f  ",gama_actual);
```

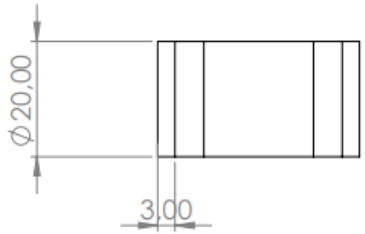
```
                printf("\nVelocidad lineal: Vd %.3f Vi: %.3f",vd,vi);
                printf("\nVelocidad lineal y angular: V %.3f w: %.3f\n",V,w);
        }
            if(i==muestreo){
                stop();
                }
        }
        printf("\nPress any key to continue! (or press ESC to quit!)\n");
   if (getch() == ESC_ASCII_VALUE)
   break;
}
 return 0;
}
```

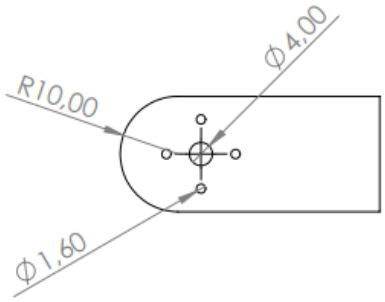
Planos


55,00
Ø 16,00
Ø1,60
2,80
23,40
35,00
3,00
20,00
TÍTULO DE PROYECTO:
MANIPULADOR MOVIL
DIBUJADO POR:
XIANG XIANG SHAN
FECHA:
21/05/2023
UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño
mim máster en ingeniería mecatrónica
NOMBRE DE DIBUJO:
SUPLEMENTO MOTOR
ESCALA:2:1
HOJA 1 DE 1

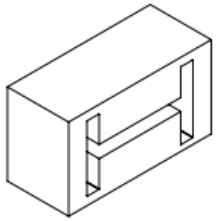


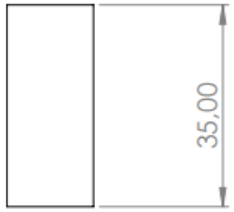


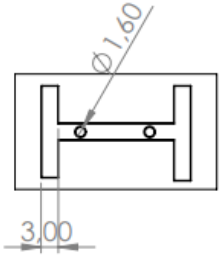


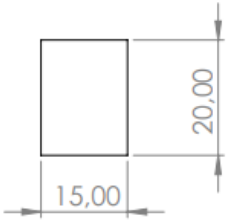


TÍTULO DE PROYECTO: MANIPULADOR MOVIL		
DIBUJADO POR: XIANG XIANG SHAN	FECHA: 21/05/2023	UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño	master en ingeniería mecatrónica	NOMBRE DE DIBUJO: SUPLEMENTO MOTOR
ESCALA:2:1		HOJA 1 DE 1

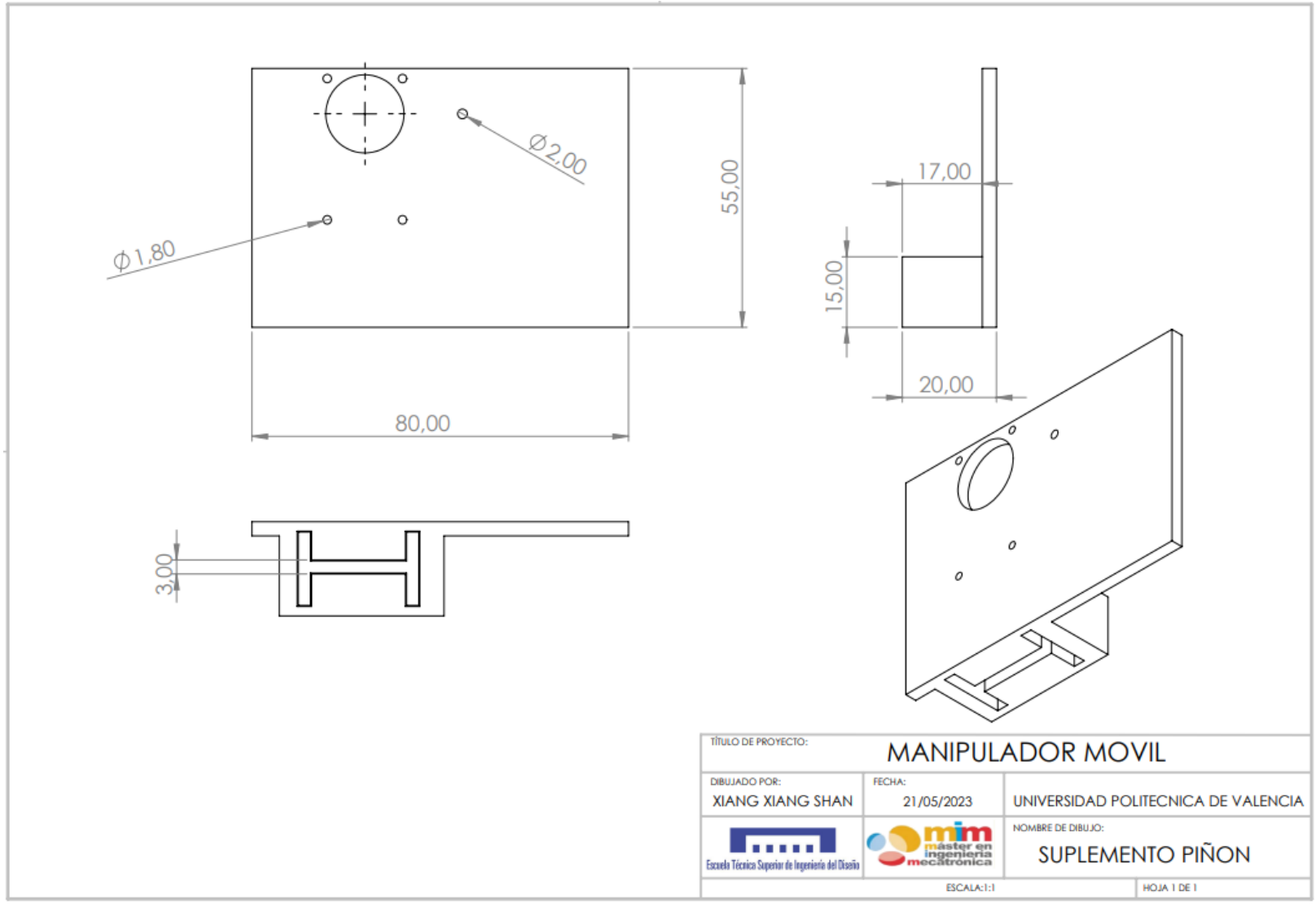


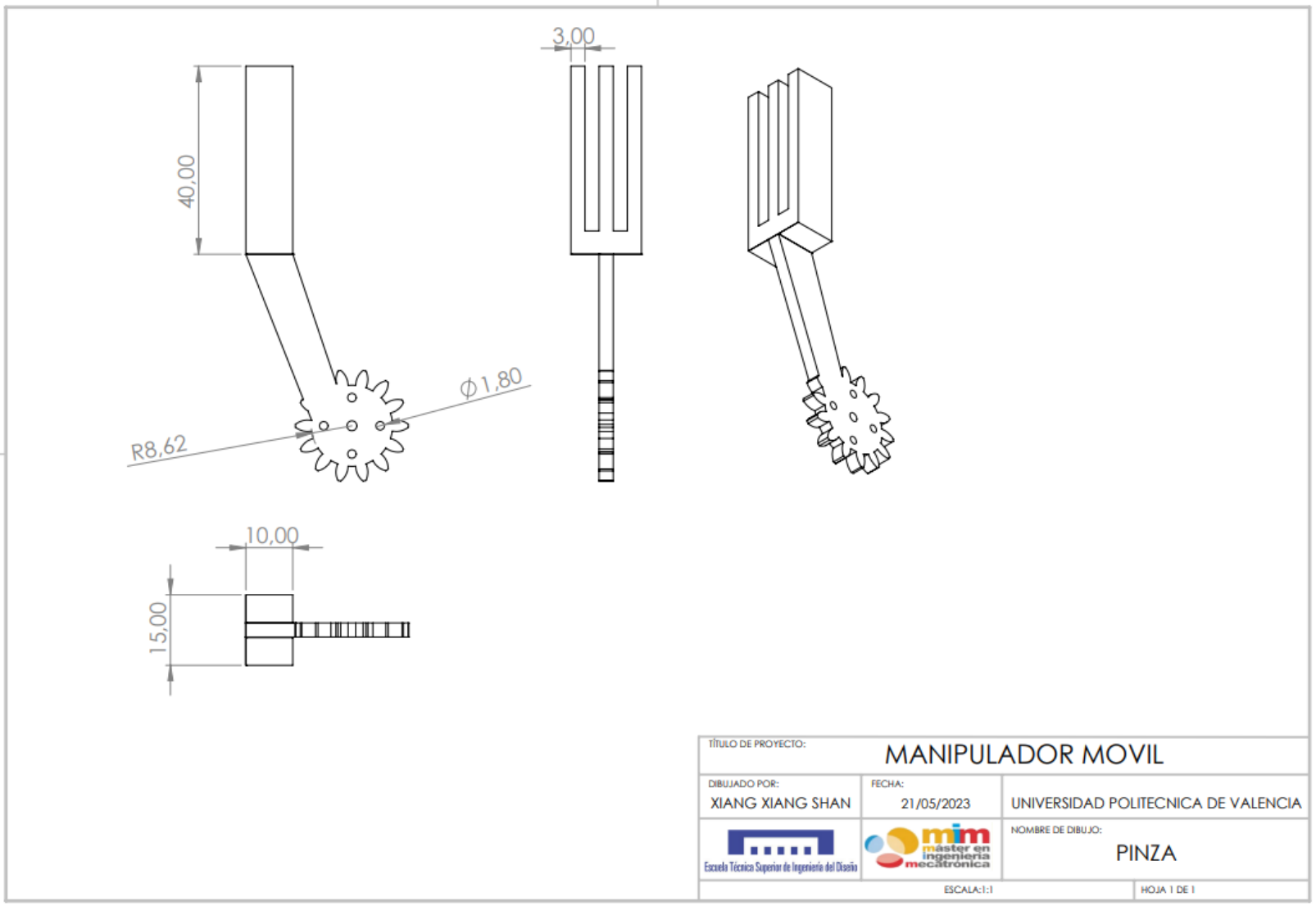


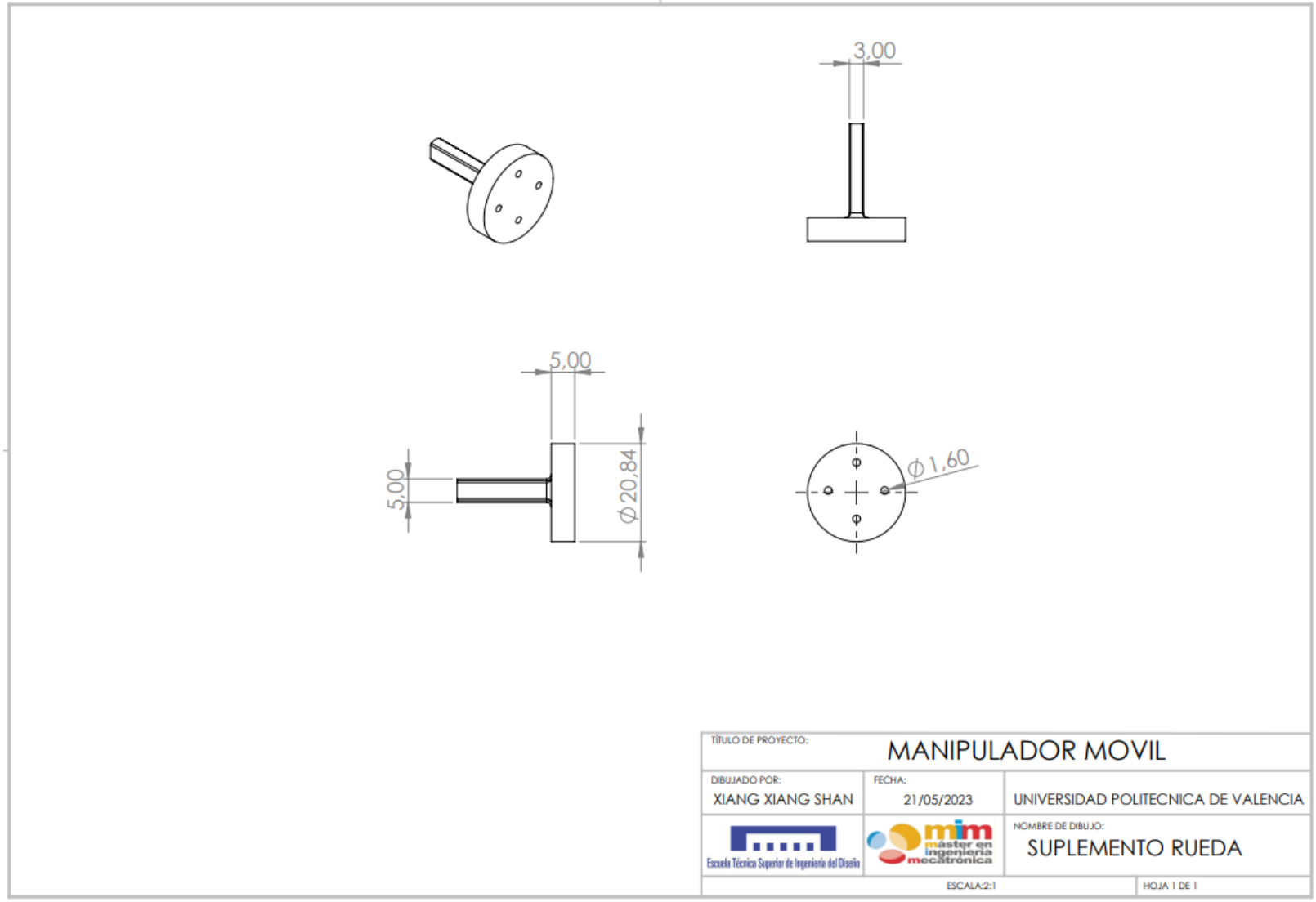


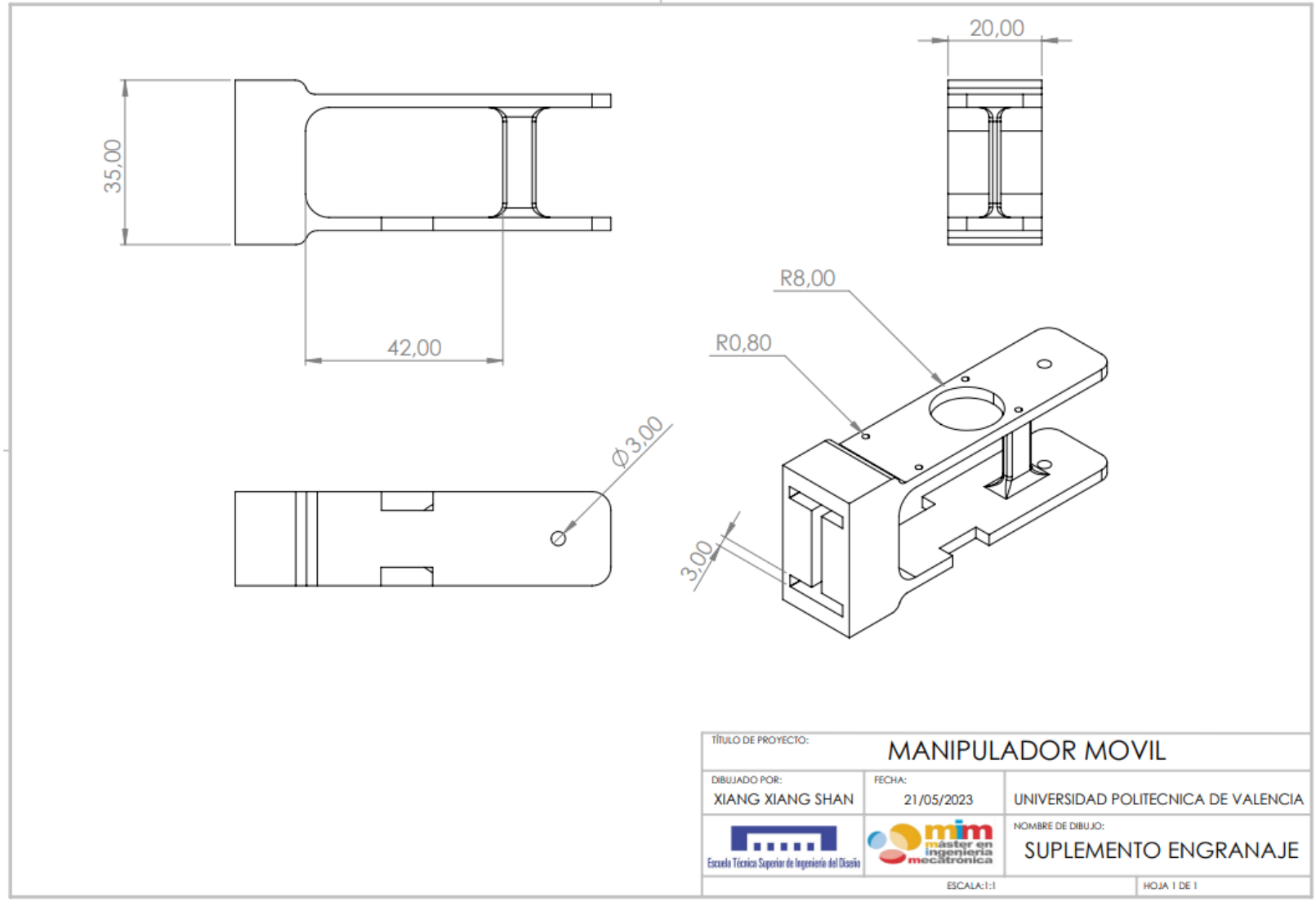


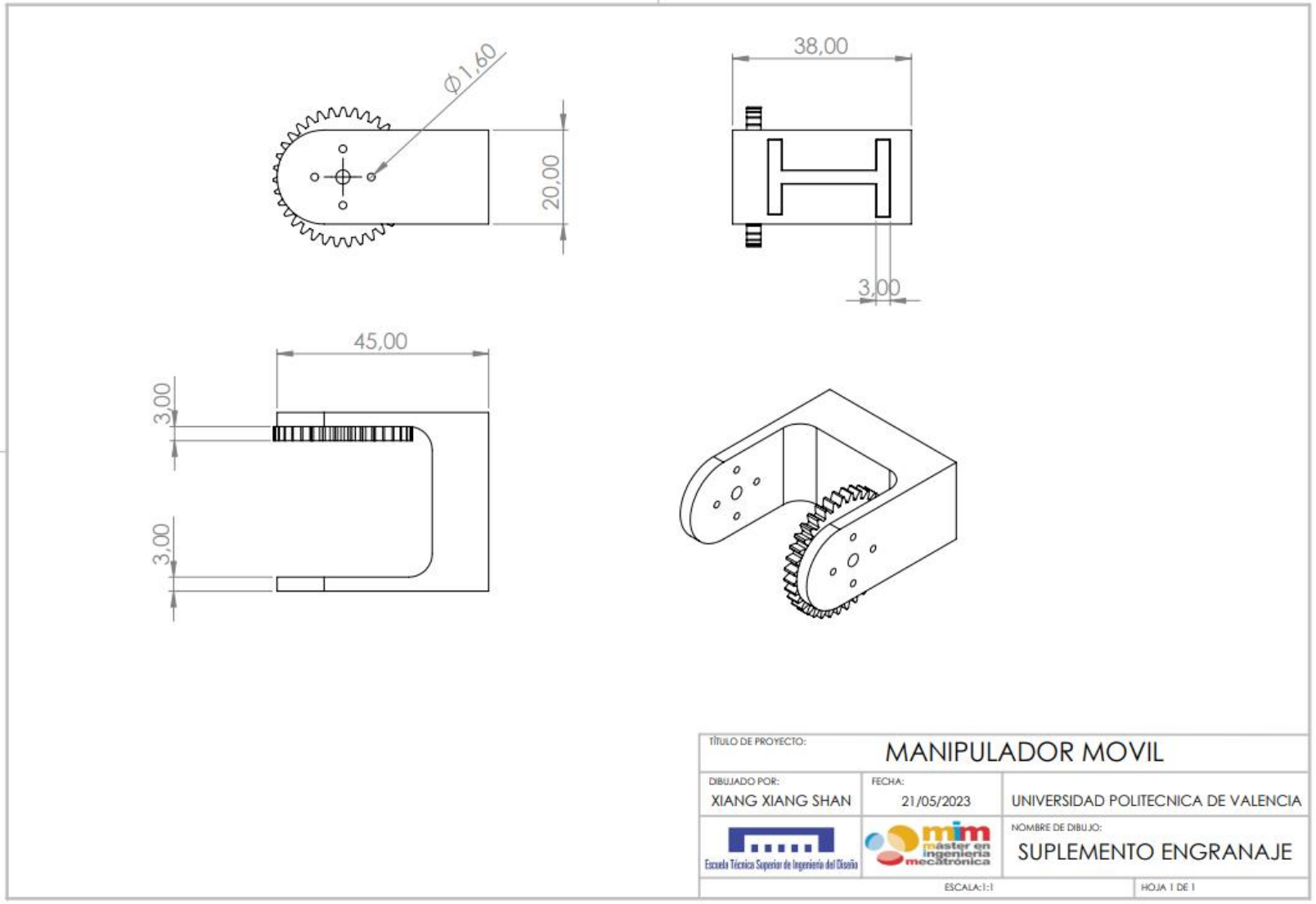
TÍTULO DE PROYECTO: MANIPULADOR MOVIL		
DIBUJADO POR: XIANG XIANG SHAN	FECHA: 21/05/2023	UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño	mim master en ingeniería mecatrónica	NOMBRE DE DIBUJO: SUPLEMENTO MOTOR
ESCALA:2:1		HOJA 1 DE 1


Ø2,00
Ø1,80
55,00
80,00
3,00
17,00
15,00
20,00
TÍTULO DE PROYECTO:
MANIPULADOR MOVIL
DIBUJADO POR:
XIANG XIANG SHAN
FECHA:
21/05/2023
UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño
master en ingeniería mecatrónica
NOMBRE DE DIBUJO:
SUPLEMENTO PIÑON
ESCALA:1:1
HOJA 1 DE 1

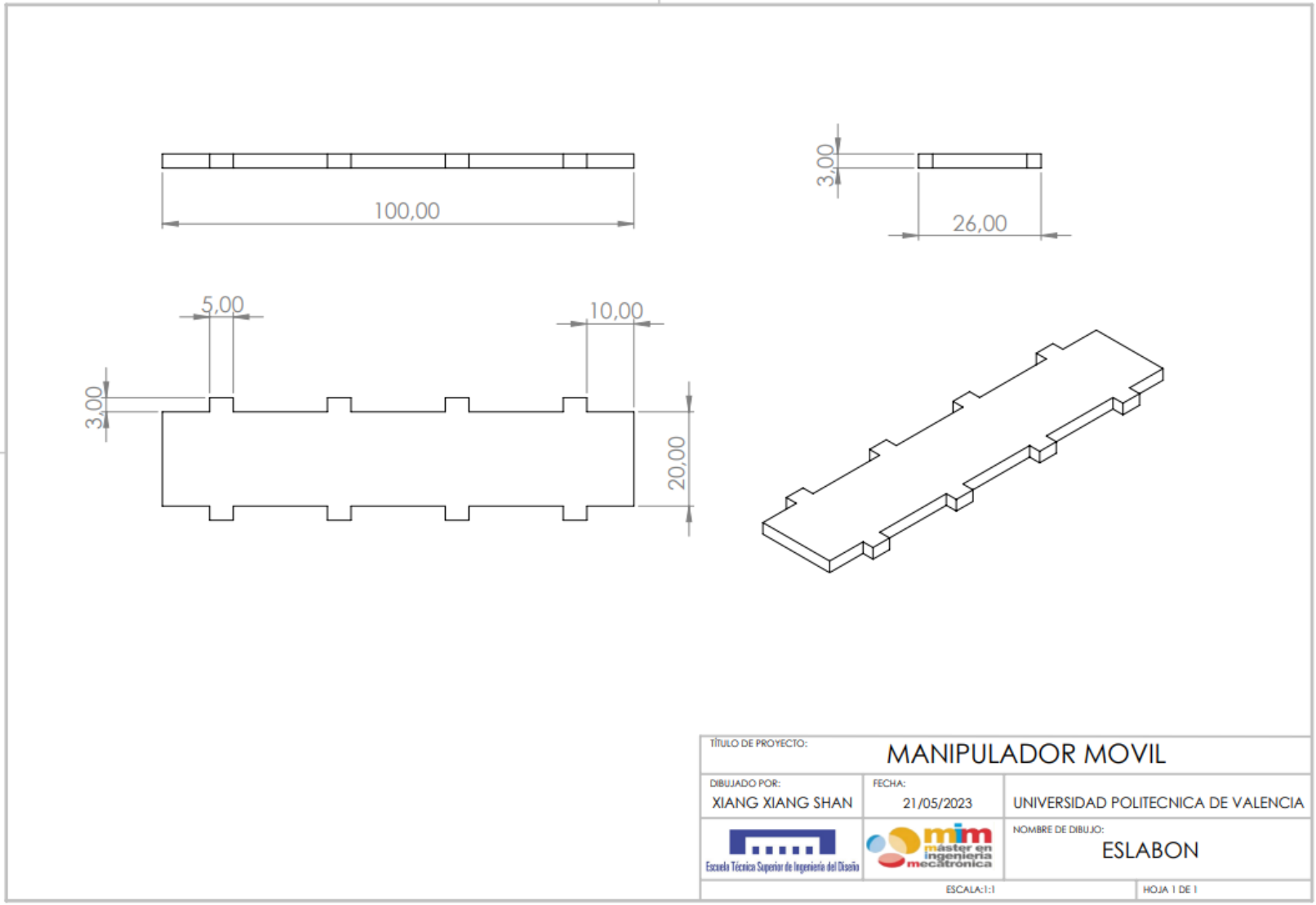

3,00
40,00
R8,62
Φ1,80
10,00
15,00
TÍTULO DE PROYECTO:
MANIPULADOR MOVIL
DIBUJADO POR:
XIANG XIANG SHAN
FECHA:
21/05/2023
UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño
máster en ingeniería mecatrónica
NOMBRE DE DIBUJO:
PINZA
ESCALA:1:1
HOJA 1 DE 1

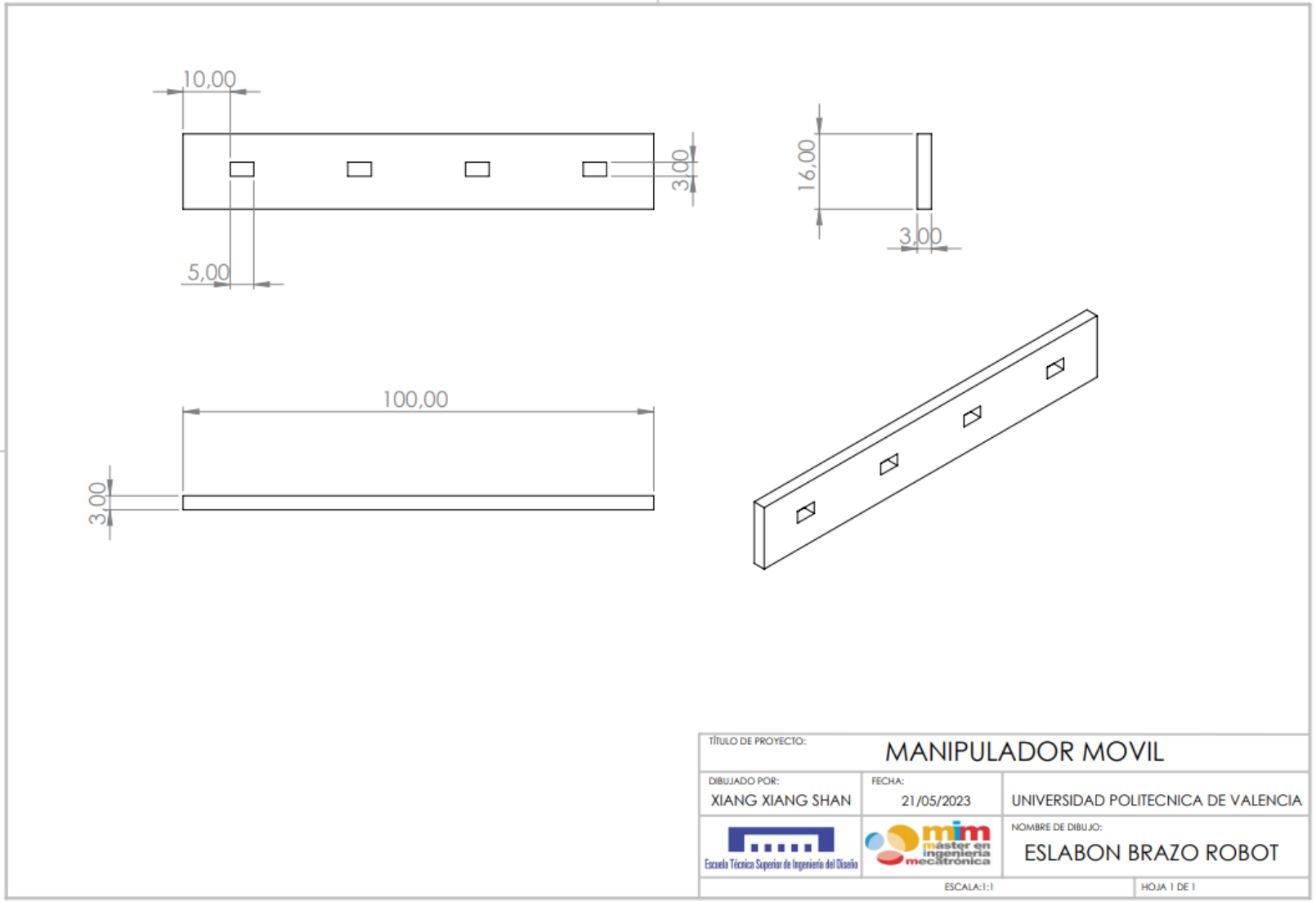

3,00
5,00
5,00
Ø 20,84
Ø 1,60
TÍTULO DE PROYECTO:
MANIPULADOR MOVIL
DIBUJADO POR:
XIANG XIANG SHAN
FECHA:
21/05/2023
UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño
mim master en ingeniería mecatrónica
NOMBRE DE DIBUJO:
SUPLEMENTO RUEDA
ESCALA:2:1
HOJA 1 DE 1

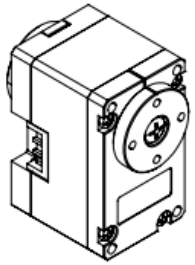

20,00
35,00
42,00
R8,00
R0,80
Ø3,00
3,00
TÍTULO DE PROYECTO:
MANIPULADOR MOVIL
DIBUJADO POR:
XIANG XIANG SHAN
FECHA:
21/05/2023
UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño
master en ingeniería mecatrónica
NOMBRE DE DIBUJO:
SUPLEMENTO ENGRANAJE
ESCALA:1:1
HOJA 1 DE 1

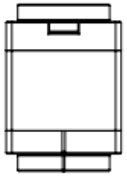


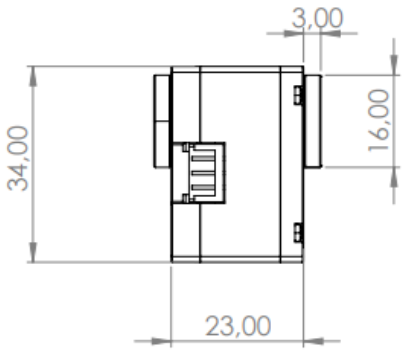
TÍTULO DE PROYECTO: MANIPULADOR MOVIL		
DIBUJADO POR: XIANG XIANG SHAN	FECHA: 21/05/2023	UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño	master en ingeniería mecatrónica	NOMBRE DE DIBUJO: SUPLEMENTO ENGRANAJE
ESCALA:1:1		HOJA 1 DE 1

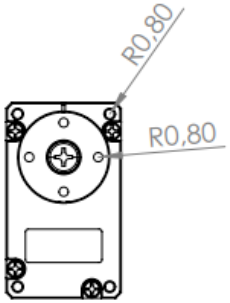

100,00
3,00
26,00
5,00
10,00
3,00
20,00
TÍTULO DE PROYECTO:
MANIPULADOR MOVIL
DIBUJADO POR:
XIANG XIANG SHAN
FECHA:
21/05/2023
UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño
master en ingeniería mecatrónica
NOMBRE DE DIBUJO:
ESLABON
ESCALA:1:1
HOJA 1 DE 1


10,00
3,00
5,00
16,00
3,00
100,00
3,00
TÍTULO DE PROYECTO:
MANIPULADOR MOVIL
DIBUJADO POR:
XIANG XIANG SHAN
FECHA:
21/05/2023
UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño
máster en ingeniería mecatrónica
NOMBRE DE DIBUJO:
ESLABON BRAZO ROBOT
ESCALA:1:1
HOJA 1 DE 1

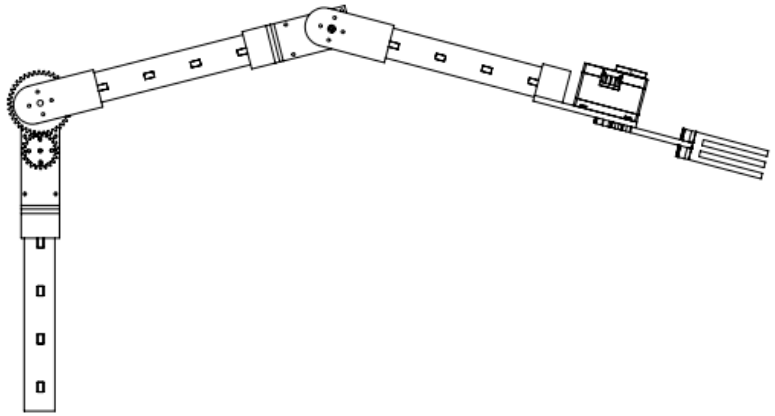


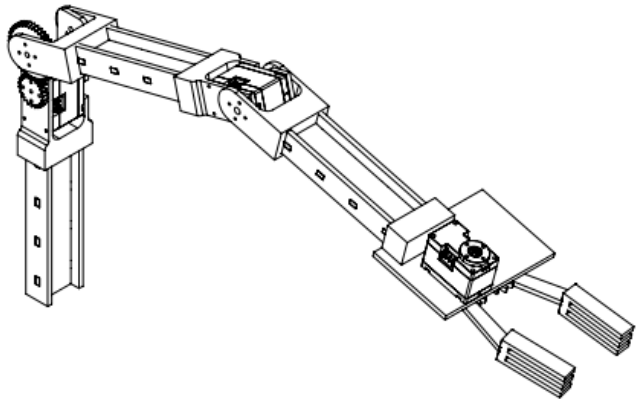


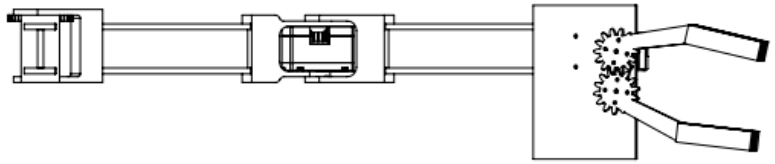




TÍTULO DE PROYECTO: MANIPULADOR MOVIL		
DIBUJADO POR: XIANG XIANG SHAN	FECHA: 21/05/2023	UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño	mim master en ingeniería mecatrónica	NOMBRE DE DIBUJO: MOTOR
ESCALA:1:1		HOJA 1 DE 1







TÍTULO DE PROYECTO: MANIPULADOR MOVIL		
DIBUJADO POR: XIANG XIANG SHAN	FECHA: 21/05/2023	UNIVERSIDAD POLITECNICA DE VALENCIA
Escuela Técnica Superior de Ingeniería del Diseño	mim master en ingeniería mecatrónica	NOMBRE DE DIBUJO: MONTAJE BRAZO ROBOT
ESCALA:1:5		HOJA 1 DE 1