UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

MÁSTER EN INGENIERÍA INDUSTRIAL

"DESARROLLO DE UN PROTOTIPO DE ROBOT PARALELO PARA ENSAYOS DE CONTROL SUB-ACTUADO"

TRABAJO FIN DE MÁSTER

Junio -2024

AUTOR: Esther González Amorós

DIRECTOR/ES: Adrián Peidró Vidal

AGRADECIMIENTOS

A mi familia y amigos por estar en cada momento de esta etapa.

A mi tutor, Adrián Peidró, por confiar en mí, darme la oportunidad de trabajar junto a él y orientarme durante todo el proceso.

Gracias infinitas.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En la actualidad, uno de los problemas de los robots paralelos completamente actuados es que, cuando un robot sufre un fallo en una de sus articulaciones el giro de dicha articulación se vuelve libre y el sistema, por consecuencia, sub-actuado. Este fallo, conocido en inglés como *Free Swing Joint Failure*, FSJF, puede generar problemas importantes, ya que el robot podría oscilar y proporcionar movimientos indeseados que ocasionen problemas de colisiones con otros objetos.

Las soluciones adoptadas habitualmente se basan en el uso de frenos y la adición de más motores, volviendo al sistema redundante, ya que se usarían más de los necesarios. Además de utilizar más elementos de los necesarios, también se encarece el sistema.

Siguiendo el método propuesto en el artículo *"Peidró et al. (2023), Mechanism and Machine Theory, vol 188, 105403"* se propone que, cuando ocurra la falla se lleve cada elemento a posiciones seguras. Estas posiciones serán las llamadas configuraciones de bloqueo durante la memoria y vendrán calculadas por el método de la nueva solución propuesta en artículo.

Por lo tanto, en la presente memoria se diseñará, realizará, ensamblará y programará un robot que estudia una nueva solución al problema de FSJF citado anteriormente. El robot está diseñado originalmente en un artículo científico que se puede encontrar en las referencias de la memoria. El prototipo se basa en original del artículo, pero no se representa fielmente debido a que, para facilitar su implementación física, se tuvieron que añadir dos offsets que varían ligeramente la cinemática del robot, aunque no lo suficiente como para no aplicar el método de forma muy similar a como se realiza en el citado artículo.

Finalmente, durante la memoria se detalla el proceso de montaje del prototipo del robot y la conducción de las articulaciones a las configuraciones de bloqueos calculadas por el método del artículo.

ABSTRACT

Nowadays, one of the main problems with fully actuated parallel robots is when a robot experiences a failure in one of its joints, the rotation of this joint becomes free, resulting in a underactuated system. This failure can lead to significant issues, as the robot oscillating and making unintended movements, which could result in collisions with nearby objects.

The commonly adopted solutions typically rely on the use of brakes and the addition of more motors, making the system redundant by using more components tan necessary. In addition, this also increases the cost of the system.

Following the method proposed in the article "*Peidró et al. (2023), Mechanism and Machine Theory, vol 188, 105403*", it is proposed that when a failure occurs, each element should be brought to safe positions. These positions, termed locking configurations in the article, are calculated using the new solution method proposed in the article. It is noteworthy that this new solution does not require brakes or additional motors.

Therefore, in this report, a robot will be designed, built, assembled, and programed to explore a new solution to the aforementioned free swing joint failure problem. The robot is originally designed in a scientific article referenced in this report. The prototype is based on the original design from the article, but it is not represented exactly, as two offsets were added to facilitate its physical implementation. These offsets slightly modify the robot's kinematics, but the method will be able to aply in a manner very similar to the cited article.

Finally, this report details the process of assembling the robot prototype and guiding the joints to the locking configurations calculated by the method described in the article.

1. Introducción

1.1 Antecedentes

En la actualidad existen diferentes tipos de robots, y por ende, diferentes causas de fallos en ellos. En este proyecto, se desarrollará un prototipo de robot con el objetivo de validar un estudio realizado mediante simulación y programación, que propone una solución a uno de los fallos más comunes en robots, el cual se resumirá brevemente a continuación.

Este estudio surge porque uno de los fallos más frecuentes en robots paralelos es el que ocurre entre el motor y la articulación, conocido en inglés como "Free Swing Joint Failure (FSJF)". Este fallo provoca un movimiento libre de la articulación, lo cual puede ser peligroso debido al riesgo de colisión. Dicho movimiento es incontrolable, lo que aumenta su peligrosidad.

Este tipo de fallo puede producirse por diferentes motivos, entre ellos tenemos: desgaste mecánico, falta de mantenimiento, errores de diseño, impactos o sobrecargas, problemas de control… Para poder evitarlo es necesario un buen seguimiento y mantenimiento de los robots, a la vez que un buen diseño mecánico y controles que nos permitan garantizar su estabilidad y precisión en los movimientos efectuados.

Algunas de las soluciones propuestas suelen ser el uso de frenos o el uso de actuadores redundantes, es decir, más motores. Estas soluciones tienen como inconveniente que encarecen el sistema, ya que son más elementos para este y que no serían del todo necesarios. Por lo tanto, siguiendo el método propuesto en el artículo científico “Locking underactuated robots by shrinking their manifolds of free-swinging motion” [1] se procede a la realización del prototipo del robot para la comprobación de la propuesta de solución.

La solución propuesta presenta un enfoque alternativo para garantizar la estabilidad de los robots sub-actuados después del fallo en la articulación de giro libre. Este método se basa en llevar las articulaciones a posiciones de freno o lo que llamaremos durante el proyecto como configuraciones de bloqueo, eliminando de esta manera los movimientos incontrolables y la inestabilidad que pueda desarrollar el sistema.

Los robots sub-actuados son los que tienen un número menor de motores que de grados de libertad. Diferenciándose de los completamente actuados en que estos últimos son los que tienen un motor para cada grado de libertad, y de los redundantes, que tienen más motores que grados de libertad. El prototipo a desarrollar será de dos grados de libertad completamente actuado, pero con el plan de poder convertirlo a un robot sub-actuado, que simulará el fallo FSJF, mediante el uso de un embrague electromagnético que se describirá en los siguientes apartados.

En la presente memoria haremos referencia a un robot paralelo RPRRR, es decir, sigue una cadena cinemática con cuatro juntas de revolución y una prismática.

En la siguiente imagen se observará el robot a desarrollar. Concretamente, se tiene que la barra L1 es el bastidor del sistema, con juntas de revolución en los extremos O y D. La barra L2, con juntas de revolución en los extremos D y C. La barra L3, con juntas de revolución en B y C. Cabe destacar que en la imagen se observa una pinza en la barra L3 que no se dispondrá en el prototipo del robot, ya que para la comprobación del método no es necesaria. Y, por último, la barra ρ con juntas de revolución en los extremos B y O y, además, una prismática.

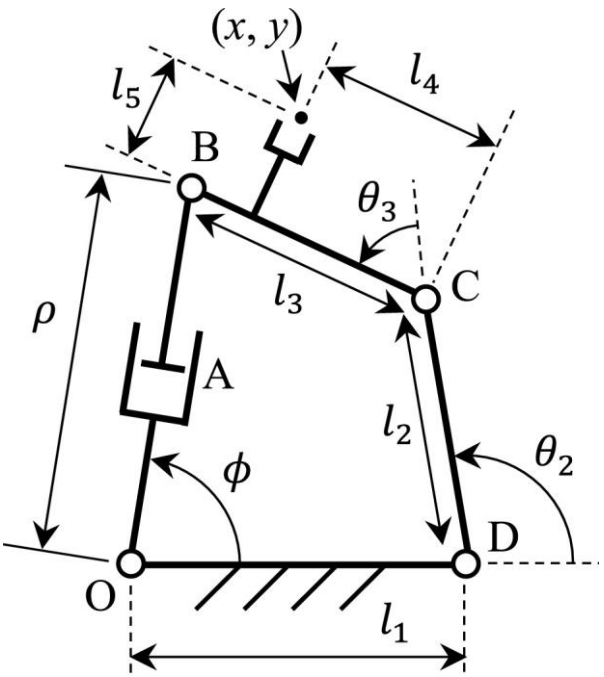


Figura 1. Esquema del robot sub-actuado reproducido de [1] con permiso.

Es importante mencionar que las dimensiones del prototipo serán explicadas con detalle en los siguientes puntos de la memoria, debido a que el prototipo será a una escala determinada siguiendo los requerimientos de montaje del sistema.

Como podemos observar en la imagen de la Figura 1 se tiene el robot paralelo sub-actuado con la cadena cinemática RPRRR en el que basaremos el montaje de nuestro sistema para poder llevar las barras a sus distintas configuraciones de bloqueo y así, poder validar los resultados de simulación presentados en [1].

1.2 Objetivos

Respecto a los objetivos del proyecto se tiene el diseño, construcción y programación del sistema previamente descrito.

Como objetivo principal tenemos la materialización del sistema de la Figura 1 en forma de prototipo para su posterior puesta en marcha y comprobación del método propuesto en el artículo [1].

Sin adentrarnos demasiado en el método propuesto del artículo [1], la nueva solución propone que para gestionar el FSJF sin necesidad de actuadores redundantes ni frenos hay que centrarse en encontrar configuraciones de bloqueo para las articulaciones actuadas que aún funcionan. Al utilizar un proceso de discretización y clasificación de estas configuraciones, se pueden identificar posiciones específicas que limitan el movimiento del robot a puntos fijos, eliminando movimientos incontrolados y asegurando el bloqueo seguro del robot sub-actuado. El conjunto de curvas de movimiento que ofrecen dichas configuraciones da lugar al *manifold* de movimiento libre, después del producirse el fallo en la articulación.

Para el montaje del robot, nos basaremos en las medidas propuestas en el artículo [1] con modificaciones debido a especificaciones que surgen durante el ensamblado de piezas. Estas modificaciones constan de la variación de las medidas de las barras, adecuándolas a las necesidades actuales. Para el cálculo de las dimensiones, se especifica posteriormente el desarrollo que se ha seguido para llegar a su obtención.

Una vez definido el sistema, se propondrán nuevas líneas de mejora del robot y su montaje actual para el afinamiento y precisión del sistema.

1.3 Estructura de la memoria

La presente memoria se dividirá en 6 puntos genéricos, siguiendo las directrices de la Escuela Politécnica Superior de Elche de la Universidad Miguel Hernández. En el primer punto, el que nos encontramos, tendremos una breve presentación del tema que se abordará en los siguientes puntos y en qué se basará el proyecto.

Respecto al punto 2, material y métodos, serán descritos los elementos y materiales que se han utilizado para el ensamblado del

robot, así como la explicación de los procesos que se han seguido para llegar al punto final del proyecto.

En cuanto al punto 3, se presentarán los resultados obtenidos a partir del punto 2, pudiendo validar el método propuesto en el artículo [1] que se ha basado esta memoria.

Relativo a la parte final de la memoria, se presentarán las conclusiones del proyecto, incluyendo futuras líneas de trabajo.

Por último, se tendrá en cuenta en el apartado de anexos los códigos utilizados en Arduino para el control del robot y las referencias en las que nos hemos basado para la redacción y estudio de la memoria.

2. Material y métodos

2.1 Definición de los componentes del sistema

Para el desarrollo del proyecto, en primer lugar, se han definido todos los elementos relevantes del sistema. Por lo tanto, se ha diferenciado el montaje por partes. En primer lugar, siguiendo la Figura 1, podemos observar las piezas necesarias y básicas del sistema. Entre ellas encontramos:

- Embrague electromagnético para montaje sobre el eje de la junta O, HUCO denomina a este tipo de embragues SO porque el montaje es en serie, nuestro modelo es el SO19, quedando sus características como se observa en la siguiente tabla:

Atributo	Valor
Tipo de orificio	10H9
Inercia del rotor (kg/cm²)	0,24
Par estático (Nm)	2,83
Masa (Peso) (kg)	0,34
Dimensión A, máx. (mm)	40,869
[B] Altura (mm)	33,376
Dimensión C (mm)	26,924

Tabla 1. Especificaciones embrague.

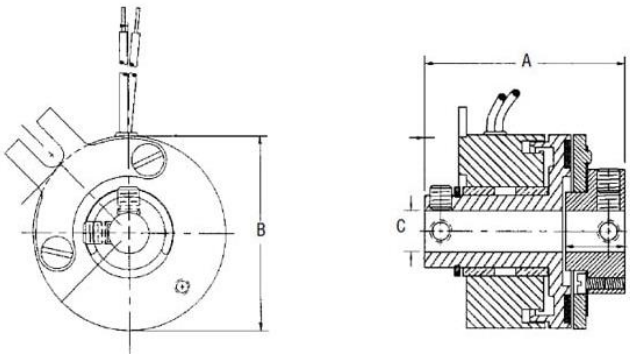


Figura 2. Plano embrague. [2]

La decisión de compra de este embrague viene dada debido a la necesidad de montar dos ejes en serie, por un lado, el eje del motor que se especificará a continuación, y por otro lado, el eje roscado para el montaje del robot. Además, este tipo de embrague ha sido elegido también porque dejará pasar el torque mecánico para hacer el movimiento normal de un sistema cuatro barras. Sin embargo, cuando deje de pasarlo la articulación O será pasiva, y con ello simularemos el fallo del robot, el FSJF, para así poder comprobar el método propuesto en el artículo [1].

- Motorreductor DC DOGA, con las siguientes especificaciones:

Atributo	Valor
Tensión de Alimentación	24 V dc
Tipo de Motor DC	Con caja reductora y con escobillas
Velocidad de Salida	100 rpm
Diámetro del Eje	10mm
Par de Salida Máximo	20 Nm, 3 Nm
Tipo de Cabezal de Engranajes	Cilíndrico con tornillo sin fin
Dimensiones	178 x 60 x 100,6 mm
Corriente Nominal	24 A, 3 A
Longitud	178mm
Anchura	60mm
Profundidad	100,6 mm
Construcción de la Bobina	Núcleo de Hierro

Tabla 2. Especificaciones motorreductor. [3]

Figura 3. Imagen del motorreductor.

La elección de este motorreductor para nuestro montaje se basa en la necesidad de dotar un par adecuado al sistema. Por lo tanto, una de las opciones que cumplía este requisito era este modelo. Además, mediante dicho motor se podrá controlar el ángulo de giro ϕ cuando se simule el fallo FSJF.

- Actuador lineal Actuonix P16-200-64-12-P, con las siguientes especificaciones:

Atributo	Valor
Fabricante	Actuonix Motion Devices Inc
Tipo	Actuador lineal
Tipo de motor	Con escobilla
Longitud de carrera	200 mm
Fuerza de carga (Dinámica)	90 N
Fuerza de carga (Estática)	500 N
Velocidad	18 mm/s
Ciclo de trabajo máximo	20%
Voltaje nominal	12 VCC
Construcción	Ball Screw
Precisión	±0.80mm
Ratio de reducción del engranaje	64
Tipo de retroalimentación	Potenciómetro
Tipo de terminación	Extremidad de hilo con conector
Temperatura de funcionamiento	-10°C ~ 50°C
Protección de entrada	IP54 - Resistente al polvo, resistente al agua
Número de producto base	P16-200

Tabla 3. Especificaciones actuador lineal. [4]

La elección de este actuador lineal viene dada por la necesidad de tener una longitud de carrera de unos 200 mm. Además, se parte el proceso de montaje con este elemento y será el elemento con el que controlará ρ, el valor de la longitud de carrera del actuador lineal.

Figura 4. Imagen del actuador.

- Rodamiento de bolas con diámetro interior y exterior de 10 y 30 mm, respectivamente y siguiendo las características mostradas a continuación:

Atributo	Valor
Diámetro de Entrada	10 mm
Diámetro de Salida	30 mm
Anchura del Anillo	9 mm
Tipo de Rodamiento de Bola	Rodamiento de bolas de ranura profunda de fila única
Disposición del sellado	Ambos lados sellados
Distancia de rodamientos	CN
Material	Acero
Material de Bola	Acero
Material de la Rejilla	Acero
Número de Filas	1
Valor Nominal de Carga Estática	2,36 kN
Valor Nominal de Carga Dinámico	5,07 kN
Tipo de Calibre	Paralelo
Series de cojinetes	2 ligero
Tipo de anillo	Plano
Material del Anillo	Acero
Código parcial de rodamiento	6200
Número de referencia del rodamiento	6200-2RS

Tabla 4. Especificaciones rodamiento. [5]

Figura 5. Imagen del rodamiento.

La decisión de estos rodamientos se debe a la necesidad de alojar el eje roscado en la pieza 1 – izquierda. Por ello, se dispondrá uno por la cara delantera de la pieza, y otro por la trasera.

- Potenciómetro de eje, siguiendo las características mostradas a continuación:

Atributo	Valor
Tipo de Potenciómetro	Giratorio
Resistencia Máxima	5kΩ
Número de Módulos	1
Potencia Nominal	2W
Curva de variación de resistencia	Lineal
Material de elemento	Plástico conductor
Tipo de Montaje	Montaje en Panel
Estilo de Terminación	Soldadura
Diámetro del Eje	8 mm
Tolerancia	±20%
Serie	NP32HS

Tabla 5. Especificaciones del potenciómetro. [6]

Figura 6. Imagen del potenciómetro.

La decisión de este tipo de potenciómetros viene dada por las especificaciones que se detallan en el proceso de montaje. Donde se detallará que se necesitaba un potenciómetro hueco para poder pasar las piezas 2 y 3 por ese orificio.

- Placa Arduino DUE, siguiendo las especificaciones que se detallan a continuación:

Atributo	Valor
Name	Arduino® Due
SKU	A000062
Microcontroller	AT91SAM3X8E
USB connector	Micro USB
Built-in LED Pin	13
Digital I/O Pins	54
Analog input pins	12
Analog output pins	2
PWM pins	12
CAN	Yes (ext. transceiver needed)
UART	Yes, 4
I2C	Yes
SPI	Yes
I/O Voltage	3,3V
Input voltage (nominal)	7-12V
DC Current per I/O pin (group 1)	9 mA
DC Current per I/O pin (group 2)	3 mA
Power Supply Connector	Barrel Plug
Total DC Output Current on all I/O lines	130 mA
Processor	AT91SAM3X8E 84 MHz
AT91SAM3X8E	96KB SRAM, 512KB flash
Weight	36 g
Width	53,3 mm
Length	101,5 mm

Tabla 6. Especificaciones Arduino DUE. [7]

Figura 7. Imagen de la placa Arduino DUE.

Respecto a la Arduino DUE, se utiliza en el montaje del sistema para la lectura y ejecución de los comandos necesarios en el

código final, que se detalla en posteriores puntos de la memoria.

- Servocontroladora ESCON 36/2

Atributo	Valor
Weight	30 g
DC motors up to	72 W
Without sensor (DC motors)	Yes
DC tacho	Yes
Digital incremental encoder (2-channel, single-ended)	Yes
Digital incremental encoder (2-channel, differential)	Yes
Digital incremental encoder (3-channel, differential)	Yes
Current controller	Yes
Speed controller (open loop)	Yes
Speed controller (closed loop)	Yes
Operating voltage Vcc (min.)	10 V
Operating voltage Vcc (max.)	36 V
Max. output voltage (factor * Vcc)	0.98
Max. output current Imax	0,166666667
Max. time of peak output current Imax	60 s
Continuous output current Icont	0,083333333
PWM clock frequency of power stage	53,6 kHz
Sampling rate PI current controller	53,6 kHz
Sampling rate PI speed controller	5,36 kHz
Max. efficiency	0,95
Max. speed (DC)	150000 rpm
Built-in motor choke per phase	300 µH
Encoder signals	A, A, B, B\
Max. encoder input frequency	1 MHz
Digital inputs	2
Functionality of digital inputs	Enable, enable CW, enable CCW, enable CW+CCW, enable + direction of rotation, stop, PWM set value, RC Servo set value, fixed set value
Analog inputs	2
Resolution, range, circuit	12-bit, -10…+10V, differential
Functionality of analog inputs	Set value, current limit, offset, speed ramp
Potentiometers	1
Functionality of the potentiometers	set value, current limit, offset, speed ramp, gain, IxR factor
Digital outputs	2
Functionality of digital outputs	ready, speed comparator, current comparator, commutation frequency

Analog outputs	2
Resolution, range	12-bit, -4…+4V
Functionality of analog outputs	current monitor, speed monitor, temperature, fixed value
Encoder supply voltage	+5 VDC, max. 70 mA
Auxiliary output voltage	+5 VDC, max. 10 mA
USB (full speed)	Yes
Status indicator "Ready"	green LED
Status indicator "Error"	red LED
Protective functions	current limit, overcurrent, excess temperature, undervoltage, overvoltage, voltage transients, short-circuits in the motor cable
Temperature – Operation (min.)	-30 °C
Temperature – Operation (max.)	45 °C
Temperature – Extended Range	+45…+81 °C, Derating: -0.055 A/°C
Temperature – Storage (min.)	-40 °C
Temperature – Storage (max.)	85 °C
Humidity (non-condensing) (min.)	0,05
Humidity (non-condensing) (max.)	0,9
Weight	30 g
Dimension (length)	55 mm
Dimension (width)	40 mm
Dimension (height)	16.1 mm
Mounting	Mounting holes for M2.5 screws
Installation program	ESCON Setup
Graphical User Interface	ESCON Studio
Operating system	Windows 10, 8, 7, Windows XP SP3

Tabla 7. Especificaciones ESCON 36/2. [8]

Figura 8. Imagen de la servocontroladora ESCON 36/2.

Mediante la servocontroladora ESCON 36/2 se recibirán datos de la Arduino DUE que se encargarán de controlar el actuador lineal y el motor.

2.2 Montaje del robot

Una vez definidos los componentes principales del sistema se pasa al ensamblado de las piezas para obtener el montaje del robot. De manera general, para la construcción de un robot el proceso a seguir es la planificación o diseño, que en el presente proyecto viene definido siguiendo el artículo [1]. Posteriormente, se adquieren los materiales y componentes básicos para el inicio, se continúa en el caso de este sistema con el ensamblado de las primeras piezas. Una vez con el esqueleto del sistema montado, se prueban los componentes electrónicos para comprobar el correcto funcionamiento habiendo programado y cableado según las necesidades del sistema, en el caso del robot paralelo sub-actuado RPRRR se detallarán a continuación.

Detallando el sistema presentado en la memoria se empieza por la definición de las medidas de las barras de la Figura 1. Para esta decisión, desde un principio se considera el aumentar dichas dimensiones, ya que en el artículo [1] están definidas como L1=0,06 m, L2=0,03 m y L3=0,02 m y son dimensiones que se quedarían demasiado pequeñas para el ensamblaje final. Por ello, inicialmente se decide escalar las medidas originales a 5:1, es decir, se multiplican por 5. Cuando las tenemos aumentadas al tamaño escalado, obtenemos que para la compra de los componentes necesarios para el sistema necesitaríamos grandes elementos y que, por consecuencia, aumentarían en gran manera el valor del peso del robot, por lo que finalmente se descarta esta escala.

Para continuar con la selección de las dimensiones que más se ajusten a nuestras necesidades, se diseña una pieza que acogerá al actuador lineal y a una varilla roscada que actuará como eje, será la articulación O. Esta pieza será clave en las dimensiones porque existirá una

diferencia de distancias entre el orificio del vástago y del orificio de la pieza diseñada.

En la siguiente imagen podemos observar el detalle de las medidas citadas, por facilidad de cálculos las fijamos en 2 cm, aproximadamente.

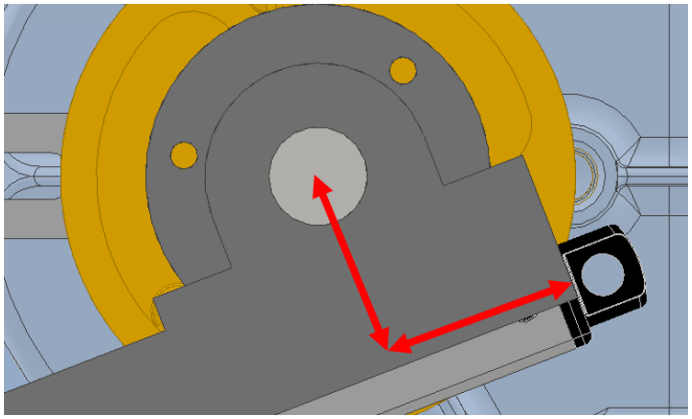


Figura 9. Imagen de la pieza de agarre del actuador lineal.

Una vez definidas estas medidas, se tiene que cuando el vástago del actuador lineal está completamente retraído y extendido es de unos 2 cm y 21 cm, respectivamente. Por lo tanto, la carrera del actuador vendrá dada por $21 - 2 = 19$ cm, siguiendo este resultado escalaremos el prototipo y, además, por facilidad de manufactura, aplicamos un coeficiente de 10, quedando la escala finalmente como 1,9:1. De esta manera nos quedarán las longitudes de las barras de la siguiente forma L1=0,114 m, L2=0,057 m y L3=0,038 m. Por otro lado, tenemos el valor de L4 que será ρ y variará según su longitud de carrera.

Estas piezas serán impresas en 3D y con el añadido de seguir los límites angulares que vienen dados por el artículo [1], donde el valor de los ángulos de las articulaciones D y C variará en el siguiente rango $-2{,}508\,rad \leq \vartheta_2 \leq 3{,}023\,rad$ y $-1{,}911\,rad \leq \vartheta_3 \leq 2{,}419\,rad$. Por ello, para que esos rangos sean los que se den en nuestro sistema, se implementará a las piezas topes mecánicos, que se encargarán de servir de freno a las barras que choquen con estos.

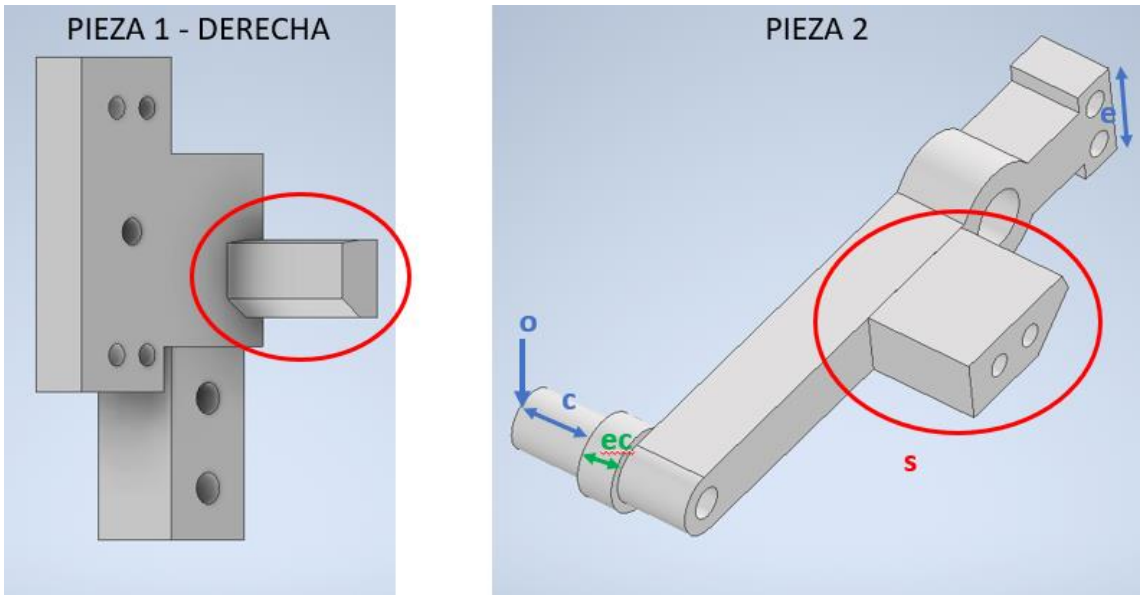


Figura 10. Imagen de los topes mecánicos en la barra 1 – derecha, articulación O, y la pieza 2, articulación D.

Una vez tenemos las piezas impresas en 3D y otros componentes descritos anteriormente, se da comienzo al ensamblaje. En el presente proyecto se da comienzo al montaje por la parte izquierda, parte donde tenemos el motor, embrague y actuador.

Para el comienzo del montaje, se necesita una base sobre la que iniciar el proceso, en este caso se hace uso de una base de metacrilato. Con el metacrilato se atornillan los 3 tornillos del motor, quedando este por la parte trasera del metacrilato y dejando un orificio en el metacrilato para el eje del motor. Cabe destacar que, para anclar el motor al metacrilato también se utilizó una especie de arandela impresa en 3D para separar el motor del metacrilato, ya que estos chocaban. Por la cara delantera del metacrilato se atornilla el embrague por un extremo y por el otro, la pieza diseñada como agarre del actuador para transmitir el giro al sistema, Figura 12.

Respecto al diseño de esta pieza, en su idea inicial surgen varios inconvenientes. En un primer momento y como se muestra en la figura 11, la pieza se diferencia en dos partes, la atornillada al embrague y la

que sería el agarre del actuador, pero este sistema generaba un juego que iba a proporcionar un error de lectura del giro debido a la poca precisión de la impresión 3D en este tipo de piezas.

Hay que destacar que las circunferencias verdes mostradas en la Figura 11 son donde se atornilló la pieza para transmitir el giro y esta constaba de dos orificios para tuercas, el rectángulo rojo para una tuerca hexagonal y el azul para una tuerca de freno, con un orificio circular.

Por otro lado, observamos otra pieza en la Figura 11, la de agarre del actuador lineal. En esta pieza se diseña una parte para alojar una tuerca hexagonal también, que se observa en el rectángulo lila. Entonces, en el hueco intermedio que queda entre la pieza atornillada al embrague y la pieza de agarre del actuador lineal se hará uso de otra tuerca hexagonal para evitar movimientos axiales.

Respecto a este montaje, se destaca que cuando lo montamos se produce el juego anteriormente dicho entre las piezas, el cual no estaba previsto. Por lo que se decide pasar a un rediseño de las piezas que no genere este problema.

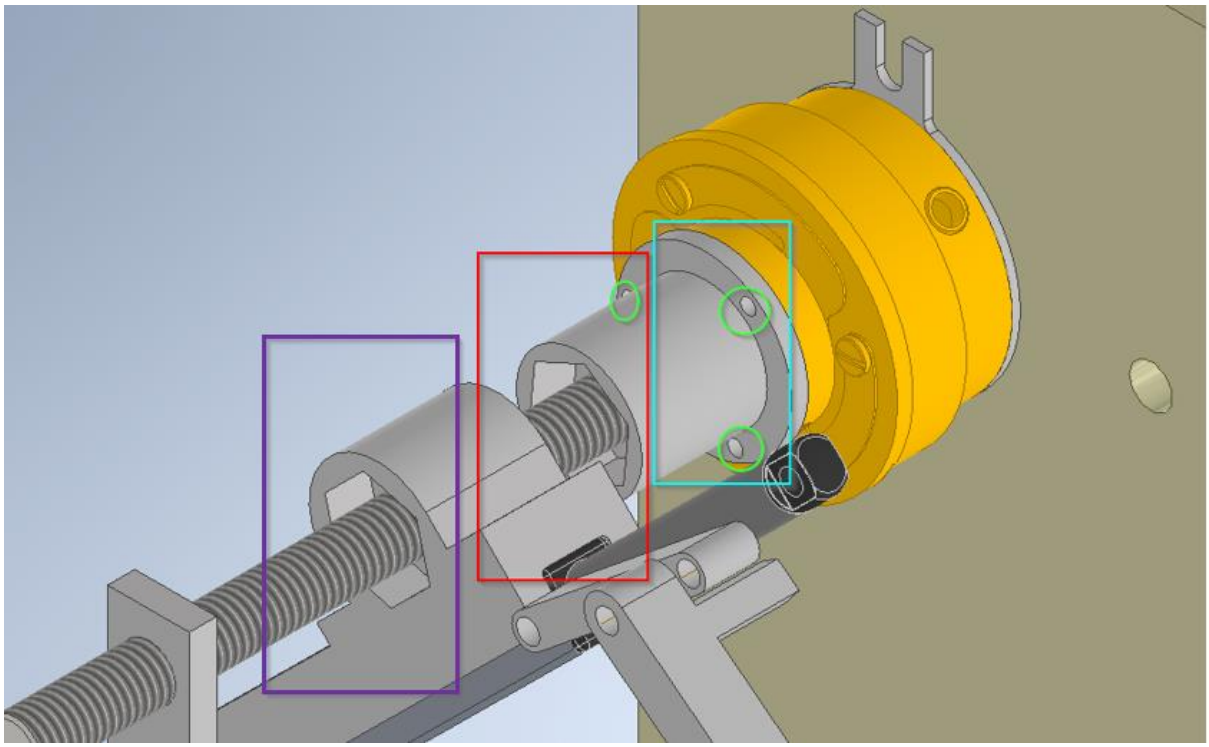


Figura 11. Imagen del agarre del actuador y pieza cilíndrica.

El rediseño de las piezas anteriores se basa en unir la pieza atornillada y el agarre del actuador, es decir, que de las dos piezas se pase a una, como muestra la Figura 12. Se fortalece el espesor de uno de los extremos, ya que debido a la forma del actuador se había diseñado ese extremo para insertar el plástico del actuador ahí, quedando muy fino. Siguiendo la misma línea, también se incrementa el espesor de las orejetas. Y, por último, en el óvalo rosa de la Figura 12 podemos observar la unión de las piezas, se ha eliminado los orificios para las tuercas hexagonales, pero se ha dejado el de la tuerca de freno para evitar movimientos indeseados, como se observa en la parte derecha de la siguiente figura. Quedando como se observa en la siguiente imagen:

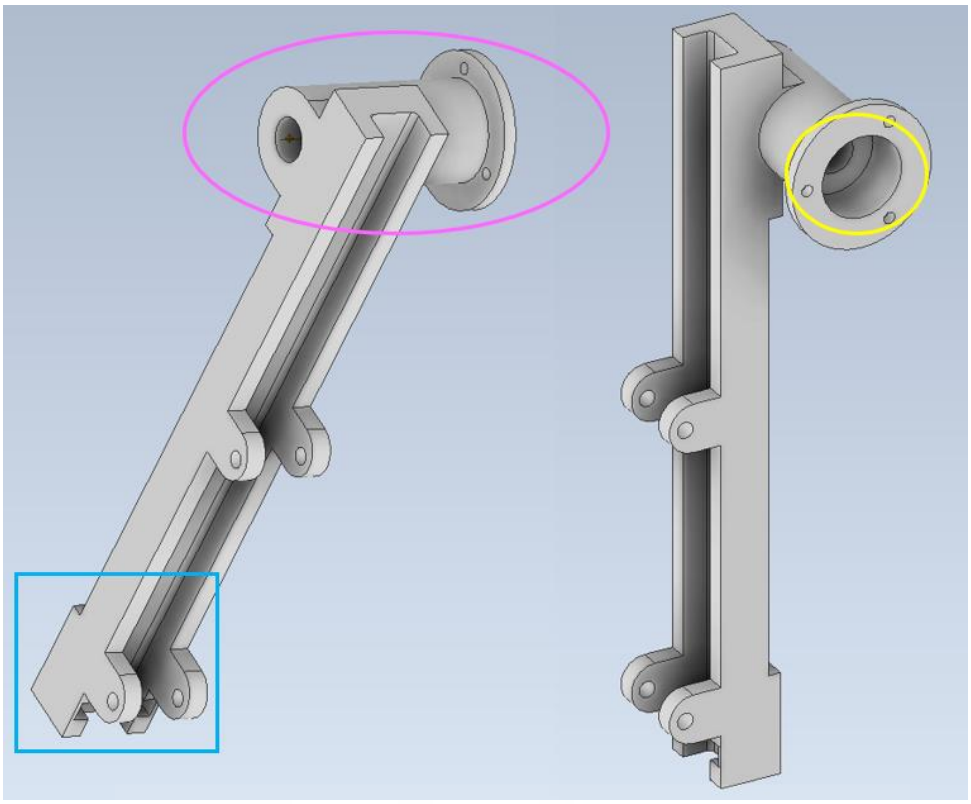


Figura 12. Imagen de la nueva propuesta de pieza de agarre.

Continuando con el montaje de la parte izquierda del sistema, se ha de tener en cuenta la parte del bastidor izquierdo, es decir, la articulación O de la barra 1. En este caso se diseñará y se imprimirá otra pieza en 3D en la que se alojará tanto por la cara delantera, como por la trasera un rodamiento 6200 – 2RS y tendrá un orificio que también incluirá al eje roscado, para evitar su movimiento axial se colocarán tuercas de freno, impidiendo de esta manera movimientos no deseados.

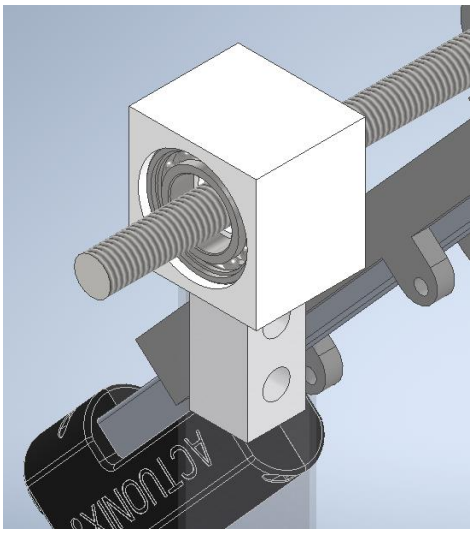


Figura 13. Imagen pieza 1 - izquierda.

Dicha pieza será atornillada a un tubo metálico hueco cuadrado para darle más rigidez al sistema y no imprimir una pieza muy larga mediante impresión 3D, ya que uno de los inconvenientes de este método es el tiempo que tarda en imprimir una pieza de grandes dimensiones.

Una vez tenemos ensambladas todas las piezas anteriormente dichas, queda de la siguiente manera:

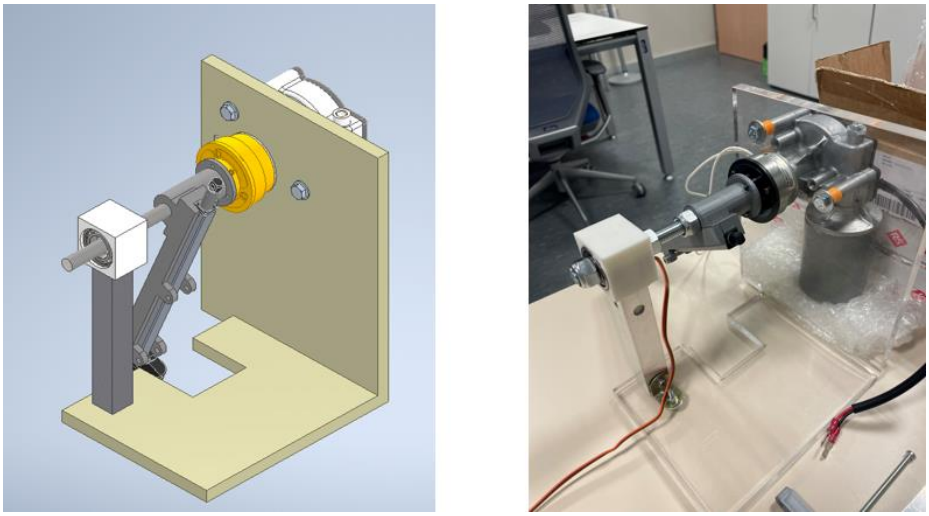


Figura 14. Imagen CAD y real de la parte izquierda del robot.

Para continuar con el montaje del robot, se procede a ensamblar la barra 3 al vástago del actuador lineal y, posteriormente la barra 2. Respecto al diseño de estas piezas, la barra 2 se puede observar en la Figura 10 y la barra 3 en la siguiente:

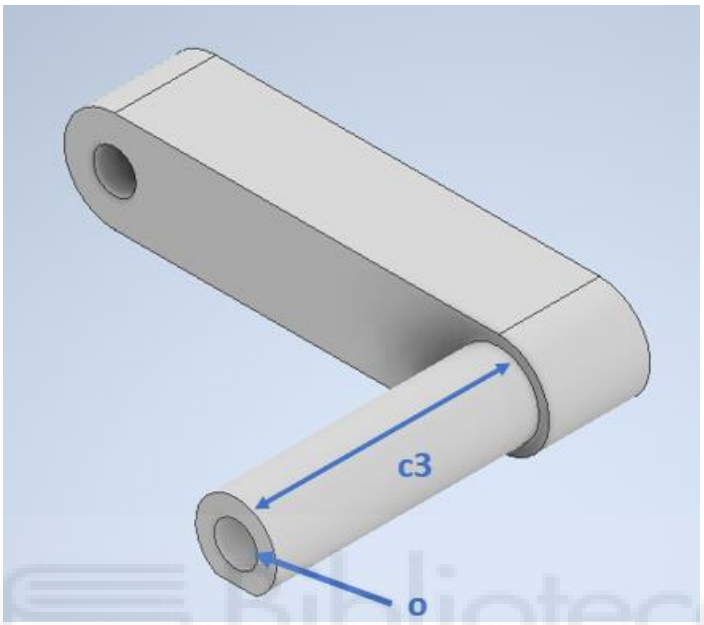


Figura 15. Imagen de la barra 3.

Hay que destacar, por un lado, que de estos diseños a los iniciales existen diferencias, es decir, en primera instancia no se pensaron de la manera en la que quedan finalmente debido a las necesidades que luego se vieron durante el ensamblado. En un primer momento, la barra 2, Figura 10 derecha, se diseñó como se comentaba anteriormente solo con el saliente, tope mecánico, y sufrió modificaciones como la separación (s) del tope mecánico a la propia barra, el ensanche (e) de uno de sus extremos para poder atornillar uno de los potenciómetros, que se usarán para medir el ángulo de giro de dicha barra. También se añade un orificio (o) al cual se le conectará un cilindro de la barra 3 y en el otro extremo, se incluirá un cilindro (c) que será el que se ajuste al orificio del potenciómetro, con la particularidad de que no será un cilindro perfecto, sino que tendrá la muesca con la forma del potenciómetro. Otro detalle de la barra 2 para tener en cuenta es el cilindro que envuelve (ec) al cilindro largo, se diseña de tal manera para que no interfiera y cause

errores en la medida leída del potenciómetro. Se puede observar el detalle de la pieza en la imagen 10.

Por otro lado, en cuanto a la barra 3, inicialmente se diseñó sin el cilindro (c3) que se observa en la imagen 15, pero de la misma manera que en la barra 2, se incluye con la forma del orificio del potenciómetro para leer el giro de la articulación.

El montaje de los potenciómetros se rige por las posiciones de los salientes cilíndricos de las barras 2 y 3. Se denominará potenciómetro 1 al que mide el giro ϑ_2 y potenciómetro 2 al que mide el giro ϑ_3. El potenciómetro 1 se atornilla a la pieza 1 derecha, punto fijo de la articulación D, por un lado, pero manteniendo una distancia mediante una calza impresa en 3D para evitar el roce que se produciría cuando gira esa parte del potenciómetro con la pared de la pieza y, por el otro lado, se atornilla el saliente cilíndrico de la barra 2 hasta el punto fijo de la articulación D, quedando entre medias el potenciómetro 1. Respecto al potenciómetro 2, se atornilla directamente a la barra 2, pero con el saliente cilíndrico de la barra 3 también atornillando hasta el centro del potenciómetro 2, quedando en línea el potenciómetro 2, la barra 2 y la barra 3. En la siguiente figura se puede observar cómo queda finalmente.

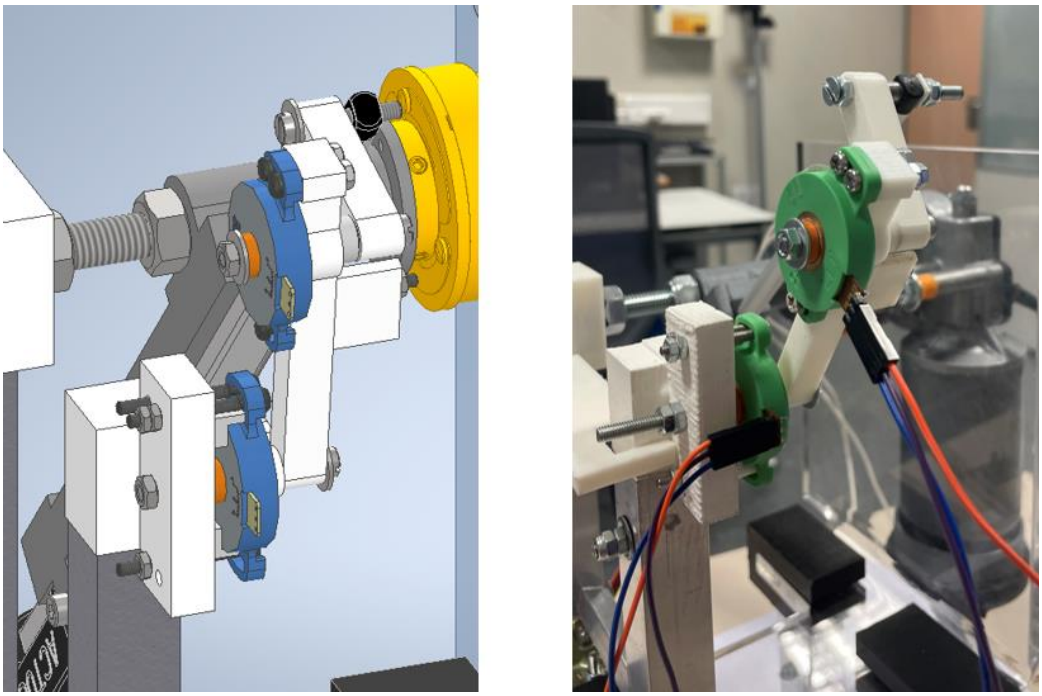


Figura 16. Imagen CAD y real de las barras 2 y 3 con los potenciómetros.

Es relevante resaltar el papel de los potenciómetros en las barras 2 y 3. La decisión de su colocación viene dada, por una parte, por la dificultad de medida en la parte izquierda del sistema, articulación O, ya que el eje que se usa en esta parte del sistema es de diámetro 10 mm, un valor muy grueso que dificulta el encontrar dispositivos que se adapten a esta medida. Y, por otra parte, por la generación de ambigüedad de dos posibles soluciones de la cadena cinemática BCD si solo se mide el ángulo de giro ϕ en la articulación O, además de ρ. Sin embargo, mediante trigonometría podemos obtener el ángulo de giro ϕ en la articulación O y evitar posibles errores que nos lleven a soluciones equívocas. Por ello, para la obtención de las ecuaciones que nos relacionan los giros medidos en ϑ_2, ϑ_3 y la longitud ρ con ϕ seguimos el siguiente proceso.

Cabe destacar que la idea inicial era centrarse en el robot mostrado en la Figura 1 de esta memoria, pero debido al montaje del sistema se decidió realizar unas pequeñas variaciones por facilidad de manufactura

en esta versión del prototipo En concreto, la principal diferencia del proyecto a la Figura 1 es los desfases a y b marcados en la Figura 17, que impiden que el actuador lineal ρ esté alineado con OB. En la siguiente figura se puede observar algunas de las variaciones de distancias y sus sistemas de coordenadas.

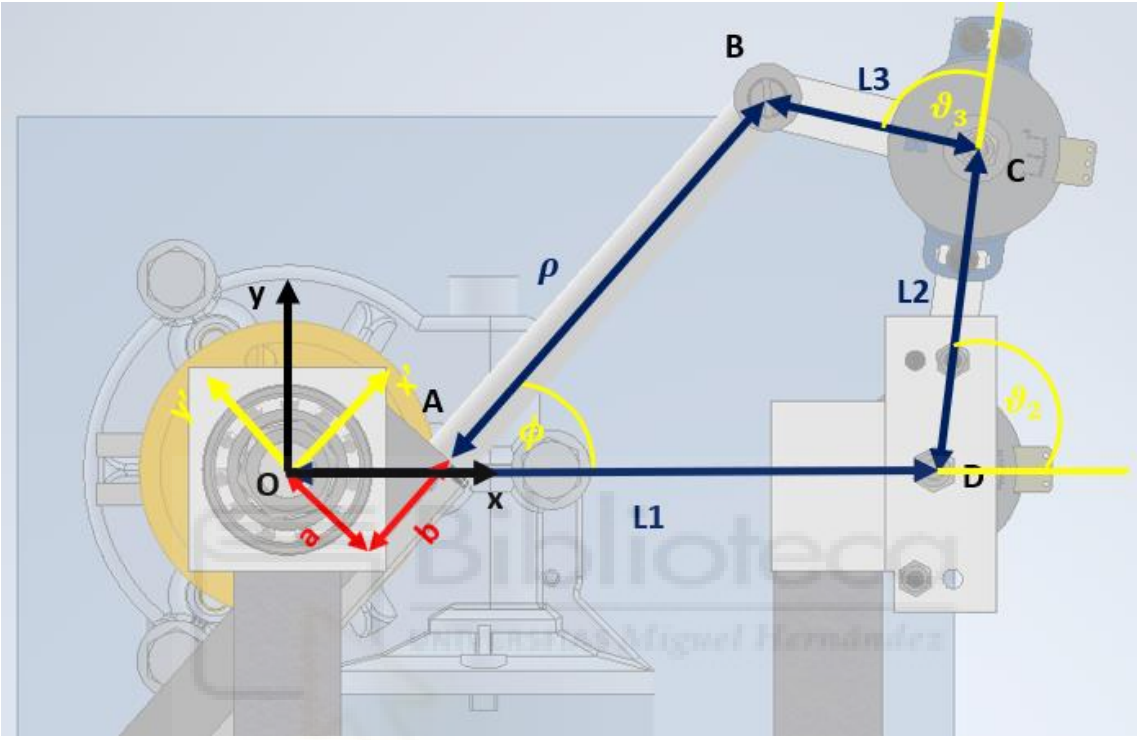


Figura 17. Imagen de la definición de los sistemas de coordenadas.

Siguiendo los sistemas de referencia de la imagen anterior, se procede a su definición analíticamente.

$$B_{xy} = \begin{pmatrix} L1 \\ 0 \end{pmatrix} + L2 \cdot \begin{pmatrix} cos\vartheta_2 \\ sin\vartheta_2 \end{pmatrix} + L3 \cdot \begin{pmatrix} cos(\vartheta_2 + \vartheta_3) \\ sin(\vartheta_2 + \vartheta_3) \end{pmatrix}$$

$$B_{x'y'} = \begin{pmatrix} b + \rho \\ -a \end{pmatrix}$$

Utilizando la matriz de transformación, se obtiene:

$$\begin{pmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} b\ +\ \rho \\ -a \\ 1 \end{pmatrix} = \begin{pmatrix} (b+\rho) * cos\phi\ + a * sin\phi \\ (b+\rho) * sin\phi\ - a * cos\phi \\ 1 \end{pmatrix}$$

Igualando esto último con B_{xy}:

$$(b+\rho) * cos\phi\ + a * sin\phi = L1 + L2 \cdot cos\vartheta_2 + L3 \cdot\ cos(\vartheta_2 + \vartheta_3)\ =\ K1$$

$$(b+\rho) * sin\phi\ -\ a * cos\phi = L2 \cdot sin\vartheta_2 + L3 \cdot\ sin(\vartheta_2 + \vartheta_3)\ =\ K2$$

$$\begin{pmatrix} b+\rho & a \\ -a & b+\rho \end{pmatrix} \begin{pmatrix} cos\phi \\ sin\phi \end{pmatrix} = \begin{pmatrix} K1 \\ K2 \end{pmatrix}$$

$$\begin{pmatrix} cos\phi \\ sin\phi \end{pmatrix} = \begin{pmatrix} b+\rho & a \\ -a & b+\rho \end{pmatrix}^{-1} \cdot \begin{pmatrix} K1 \\ K2 \end{pmatrix} = \begin{pmatrix} K3 \\ K4 \end{pmatrix}$$

Despejando ϕ, que es el ángulo de giro del motor obtenido mediante las ecuaciones anteriores, tendremos:

$$\phi = atan2(K4, K3) \qquad (1)$$

El cálculo de K3 y K4 queda:

$$\Delta = det = (b+\ \rho)^2 + a^2$$

$$\begin{pmatrix} K3 \\ K4 \end{pmatrix} = \frac{1}{\Delta} \cdot \begin{pmatrix} b+\rho & -a \\ a & b+\rho \end{pmatrix} \cdot \begin{pmatrix} K1 \\ K2 \end{pmatrix} \qquad (2)$$

$$K3 = \frac{1}{\Delta} \cdot ((b+\rho) \cdot K1\ -\ a \cdot K2)$$

$$K4 = \frac{1}{\Delta} \cdot (a \cdot K1\ +\ (b+\rho) \cdot K2)$$

De esta manera, se definen los parámetros K3 y K4, que posteriormente se incluirán en el código de Arduino y este los irá calculando, siguiendo las ecuaciones anteriores. De este modo, podremos estimar ϕ a partir de ϑ_2, ϑ_3 y ρ, sin tener que medir ϕ.

Respecto a la pieza 1 derecha, se observa en la Figura 10, y de la misma forma que la parte izquierda, se atornilla a un tubo metálico para proporcionarle estabilidad y rigidez. Ambos tubos metálicos se atornillarán también al metacrilato mediante escuadras.

Una vez tenemos también la parte derecha del sistema, estarían las partes principales ensambladas, pero aparece la necesidad de diseñar un soporte para el motor que mantenga la altura del metacrilato, debido a que dicho motor tiene un peso elevado e inclina todo el sistema. Por lo tanto, la pieza se diseña como cilindros que hacen de tope con las partes lisas del motor, quedando como en la siguiente imagen.

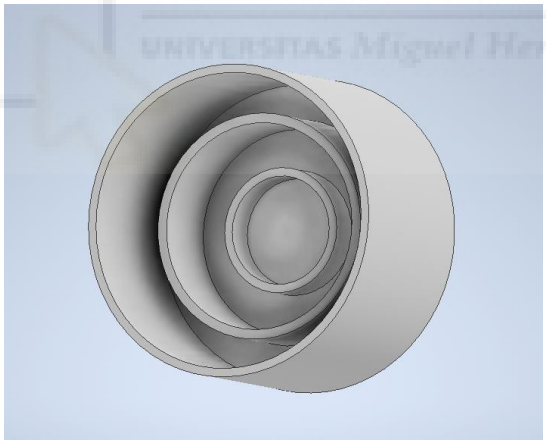


Figura 18. Imagen de la pieza soporte del motor.

Teniendo la pieza anterior, el sistema queda de la siguiente manera:

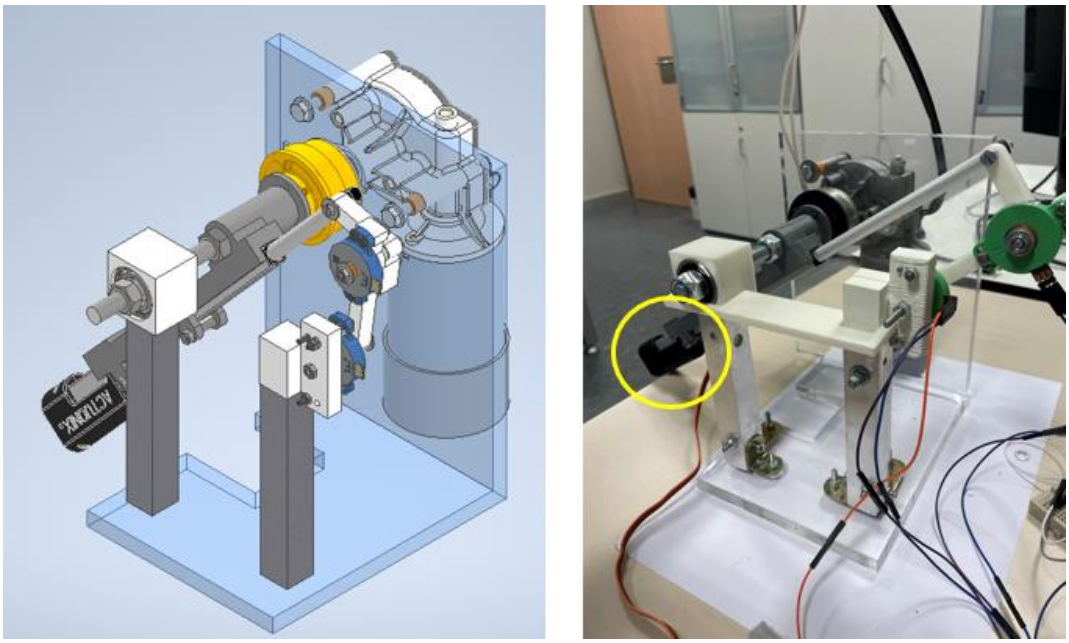


Figura 19. Imagen del sistema sin altura.

Sin embargo, nos damos cuenta de que, al estar directamente la pieza de metacrilato sobre una superficie plana, el extremo inferior del actuador no tiene espacio suficiente para moverse dentro de su rango, ya que choca con dicha superficie, ver Figura 19, resaltado amarillo. Debido al inconveniente anterior, se plantea poner unas patas que puedan dotar de altura al sistema y así, el actuador se pueda mover libremente sin colisionar con otros elementos. Dichas piezas serán 4 y se diseñan de una manera similar a las piezas 1 – derecha y 1 - izquierda, es decir, tendrán una parte que agarrará al metacrilato y se atornillarán a tubos metálicos. Sin embargo, se aprovechará la parte trasera, donde se sitúa el motor, para poner otra pieza diferente. Por lo tanto, tendremos 3 de las patas en el metacrilato y 1 que irá al motor.

Las piezas que van en el metacrilato se diseñan de manera que no haya que atornillarlas al sistema, por facilidad de manufactura. Por lo tanto, se proyectan 3 piezas de manera que hagan tipo pinza con el metacrilato. Sin embargo, aprovechando el soporte del motor se diseña otra pieza para que esta lo aloje. En la siguiente figura se puede ver el resultado final de dichas piezas.

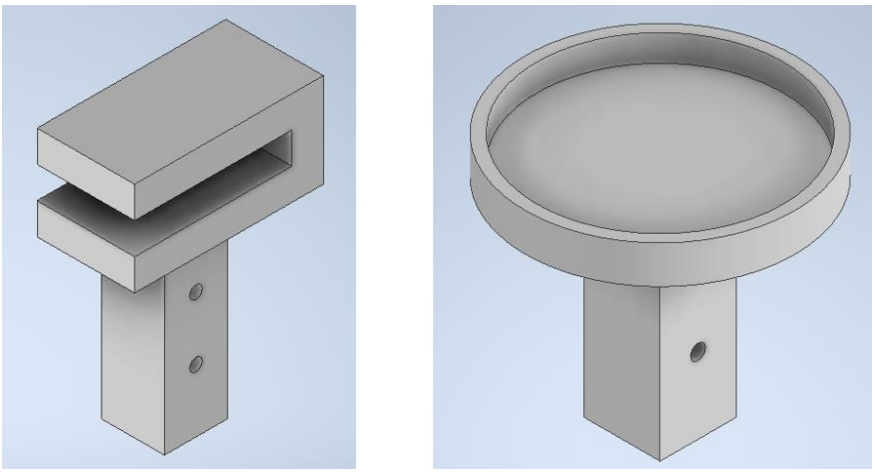


Figura 20. Imagen de las piezas que dan altura al sistema.

Quedando finalmente, una vez ensamblado y montado el robot, como en la figura siguiente. Donde se puede comprobar que se han añadido otras piezas para hacer el trabajo de zapata de la estructura, estas piezas son cuadradas y tienen un pequeño hueco para alojar al tubo metálico.

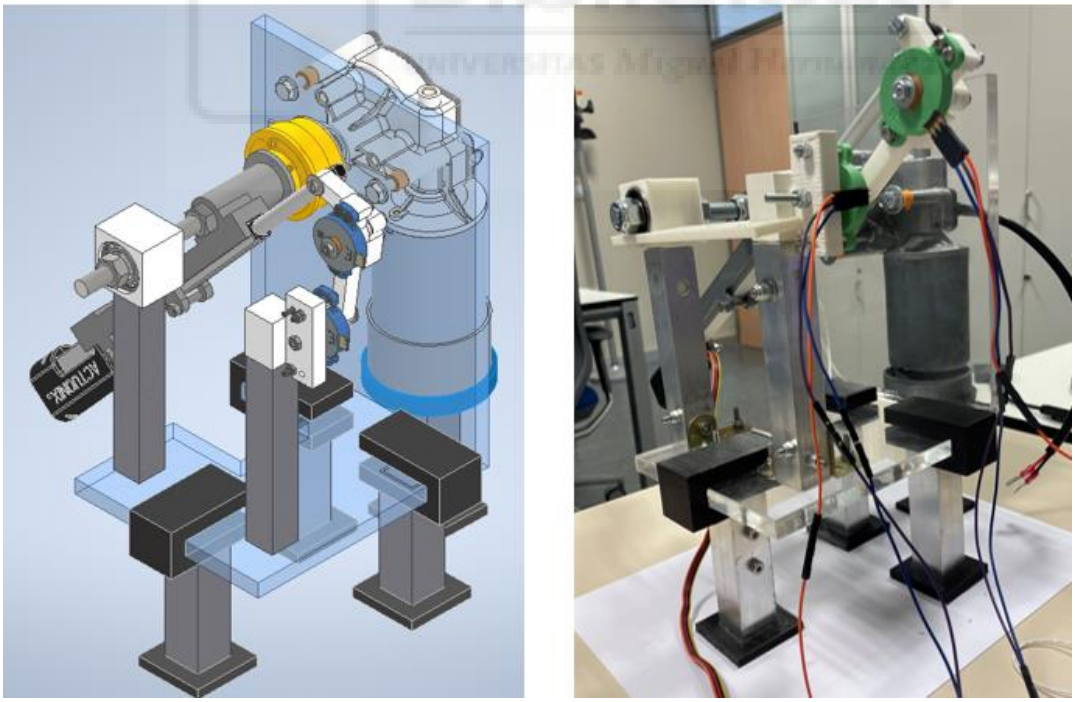


Figura 21. Imagen del robot con altura.

En el anexo 5.1 de la presente memoria se puede encontrar una lista de materiales, incluyendo los planos de las piezas impresas mediante impresión 3D en el anexo 5.3.

2.3 Cinemática directa e inversa del robot

En esta sección se resolverá la cinemática directa e inversa del robot, considerado como completamente actuado, y también como sub-actuado.

Cuando el robot es completamente actuado, los valores controlados por los actuadores son ρ y ϕ, que son las entradas cinemáticas, mientras que las salidas cinemáticas serían ϑ_2 y ϑ_3. Cuando se conocen ϑ_2 y ϑ_3 (medidas por los potenciómetros), la cinemática inversa consiste en calcular las entradas cinemáticas, ρ y ϕ. Ahora bien: dado que también estamos midiendo ρ, tan solo habría que calcular ϕ, pero esto ya se ha hecho con la ecuación (1).

Respecto a la cinemática directa, sería necesario distinguir entre el caso completamente actuado y el sub-actuado. En el caso completamente actuado, se conocen ρ y ϕ (que se eligen y controlan mediante sus respectivos actuadores), y por tanto se conoce la posición única del punto B. Conociendo B, se pueden resolver los ángulos ϑ_2 y ϑ_3 resolviendo la cinemática inversa de la cadena cinemática de tipo serie DCB, obteniendo dos soluciones: codo-arriba y codo-abajo. Dado que este problema es bien conocido en la literatura, y realmente nos interesa más el caso sub-actuado, no detallaremos las ecuaciones aquí.

En el caso sub-actuado, la cinemática directa cambia. Ahora se considera como entrada cinemática solo la longitud ρ, porque es la única que puede elegirse y controlarse directamente mediante su actuador lineal. El ángulo ϕ ahora es pasivo (al haber desacoplado este giro del motor mediante el embrague) y no queda determinado por su actuador respectivo. En esta situación, la cinemática directa se puede definir como

resolver todas las variables pasivas (ϕ, ϑ_2 y ϑ_3) conociendo el valor de ρ. Para ello, tomamos la ecuación (2), donde el lado izquierdo es (K3, K4) (que coinciden con el coseno y el seno de phi), mientras que el lado derecho involucra a ρ, y (ϑ_2, ϑ_3) (a través de K1 y K2). Si elevamos al cuadrado las dos filas de la ecuación (2) y las sumamos, el lado izquierdo quedará $\mathrm{K3}^2 + K4^2 = 1$, eliminando así la incógnita ϕ. En ese caso, tras desarrollar y simplificar la ecuación, queda:

$$1 = \left(\frac{L3 \cdot (b+\rho) \cdot \cos(\vartheta_2+\vartheta_3) - a \cdot L3 \cdot \sin(\vartheta_2+\vartheta_3) + L2 \cdot (b+\rho) \cdot \cos(\vartheta_2) - a \cdot L2 \cdot \sin(\vartheta_2) + L1 \cdot (b+\rho)}{a^2 + (b+\rho)^2}\right)^2 +$$

$$+ \left(\frac{a \cdot L3 \cdot cos(\vartheta_2+\vartheta_3) + L3 \cdot (b+\rho) \cdot \sin(\vartheta_2+\vartheta_3) + a \cdot L2 \cdot cos(\vartheta_2) + L2 \cdot (b+\rho) \cdot sin(\vartheta_2) + a \cdot L1}{a^2 + (b+\rho)^2}\right)^2$$

Multiplicando por el denominador y pasándolo todo al mismo lado de la ecuación, tras simplificar queda la siguiente:

$$0 = 2 \cdot L3 \cdot cos(\vartheta_2+\vartheta_3) \cdot (L2 \cdot \cos(\vartheta_2) + L1) + 2 \cdot L2 \cdot L3 \cdot \sin(\vartheta_2) \cdot \sin(\vartheta_2+\vartheta_3) +$$

$$+2 \cdot L1 \cdot L2 \cdot \cos(\vartheta_2) - a^2 - b^2 - 2 \cdot b \cdot \rho + L1^2 + L2^2 + L3^2 - \rho^2$$

Esta ecuación, para un valor fijo de ρ, define una o varias curvas en el plano (ϑ_2, ϑ_3). Estas curvas son llamadas en [1] como "variedades de balanceo libre" (o Free-swinging manifolds), debido a que, cuando fijamos el actuador lineal en un ρ dado, ϑ_2 y ϑ_3 son libres de moverse a lo largo de esas curvas (y tal movimiento será incontrolado, pues dependerá de factores como rozamiento, gravedad, fuerzas externas desconocidas...). El método presentado en [1] consiste en buscar los valores críticos de ρ, para los que esas curvas degeneran en puntos, de manera que el movimiento incontrolado de ϑ_2 y ϑ_3 (y, por extensión, de ϕ) queda limitado a dichos puntos. Si además el robot puede resistir cualquier fuerza externa en dichos puntos, obtenemos configuraciones de bloqueo estables. En el capítulo 3 mostraremos, para el robot construido, cuáles son las configuraciones de bloqueo en las que las curvas definidas por la ecuación anterior degeneran en puntos.

2.4 Electrónica y conexionado del sistema

Una vez tenemos preparado el montaje de las piezas del robot, se da paso a la electrónica de este. Como elementos adicionales a los anteriores, tenemos la placa de prototipos y la fuente de alimentación Tektronix PS280 DC Power Supply.

En primer lugar, para comprobar el correcto funcionamiento del actuador lineal, Figura 4, y de la placa Arduino DUE, Figura 7, se realizan diferentes pruebas. La primera consiste en un código básico donde se moverá el actuador lineal y se graficará en el eje X el tiempo y en el eje Y la variación de longitud de carrera del actuador lineal. Este código puede verse en el Anexo 5.2 “Código prueba 1”. Cabe resaltar que, esta prueba se realizó únicamente con la finalidad de dar tensión al actuador para comprobar su correcto funcionamiento.

Siguiendo con la segunda prueba, ahora incluimos la tarjeta servocontroladora ESCON 36/2 de la Figura 8. Para su conexión, inicialmente se ajustan los parámetros en el software ESCON STUDIO [9].

Una vez configurada la tarjeta pasamos al conexionado utilizando la placa Arduino, la servocontroladora, una placa de prototipos, la fuente de alimentación y el actuador lineal.

En esta prueba se añade al código Arduino que mediante una cadena de caracteres “A,B”, donde A será 1 para habilitar la servocontroladora y 0 la deshabilitará y B será la tensión a la que queremos que se mueva el vástago del actuador lineal, en bucle abierto. El código puede verse en el anexo 5.2 “Código prueba 2”. Recalcar que, respecto a la polaridad del cableado del actuador lineal, la ganancia sería negativa, ya que cuando se introducía por comando un valor negativo, el vástago del actuador se extendía y cuando se introducía un valor positivo, se retraía. Por lo tanto, se invertirá el cableado del actuador a la ESCON 36/2, ya que variaremos

la polaridad para que cuando pongamos valores negativos, se retraiga y cuando sean positivos, el vástago se extienda.

En cuanto al cableado, se ha añadido la servocontroladora con los detalles comentados anteriormente y con el añadido de que el pin blanco, número 2 digital, será utilizado para enviar tensión mediante entrada digital, el pin marrón, número 31 digital, para esa habilitación o deshabilitación de la propia servocontroladora y el gris a GND.

En cuanto a la prueba 3, seguimos por el camino de la segunda pero esta vez se varía el significado de B, que ahora servirá para fijar la posición a la que se busca que salga el vástago desde 0 hasta 1023. Con el añadido, en este caso, de los potenciómetros para poder medir los ángulos de giro ϑ_2 y ϑ_3, con todo el sistema montado.

Además, se añade un controlador PI, proporcional – integral, que ajustará y estabilizará al sistema. En este código se varía el valor de la tensión, es decir, se establece que el rango será de -11,7V a +11,7V para evitar posibles fallos y saturar la señal a ese voltaje.

Otro tema para tener en cuenta en esta prueba de código es la posición y calibración de los potenciómetros. Respecto al potenciómetro 1, que medirá ϑ_2, se tiene que su posición inicial, es cuando la barra 2 está completamente horizontal hacia la derecha. Por otro lado, en cuanto al potenciómetro 2, que mide el ángulo de giro ϑ_3, su posición inicial es cuando ϑ_3 en la Figura 1 vale 90°.

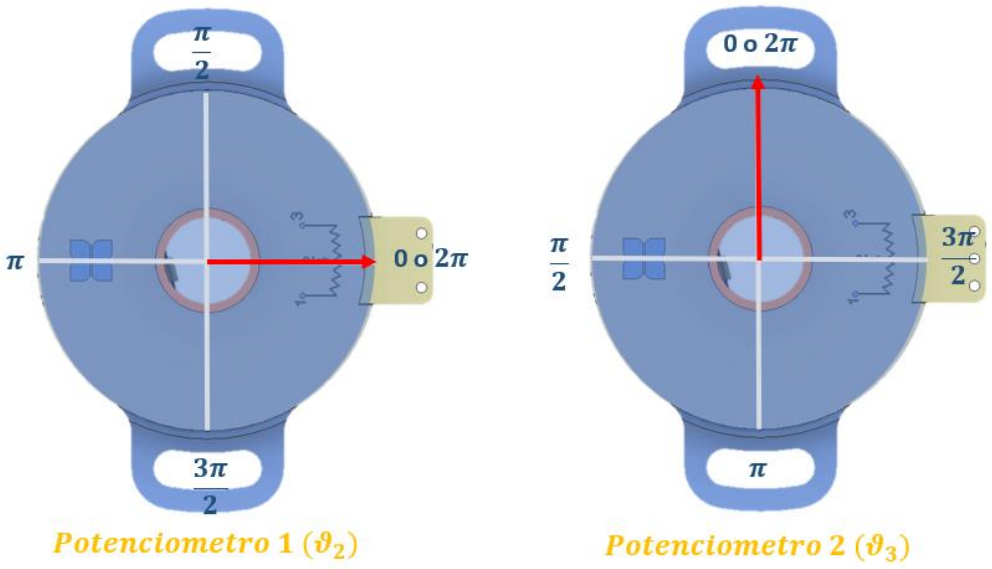


Figura 22. Imagen de las posiciones iniciales de los potenciómetros.

En este código de prueba, también se implementa el cálculo de ϕ definido anteriormente, que será dependiente de ϑ_2 y ϑ_3.

Como en el sistema se busca que el vástago se alargue hasta una posición deseada, se propone el uso de un controlador PI, como se comentaba precedentemente. Este controlador permitirá llegar a esa longitud deseada mediante la definición en Arduino de las necesidades y requerimientos del robot. Respecto a la constante proporcional e integral, se obtienen los valores mediante ensayo, empezando con valores de Kp más bajos y de Ki posteriormente. Finalmente, se fija en 2 y 0,0006 para la constante proporcional e integral, respectivamente. En este caso, se puede observar el código completo en el anexo 5.2 “Código prueba 3”.

En cuanto al cableado de este ensayo, se basa en el mismo que para el anterior, añadiendo los potenciómetros 1 y 2 para medir el ángulo de giro. Para la lectura de los valores, tenemos el pin wiper, que mostrará el giro que se quiere saber, el GND y el de la tensión de la placa Arduino de 3,3V. Respecto a la servocontroladora, se mantiene igual que en el ejemplo anterior.

Una vez se tiene todo lo anterior con su correspondiente conexión y el ensayo 3 realizado, se pasa al conexionado del embrague. Se ha de tener en cuenta que el embrague cuenta con una bobina electromagnética mediante la cual se realizará la simulación del fallo FSJF, a través del acoplamiento o desacoplamiento mecánico.

Inicialmente, se pensó en utilizar un relé para permitir o cortar el paso de corriente. Finalmente se hace uso de un cable que actuará de interruptor manual, es decir, el circuito dejará pasar la corriente cuando el cable esté conectado, pero cuando queramos que deje de pasarla, desconectaremos uno de los extremos del cable. Cuando el cable está conectado, se produce el efecto magnético, por lo que para simular el fallo FSJF, se debe desconectar.

En cuanto a los elementos necesarios para el uso del cable a modo de interruptor, se necesitará una resistencia de 1 kΩ y un led para que la bobina electromagnética del embrague descargue la corriente por esta parte del circuito.

En la siguiente imagen se puede observar el sistema descrito anteriormente. Por ello, los elementos a destacar del esquema son la fuente de alimentación, el cable que accionará o desaccionará el paso de corriente, la bobina del embrague, la resistencia y el led.

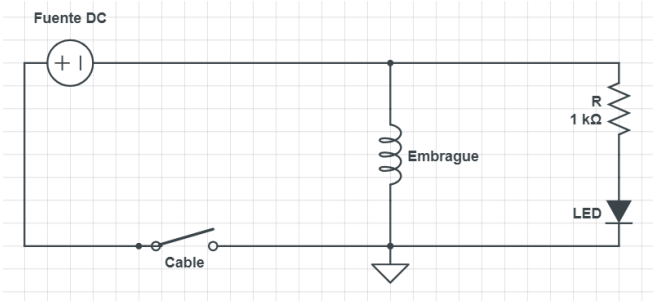


Figura 23. Imagen del circuito de descarga de corriente del embrague. [10]

Esta conexión del embrague se realiza para poder simular el fallo de FSJF, descrito al inicio de la presente memoria.

En cuanto al código final, se va a mantener el formato de las anteriores pruebas, es decir, se va a seguir enviando por consola el comando A,B con el añadido de C, que se va a establecer como el ángulo de giro ϕ. Por lo tanto, con este nuevo ensayo se ejecutará por consola la cadena de caracteres A,B,C. Por otro lado, B variará de pulsos a la medida en milímetros del vástago del actuador lineal, hasta 200 mm.

Para que C sea definido como ϕ hay que conectar el motor DOGA a todo el circuito ya montado anteriormente. El conexionado se realiza mediante otra ESCON 36/2, que también se accionará o desaccionará, utilizando 1 y 0 respectivamente, mediante el comando A. En primer lugar, se usará el cuadro J1 de la ESCON 36/2 para conectar la tarjeta a la fuente de alimentación Tektronix PS280 DC Power Supply. En segundo lugar, el cuadro J4 de la servocontroladora ESCON 36/2 para la conexión con el motor DOGA, habiendo probado antes de su conexión el sentido de giro, que ha resultado ser positivo cuando se da tensión positiva a los bornes. Por último, en la tarjeta servocontroladora se utilizará en el cuadro J5 de la ESCON 36/2 los cables blanco, pin 3 digital, marrón, pin 32 digital, y gris, GND, que nos servirán para enviar tensión al motor, habilitar o deshabilitar la tarjeta y utilizar el pin digital (PMW), respectivamente.

También se añadirá un controlador PI de manera análoga al actuador, pero en este caso para el motor, con constantes proporcionales e integrales con valores de 1 y de 0,001, respectivamente.

En cuanto al código definitivo con las características comentadas anteriormente, será expuesto en el cuerpo de la memoria a continuación.

```
#include <math.h>

// Definición de variables globales y constantes
```

```
const int pinPotenciometro = A0; // Pin al que está conectado el
potenciómetro
const int pinControlDriver1 = 31; // Pin para controlar el driver del
actuador lineal
const int pinControlDriver2 = 32; // Pin para controlar el driver del
motor
const int pinControlVoltaje = 2; // Pin para controlar el voltaje del
actuador lineal
const int pinControlMotor = 3; // Pin para controlar el voltaje del motor
const float longitudMaximaCarrera = 200.0; // Longitud máxima de la
carrera del actuador lineal
float valorDeseadoB = 0; // Valor deseado de B
float valorDeseadoC = 0; // Valor deseado de C
double L1 = 0.114; // Longitud barra 1
double L2 = 0.057; // Longitud barra 2
double L3 = 0.038; // Longitud barra 3
double a = 0.01975; // Longitud vertical desde la varilla roscada hasta
el inicio del vástago
double b = 0.0201; // Longitud horizontal desde la varilla roscada hasta
el inicio del vástago
double rho = 0; // Longitud de extensión del vástago del actuador lineal

// Definir los pines analógicos donde están conectados los potenciómetros
const int potPin1 = A1;
const int potPin2 = A2;
// Definir el número de grados por vuelta completa del potenciómetro
const int degreesPerRotation = 360;
// Definir el umbral para la zona muerta del potenciómetro
const int deadZoneThreshold = 51; // 5% de 1023
// Variable para almacenar la última lectura del potenciómetro 1 y 2
int lastPotValue1 = 0;
int lastPotValue2 = 0;

// Definición de variables para el control PI para longitud de carrera
const double Kp = 2; // Constante proporcional
const double Ki = 0.006; // Constante integral
double setpoint_carrera = 0; // Valor deseado del control PI para
longitud de carrera
double input_carrera, output_carrera; // Variables para el control PI de
longitud de carrera
double error_carrera, lastError_carrera; // Variables para el control PI
de longitud de carrera
double integral_carrera; // Variables para el control PI de longitud de
carrera

// Definición de variables para el control PI para phi
const double Kp_phi = 1; // Constante proporcional para phi
const double Ki_phi = 0.001; // Constante integral para phi
double setpoint_phi = 0; // Valor deseado del control PI para phi
```

```
double input_phi, output_phi; // Variables para el control PI de phi
double error_phi, lastError_phi; // Variables para el control PI de phi
double integral_phi; // Variables para el control PI de phi

unsigned long lastTime; // Variable para el control PI

bool printingEnabled = false; // Variable para controlar si se deben
imprimir los valores de PHI

void setup() {
  Serial.begin(9600); // Inicializar la comunicación serial
  pinMode(pinControlDriver1, OUTPUT); // Configurar el pin de la servo 1
como salida
  pinMode(pinControlDriver2, OUTPUT); // Configurar el pin de la servo 2
como salida
  pinMode(pinControlVoltaje, OUTPUT); // Configurar el pin del voltaje
del actuador lineal como salida
  pinMode(pinControlMotor, OUTPUT); // Configurar el pin del voltaje del
actuador lineal como salida

  lastTime = millis(); // Inicializar el tiempo para el control PI
}

void loop() {
  if (Serial.available() > 0) { // Si hay datos disponibles en el puerto
serial
    String input = Serial.readStringUntil('\n'); // Leer la cadena de
entrada desde el puerto serial
    processInput(input); // Procesar la cadena de entrada
  }

  // Lectura del valor del potenciómetro 1 y 2 (rango de 0 a 1023)
  int potValue1 = analogRead(potPin1);
  int potValue2 = analogRead(potPin2);

  float degrees1;
  // Verificar si se encuentra en la zona muerta
  if (abs(potValue1 - lastPotValue1) >= deadZoneThreshold) {
    // Si está dentro de la zona muerta, indicar que es la zona muerta
    // Serial.println("Zona muerta del potenciómetro 1");
    potValue1 = lastPotValue1;
    Serial.println("entro aqui");
  }
  else{
    degrees1 = potValue1 * 360.0 / 1023.0 + 90.0; //le suma 90 para
ajustar la posicion inicial del 0 a nuestro sistema, que es en la
horizontal
  }
```

```
  // Convertir los valores del potenciómetro a grados (rango de 0 a 360
grados)
  float degrees2 = potValue2 * 360.0 / 1023.0;

  // Convertir los valores del potenciómetro a radianes
  float theta_2 = degrees1 * PI / 180.0;
  float theta_3 = degrees2 * PI / 180.0;

  // Calculamos K3 y K4
  double resultadoK3 = calcularK3(theta_2, theta_3);
  double resultadoK4 = calcularK4(theta_2, theta_3);

  // Calculamos phi usando atan2(K4, K3)
  double phi = atan2(resultadoK4, resultadoK3);

  // Enviar el valor de phi a través de la comunicación serial si
printingEnabled es verdadero
  if (printingEnabled) {
    Serial.println(phi * 180.0 / PI);
  }

  // Ajustar los grados dentro del rango de 0 a 360 grados
  /*while (degrees1 >= 360) {
    degrees1 -= 360;
  }

  // Ajustar los grados dentro del rango de 0 a 360 grados
  while (degrees2 >= 360) {
    degrees2 -= 360;
  }*/

  // Verificar si se encuentra en la zona muerta
  if (abs(potValue2 - lastPotValue2) >= deadZoneThreshold) {
    // Si está dentro de la zona muerta, indicar que es la zona muerta
    // Serial.println("Zona muerta del potenciómetro 2");
  }

  // Actualizar la última lectura del potenciómetro 1 y 2
  lastPotValue1 = potValue1;
  lastPotValue2 = potValue2;

  // Lectura del valor del potenciómetro
  int lecturaPotenciometro = analogRead(pinPotenciometro);
  float longitudActualCarrera = map(lecturaPotenciometro, 0, 1023, 0,
longitudMaximaCarrera);

  // Actualización del setpoint del control PI para longitud de carrera
  setpoint_carrera = valorDeseadoB;
```

```
  // Cálculo del error para longitud de carrera
  error_carrera = setpoint_carrera - longitudActualCarrera;

  // Cálculo de la integral para longitud de carrera
  unsigned long currentTime = millis();
  double elapsedTime = (double)(currentTime - lastTime) / 1000.0; //
Convertir a segundos
  integral_carrera += (error_carrera * elapsedTime);

  // Cálculo de la salida del control PI para longitud de carrera
  output_carrera = Kp * error_carrera + Ki * integral_carrera;

  // Limitar la salida para longitud de carrera
  output_carrera = constrain(output_carrera, -11.7, 11.7);

  // Convertir la salida a un valor de voltaje B porcentaje para longitud
de carrera
  output_carrera = 15 + (70.0 / 24.0 * (output_carrera + 12)); //
Bporcentaje sin llegar a los limites del 10% y el 90%

  // Aplicar la salida al control del voltaje del actuador lineal para
longitud de carrera
  analogWrite(pinControlVoltaje, round(output_carrera * 255.0 / 100.0));

  // Igualar rho a la longitud de carrera en las unidades
correspondientes (m)
  rho = longitudActualCarrera / 1000;

  // Actualización del setpoint del control PI para phi


  // Cálculo del error para phi
  error_phi = setpoint_phi - phi * 180.0 / PI; // Convertir de radianes a
grados

  // Cálculo de la integral para phi
  integral_phi += (error_phi * elapsedTime);

  // Cálculo de la salida del control PI para phi
  output_phi = Kp_phi * error_phi + Ki_phi * integral_phi;

  // Limitar la salida para phi y saturarla
  output_phi = constrain(output_phi, -10, 10);

  // Convertir la salida a un valor de voltaje C porcentaje para phi
  output_phi = 15 + (70.0 / 48.0 * (output_phi + 24.0)); // phi
porcentaje sin llegar a los limites del 10% y el 90%

  // Aplicar la salida al control de la segunda servocontroladora
```

```
  analogWrite(pinControlMotor, round(output_phi * 255.0 / 100.0));

  //Actualiza el tiempo
  lastTime = currentTime;
}

// Función para procesar la entrada recibida a través de la comunicación
serial
void processInput(String input) {
  if (input == "PRINT") {
    printingEnabled = true; // Habilitar la impresión de valores de PHI
  } else if (input == "STOP") {
    printingEnabled = false; // Deshabilitar la impresión de valores de
PHI
  } else {
    int index = input.indexOf(',');
    if (index != -1) {
      String comandoA = input.substring(0, index);
      input = input.substring(index + 1);

      index = input.indexOf(',');
      if (index != -1) {
        String comandoB = input.substring(0, index);
        String comandoC = input.substring(index + 1);

        int valorComandoA = comandoA.toInt();
        float nuevoValorDeseadoB = comandoB.toFloat();
        float nuevoValorDeseadoC = comandoC.toFloat();

        // Limitar el valor de B a 185 si es mayor que 185
        valorDeseadoB = min(nuevoValorDeseadoB, 185.0);

        // Reiniciar el valor de la integral a 0 si se recibe un nuevo
comando de B,C
        integral_carrera = 0;
        integral_phi = 0;

        // Validación y control del comando A
        if (valorComandoA == 1) {
          digitalWrite(pinControlDriver1, HIGH);
          digitalWrite(pinControlDriver2, HIGH);
        } else if (valorComandoA == 0) {
          digitalWrite(pinControlDriver1, LOW);
          digitalWrite(pinControlDriver2, LOW);
        } else {
          // Serial.println("Error: Comando A debe ser 0 o 1");
          return;
        }
```

```
        // Asignar el valor de setpoint para phi
        setpoint_phi = nuevoValorDeseadoC;
      }
    }
  }
}

// Función para calcular K1
double calcularK1(double theta_2, double theta_3) {
    return L1 + L2 * cos(theta_2) + L3 * cos(theta_2 + theta_3);
}

// Función para calcular K2
double calcularK2(double theta_2, double theta_3) {
    return L2 * sin(theta_2) + L3 * sin(theta_2 + theta_3);
}

// Función para calcular K3
double calcularK3(double theta_2, double theta_3) {
    double K1 = calcularK1(theta_2, theta_3);
    double K2 = calcularK2(theta_2, theta_3);
    return ((b + rho) * K1 - a * K2);
}

// Función para calcular K4
double calcularK4(double theta_2, double theta_3) {
    double K1 = calcularK1(theta_2, theta_3);
    double K2 = calcularK2(theta_2, theta_3);
    return (a * K1 + (b + rho) * K2);
}
```

Siguiendo la estructura del código, se tiene la siguiente distribución.

1. Definición de variables y constantes:

   Por un lado, es donde se establecen los pines en los que se conectan los diferentes elementos del sistema y, además, se definen otros parámetros como longitudes de barras del robot. Por otro lado, también se crean las variables para almacenar otros valores como la longitud de carrera del actuador o el ángulo de giro de las barras.

2. Configuración inicial (*Setup*):

Se inicia la comunicación serial y se configuran los pines como salidas para controlar dichas salidas sobre el actuador lineal y el motor.

3. Bucle principal (*Loop*):

   En primer lugar, se verifica que haya datos disponibles para la lectura de estos. Una vez leídos los datos, se convierten a las unidades que buscamos, a milímetros la longitud de carrera y a grados el ángulo de giro de los potenciómetros 1 y 2. Dentro de esta parte del código también se calcula ϕ siguiendo las ecuaciones definidas anteriormente. Debido a la zona muerta, en los potenciómetros existen saltos entre los valores de su lectura que pueden llevar a error, por lo que se marca en el código una serie de líneas para evitarlos. Además, están definidos los controles PI tanto del actuador lineal, como del motor.

   Para la definición de los controladores se sigue la siguiente fórmula $Output = K_p \cdot error + K_i \cdot error\ integral$ y el error vendrá dado por la diferencia entre el valor deseado y el valor actual, mientras que el error integral es el error acumulado en el tiempo.

4. Procesamiento de entrada serial (*Processinput*):

   En este fragmento del código, se interpreta los comandos recibidos, es decir, activar o desactivar el *print* de valores mediante la ejecución de *PRINT* o *STOP*, actualiza los valores deseados de B y C, longitud de carrera y ϕ, y reinicia las integrales de los controladores al recibir nuevos comandos, para evitar la acumulación de errores.

5. Funciones auxiliares para cálculos trigonométricos

   En la definición de ϕ se usan valores como K1, K2, K3 y K4, por lo tanto, es necesaria la definición de estos parámetros en el código para los cálculos.

2.5 Configuraciones de bloqueo

Como se citaba anteriormente, las configuraciones de bloqueo son las posiciones que se buscan en el robot para cumplir con la seguridad. Estas posiciones vendrán descritas en el artículo [1] en el punto 3.1.

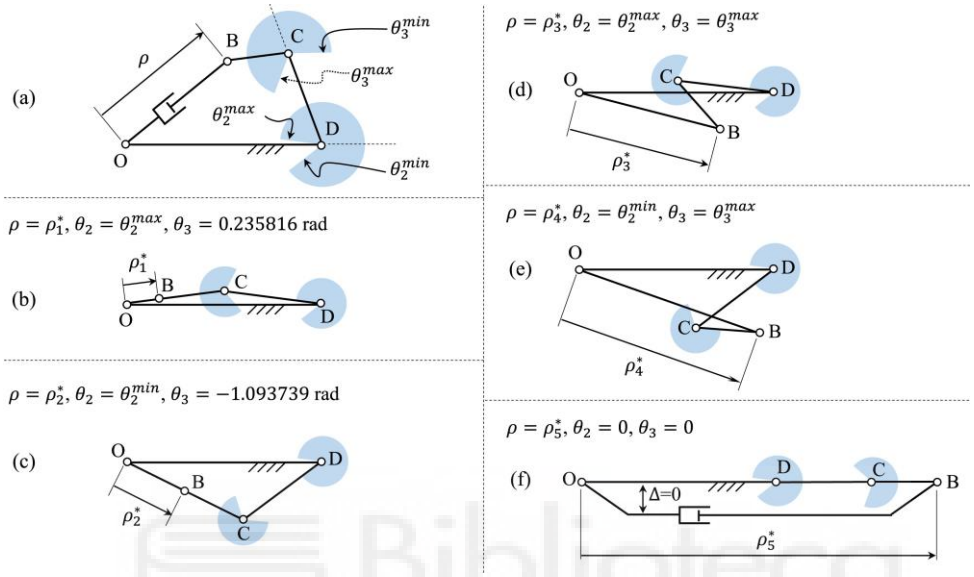


Figura 24. Imagen de las configuraciones de bloqueo y límites de giro reproducido de [1] con permiso.

En la imagen a se observa el sistema con los límites de giro de los ángulos ϑ_2 y ϑ_3. Por lo tanto, se tiene que el rango de giro de los ángulos será de $-2{,}508\,rad \leq \vartheta_2 \leq 3{,}023\,rad$ y $-1{,}911\,rad \leq \vartheta_3 \leq 2{,}419\,rad$. En las imágenes siguientes sí que se puede ver las cinco configuraciones de bloqueo, hay que detallar que el desfase que se ve en la imagen f de $\Delta = 0$ es una distancia que no debería existir, pero se añade por clarificar la imagen.

En la siguiente tabla vamos a observar los valores teóricos que se obtiene en cada una de las cinco configuraciones de bloqueo, respecto a ρ, ϑ_2 y ϑ_3.

Configuración	Valor artículo (m)	θ_2 (rad)	θ_3 (rad)
ρ_1	0,01041	θ_2 max	0,235816
ρ_2	0,019983	θ_2 min	-1,093739
ρ_3	0,044998	θ_2 max	θ_3 max
ρ_4	0,059068	θ_2 min	θ_3 max
ρ_5	0,11	0	0

Tabla 8. Valores teóricos de ρ, ϑ_2 y ϑ_3.

Otro factor importante que se puede observar en el artículo [1] es que las configuraciones de bloqueo de las figuras b, c y f son inestables, es decir, son posiciones en las que, en ocasiones, el sistema no va a poder mantener un equilibrio o funcionamiento deseado. En dicho artículo, en el apartado 5.1 se puede observar detenidamente el cálculo y la justificación de la estabilidad de las configuraciones.

Por otro lado, en la siguiente imagen se puede observar las trayectorias de movimiento libre de los ángulos ϑ_2 y ϑ_3 muy cerca de las configuraciones de bloqueo del robot original. Lo que se tiene en la imagen es que al ir aumentando el valor de ρ se van obteniendo los diferentes puntos de corte, de las curvas con el rectángulo interior de cada imagen, que establecerán las configuraciones de bloqueo. Los rectángulos interiores son los límites articulares para cada articulación, como se expresa en la imagen c de la figura.

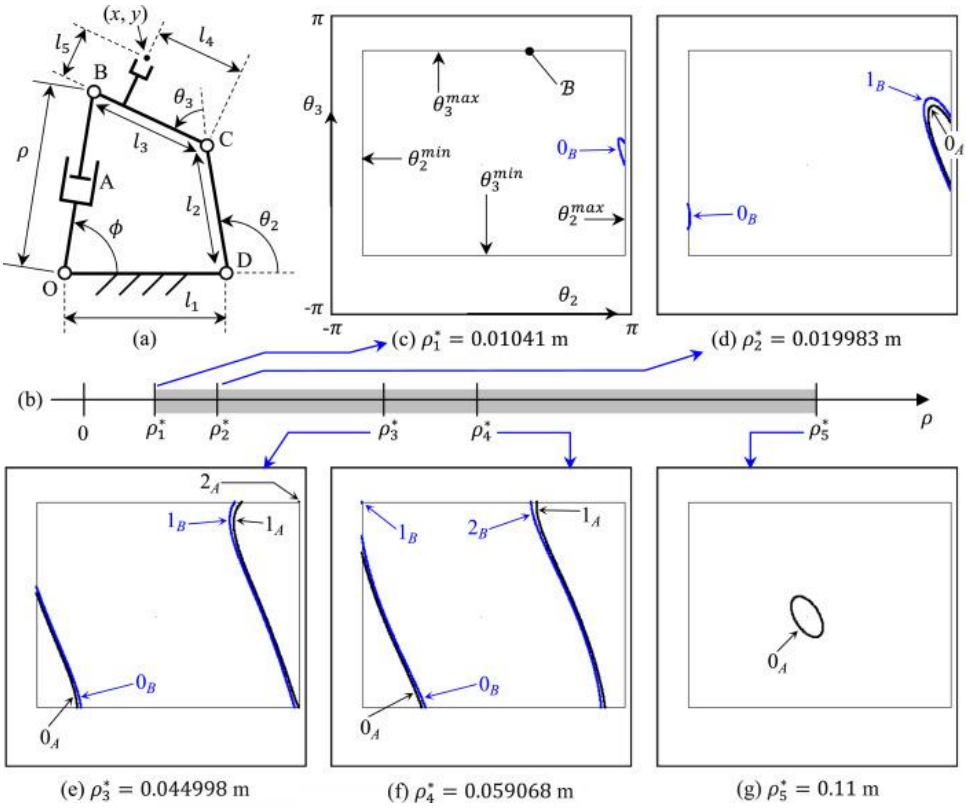


Figura 25. Imagen del espacio de las articulaciones en las configuraciones de bloqueo del robot original reproducido de [1] con permiso.

Es importante resaltar que, existe una diferencia de distancias en el punto O, se puede observar con detalle en la Figura 17, este desfase viene dado por los requerimientos y necesidades de montaje del sistema. Por ello, como se comentaba en otros puntos de la memoria, no es un prototipo a fiel escala, pero sí se basa generalmente en el robot descrito en el artículo [1]. Por ello, en los siguientes apartados se detallará y se comparará los resultados obtenidos con el prototipo realizado en el presente proyecto.

3. Resultados

Una vez montado y cableado todo el sistema, se procede a la búsqueda de las posiciones de bloqueo del sistema. Es importante destacar que debido al desfase que tenemos entre la varilla roscada y el actuador lineal, las configuraciones de bloqueo del prototipo del robot no coincidirán con las del artículo [1], pero sí serán similares, como se ha ido comentando en la memoria del proyecto.

Por lo tanto, en el caso del sistema descrito en el proyecto se calculan las configuraciones aplicando el método descrito en [1] a partir de las ecuaciones que definen las curvas ϑ_2 y ϑ_3 para un ρ dado, tal y como se han presentado en la sección 2.3.

Además, en la Figura 26 se muestra cómo queda el equivalente a la figura 25 para el prototipo construido, que es similar a la 25, pero se pierde la configuración de bloqueo inestable inicial, quedando solamente las de la tabla 9.

En la siguiente figura, se observa desde la iamgen a hasta la d las configuraciones de bloqueo, donde en la a se aprecia una curva que degenerará en un punto y en las siguientes sí se observa el punto de configuración. Ademas también se puede comprobar la similitud con las curvas del robot original.

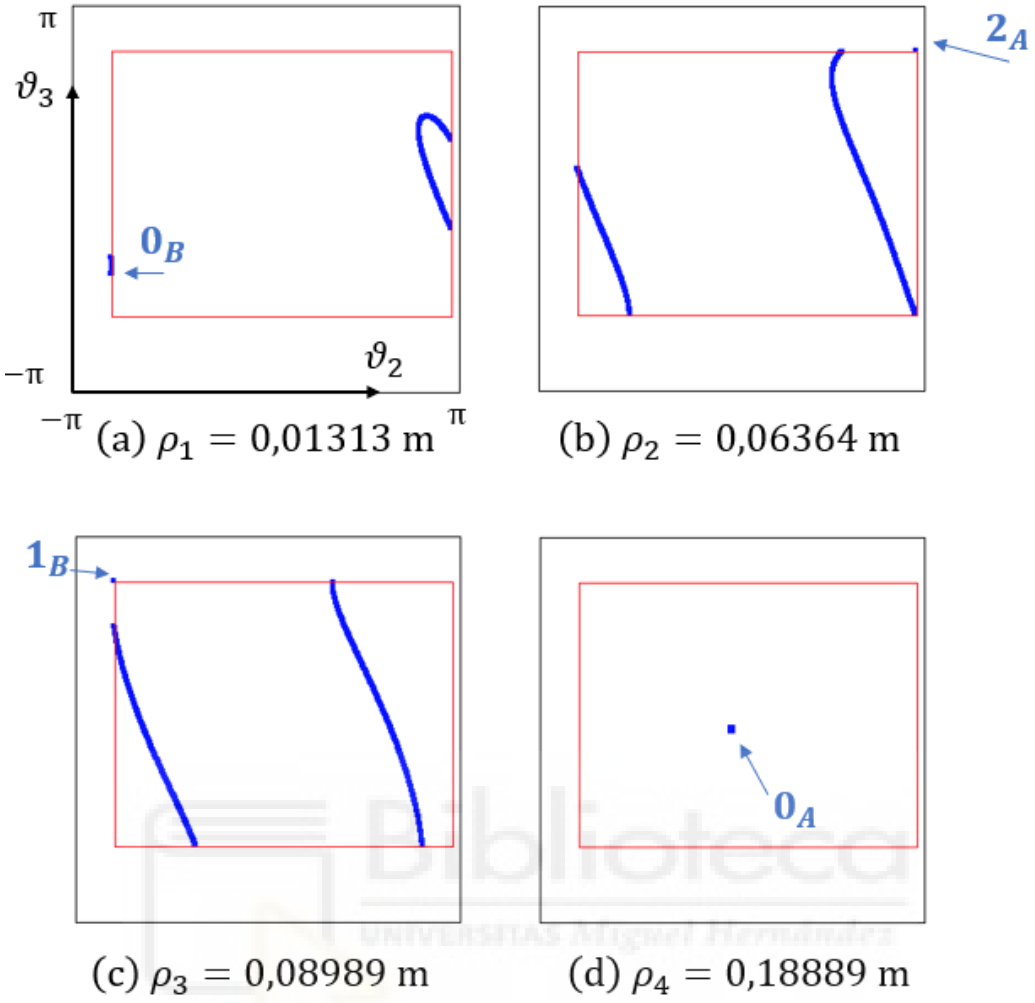


Figura 26. Imagen del espacio de las articulaciones en las configuraciones de bloqueo del prototipo de robot.

Configuración	Valor (mm)	θ_2 (rad)	θ_3 (rad)
ρ_0	-	-	-
ρ_1	13,131	θ_2 min	-1,1
ρ_2	63,636	θ_2 max	θ_3 max
ρ_3	89,899	θ_2 min	θ_3 max
ρ_4	188,889	0	0

Tabla 9. Valores para el prototipo de ρ, ϑ_2 y ϑ_3.

Observando los valores de la tabla, se puede comprobar que la configuración de bloqueo 0, la que sería la 1 del ejemplo original, desaparece, es decir, no es una posición de bloqueo para el robot con las características presentadas en la memoria. Por ello, el sistema

contará con las cuatro siguientes posiciones de bloqueo de la tabla. Respecto a la estabilidad de las posiciones, las configuraciones 1 y 4 seguirán siendo inestables, de manera análoga a los resultados en las configuraciones de bloqueo 2 y 5 obtenidas de la tabla 8, el sistema sin desfases.

Respecto a las configuraciones de bloqueo del prototipo del robot, en las imágenes siguientes se tiene de manera visual las posiciones de cada uno de los elementos en dichas configuraciones de bloqueo.

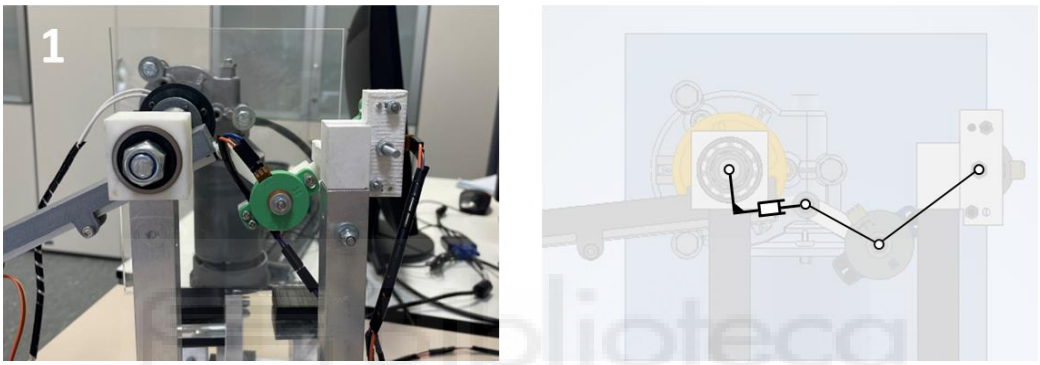


Figura 27. Imagen de la configuración de bloqueo 1 para el prototipo de robot.

En la anterior figura se tiene la imagen real, donde la longitud del vástago, ρ, es la marcada en la tabla 9, 13 mm. Por otro lado, respecto a ϕ se desenchufa la alimentación del embrague electromagnético, para simular el fallo de FSJF. Por lo tanto, la cadena de caracteres A,B,C ha sido 1,13,45, aunque el comando C no intervendrá, ya que como se ha citado anteriormente estará desenchufado. Además, al lado de la imagen real, también se observa la figura alámbrica del robot por clarificar las posiciones de los elementos en la configuración de bloqueo.

Siguiendo el mismo proceso anterior, se ejecutan las diferentes cadenas de caracteres A,B,C para cada ρ calculado, quedando como se puede observar en las siguientes imágenes.

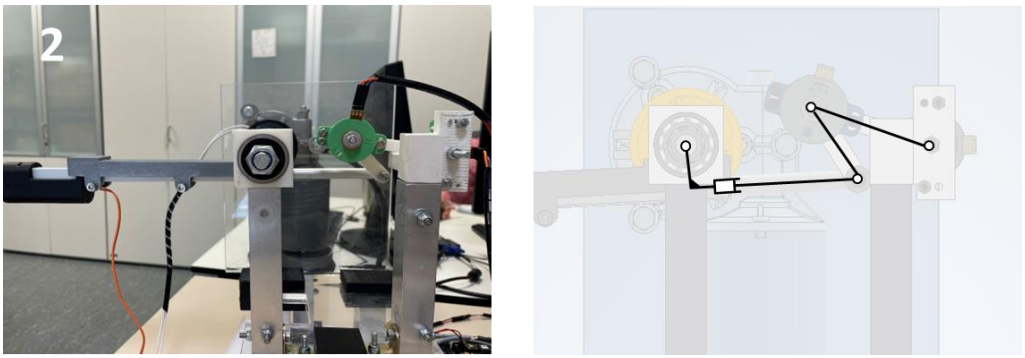


Figura 28. Imagen de la configuración de bloqueo 2 para el prototipo de robot.

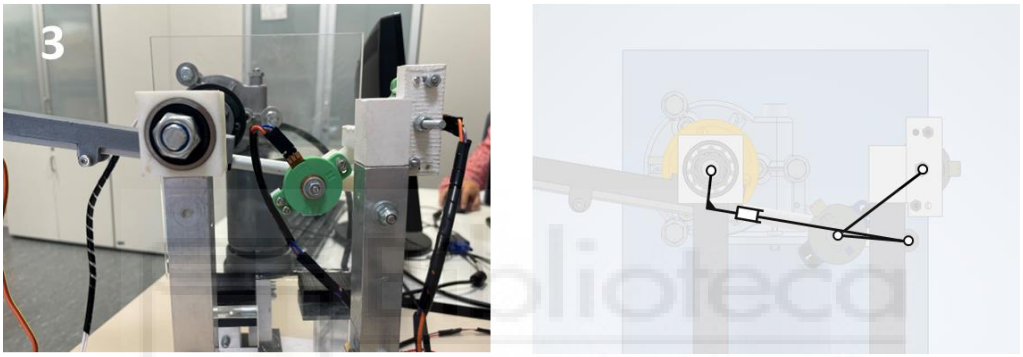


Figura 29. Imagen de la configuración de bloqueo 3 para el prototipo de robot.

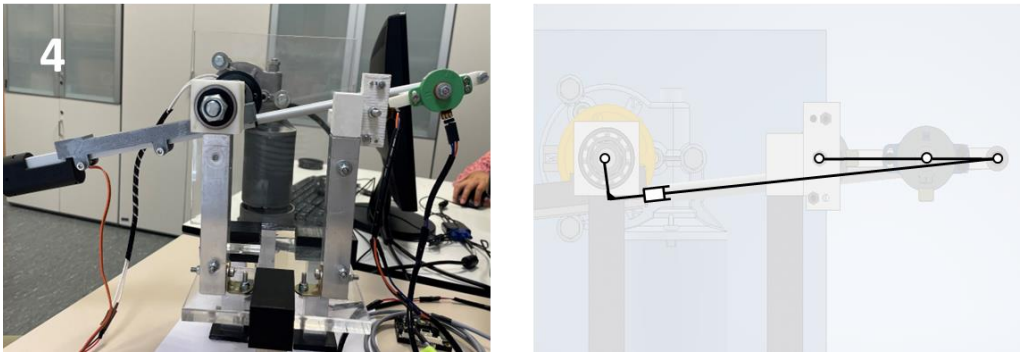


Figura 30. Imagen de la configuración de bloqueo 4 para el prototipo de robot.

Es importante resaltar que, para la configuración de bloqueo 4 como se puede observar en la figura anterior, no se ha llegado a su posición de ρ

exacta. Todo ello debido a que, cuando se extendía demasiado el vástago del actuador lineal, la falta de rigidez del sistema provocaba que se deformara la estructura. La diferencia entre la imagen real y la imagen generada en CAD es que como se comentaba anteriormente, la imagen real no corresponde con la longitud de vástago deseada en la configuración de bloqueo por problemas estructurales del sistema. Sin embargo, en la imagen CAD sí se podría observar las posiciones de los elementos para dicha configuración de bloqueo.

4. Conclusiones

Para concluir con el proyecto final de máster, se ha llevado a cabo la puesta en práctica del montaje de un robot paralelo sub-actuado RPRRR para simular el fallo generado por el *free swing joint failure*, *FSJF*. Por ello, el objetivo principal fue el diseño, realización, ensamblaje y programación del sistema descrito en el punto 4 del artículo [1]. Los resultados obtenidos cumplen con lo descrito en dicho artículo, pero con la diferencia del desfase generado en O con el actuador lineal.

Concretamente, el robot se comportó de igual manera en las configuraciones de bloqueo del sistema descrito en el artículo mencionado y en el prototipo, con la diferencia de que la configuración 1 de la tabla 8 desaparece debido al desfase descrito anteriormente. Por ello, se tiene que existen 4 configuraciones de bloqueo con las características descritas en la tabla 9. Sumado a lo anterior y, cumpliendo con los resultados del artículo respecto a la estabilidad de cada configuración, se cuenta con que las disposiciones 1 y 4 son las inestables y las 2 y 3 estables, proporcionando de esta manera un movimiento indeseado en las inestables.

Los resultados obtenidos confirman que el método descrito en el artículo [1] es aplicable en los robots que cumplan las características necesarias para ello, sin tener que acudir a la adición de más motores generando redundancias en el sistema y encareciéndolo.

Sin embargo, quedan líneas futuras de trabajo con las que seguir desarrollando el prototipo del robot y mejorando algunas características de este, como, por ejemplo:

- Diseñar la barra 1, L1, como una sola pieza, es decir, no separarlas en izquierda y derecha para poder otorgarle al sistema más rigidez.

- Siguiendo con la aportación de rigidez al robot, otra posible solución sería, en el tope mecánico de la pieza 1 por la parte derecha, diseñar una pieza que vaya desde dicho tope hasta la pieza de metacrilato y ajustar la unión para aportar esos nuevos requerimientos de montaje en esta línea de futura mejora.

- Posibilidad de mecanizar las piezas, ya que por sus pequeñas dimensiones la impresión 3D cuenta con poca precisión según la impresora 3D utilizada, materiales, calidad de la pieza…

- Implementación de otra lectura en la medida del ángulo de giro en O, ϕ. Mediante esta medida se tendrían todas las variables leídas y no habría que realizar cálculos intermedios en el código.

- Implementar una pantalla LCD para poder mostrar diferentes mensajes, como por ejemplo las medidas que están tomando los diferentes elementos del sistema.

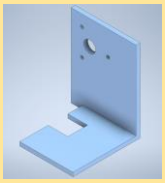
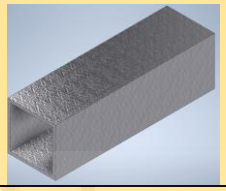
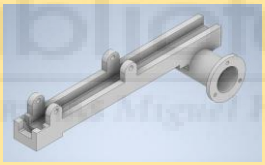
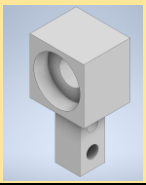
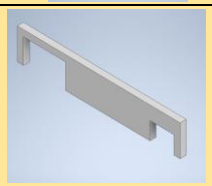
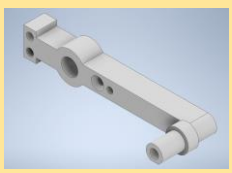
- Programar una interfaz gráfica de usuario para manejar el robot y visualizar gráficos como las curvas $(\vartheta_2, \vartheta_3)$ definidas en la sección 2.3.

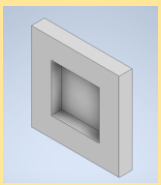
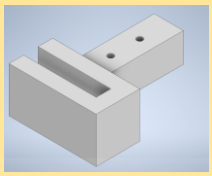
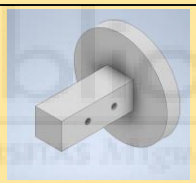
5. Anexos

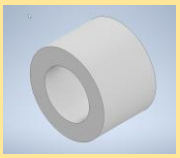
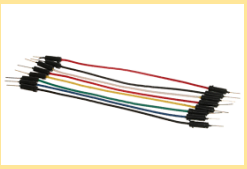
5.1 Materiales del robot y sus características

Para entrar en detalle sobre los materiales utilizados en el montaje del robot, se incluye a continuación una tabla con la lista de los materiales con su correspondiente numeración, el nombre, una imagen, las unidades utilizadas en el sistema y si se trata de una pieza impresa en 3D.

NÚMERO	DENOMINACIÓN	FOTO	UDS	IMPRESIÓN 3D
1	**MOTOR DOGA**		1	NO
2	**EMBRAGUE SO19 HUCO**		1	NO
3	**TORNILLOS M5**		3	NO
4	**VARILLAS M10**		1	NO
5	**TUERCA HEXAGONAL M10**		1	NO
6	**TUERCA FRENO M10**		3	NO

7	**RODAMIENTOS**		2	NO
8	**POTENCIÓMETRO**		2	NO
9	**METACRILATO**		1	NO
10	**TUBO METÁLICO**		6	NO
11	**PIEZA AGARRE ACTUADOR**		1	SI
12	**PIEZA 1 DERECHA**		1	SI
13	**PIEZA 1 IZQUIERDA**		1	SI
14	**UNIÓN PIEZA 1**		1	SI
15	**PIEZA 2**		1	SI

16	**PIEZA 3**		1	SI
17	**ESCON 36/2**		2	NO
18	**ZAPATA**		4	SI
19	**SUJECIÓN TUBO METÁLICO A METACRILTAO**		3	SI
20	**SOPORTE MOTOR PATA A MADERA**		1	SI
21	**TORNILLOS M4**		18	NO
22	**TORNILLOS M3**		9	NO
23	**TUERCAS M4**		5	NO
24	**TUERCAS M3**		8	NO

25	**TUERCAS DE FRENO M4**		18	NO
26	**ESCUADRAS**		4	NO
27	**ARANDELAS SEPARACIÓN MOTOR - METACRILTO**		3	SI
28	**ACTUADOR LINEAL**		1	NO
29	**PLACA ADUINO DUE**		1	NO
30	**FUENTE DE ALIMENTACIÓN**		1	NO
31	**CABLES**		VARIOS	NO
32	**PROTOBOARD**		1	NO
33	**RESISTENCIA 1KOhm**		1	NO

34	**LED**		1	NO
35	**CALZA**		2	SI

Tabla 10. Lista de materiales utilizados.

Cabe destacar que el material utilizado en la impresión 3D ha sido el PLA, ácido poliláctico debido a sus buenas características para un primer prototipo. Este material es derivado de recursos naturales, es de fácil uso porque tiene unas temperaturas de funcionamiento relativamente bajas en comparación a otros materiales, existe una variedad de colores amplia por lo que hemos podido utilizar diferentes en las piezas y cumplía con la resistencia que debe aportar al sistema.

Las impresoras 3D utilizadas para la impresión de las piezas han sido la BQ Witbox 2 y CTC Bizer.

Respecto a la base del robot, se solicita una pieza de metacrilato para poder observar todo el sistema, debido a su transparencia. Es ligero en comparación a otros materiales y resistente. Es importante resaltar, la facilidad que ha aportado dicha pieza a la hora de realizar taladros nuevos, ya que los anteriores huecos estaban hechos mediante corte láser.

Por último, los tubos utilizados son metálicos. Por lo tanto, aportan la rigidez y la solidez necesarias al sistema.

5.2 Códigos Arduino

- Código prueba 1

```
const int pinPotenciometro = A0;  // Pin analógico al que está
conectado el potenciómetro
const float longitudMaximaCarrera = 200.0;  // Longitud máxima
de la carrera en milímetros
void setup() {
  Serial.begin(9600);
}
void loop() {
  // Lee la posición del potenciómetro
  int lecturaPotenciometro = analogRead(pinPotenciometro);
  // Convierte la lectura analógica a milímetros
  float longitudActualCarrera = map(lecturaPotenciometro, 0,
1023, 0, longitudMaximaCarrera);
  // Envía la longitud de la carrera por el puerto serie
  Serial.println(longitudActualCarrera);
  // Espera un breve tiempo antes de la siguiente lectura
  delay(1000);
}
```

En cuanto al código anterior, tenemos la función *map* de Arduino que se encarga de traducir el valor de la longitud de carrera de los 0 a 1023, ya que tiene una resolución de 10 bits que tiene la lectura analógica del actuador lineal a la longitud de carrera de 0 a 200 mm.

Respecto al conexionado, este sería el realizado para la prueba inicial de movimiento del actuador lineal:

Arduino	Fuente Alimentación	Actuador lineal	Denominación
Pin A0	-	Morado	Variable que queremos medir
3,3 V	-	Amarillo	Referencia + tensión
GND	-	Naranja	Referencia - tensión
-	**Referencia -**	Rojo	Movimiento actuador lineal
-	**Referencia +**	Negro	Movimiento actuador lineal

Tabla 11. Conexionado prueba 1.

- Código prueba 2

```
const int pinPotenciometro = A0;
```

```
const int pinControlDriver = 31;
const int pinControlVoltaje = 2;
const float longitudMaximaCarrera = 200.0;

void setup() {
  Serial.begin(9600);
  pinMode(pinControlDriver, OUTPUT);
  pinMode(pinControlVoltaje, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    processInput(input);
  }
  int lecturaPotenciometro = analogRead(pinPotenciometro);
  float longitudActualCarrera = map(lecturaPotenciometro,
0, 1023, 0, longitudMaximaCarrera);
  // Imprime en el Monitor Serie solo si no hay nuevos
comandos
  if (Serial.available() == 0) {
    Serial.print("Longitud de carrera (mm): ");
    Serial.println(longitudActualCarrera);
    // Imprime en el Plotter
    Serial.print(longitudActualCarrera);
    Serial.println();  // Agrega un salto de línea
  }
}
void processInput(String input) {
  int index = input.indexOf(',');
  if (index != -1) {
    // Divide la cadena en partes separadas por comas
    String comandoA = input.substring(0, index);
    String comandoB = input.substring(index + 1);
    // Convierte los comandos a números
    int valorComandoA = comandoA.toInt();
    float valorComandoB = comandoB.toFloat();
    // Validación y control del comando A
    if (valorComandoA == 1) {
      digitalWrite(pinControlDriver, HIGH);
    } else if (valorComandoA == 0) {
      digitalWrite(pinControlDriver, LOW);
    } else {
      Serial.println("Error: Comando A debe ser 0 o 1");
      return;  // Sale de la función si el comando A no es
válido
    }
    // Calcula el porcentaje y el valor en la escala de 255
    float Bporcentaje = 10 + (80 / 24.0 * (valorComandoB -
(-12.0)));

    // Ajusta B255 para estar en el rango de 26 a 229
    int B255 = round(constrain(Bporcentaje * 255 / 100, 26,
229));

    // Controla el voltaje utilizando el valor ajustado de
B255
    analogWrite(pinControlVoltaje, B255);
```

```
        // Imprime en el Monitor Serie solo cuando se recibe un
    nuevo comando
        Serial.print("Comando B, VOLTAJE (-12V a +12V): ");
        Serial.println(valorComandoB);
        Serial.print("Porcentaje de B: ");
        Serial.println(Bporcentaje);
        Serial.print("Valor de B en escala de 255 ajustado: ");
        Serial.println(B255);
      }
    }
```

Este código está diseñado en bucle abierto, donde la señal de salida no afecta a la de control.

Hay destacar que, en el código de esta prueba, por un lado, continuamos con el plotter graficando la longitud de carrera en milímetros respecto al tiempo. Y, por otro lado, tenemos lo que nos imprime el monitor serie que será la longitud de carrera igualmente y cuando ejecutemos los comandos A y B, nos los mostrará también, incluyendo el valor de Bporcentaje y B255. Para el cálculo de B porcentaje utilizamos la ecuación de una recta, ya que esto viene definido desde la configuración de la servocontroladora, donde hemos puesto que para una tensión de -12V tendremos un porcentaje, PMW, del 10, y para 12V, 90%, quedando la ecuación como $Bporcentaje = 10 + (80/24 * (B - (-12)))$. Respecto a B255, es el valor traducido a la escala de 0V a 3,3V, que es el valor que utiliza el comparador para determinar el ciclo útil, ya que no se expresa en tanto por ciento. En su lugar se expresa como un número entre 0 y 255 porque este es el valor máximo que puede alcanzar el contador.

Para generar la señal PWM con Arduino, el comparador mantiene su salida en estado alto (HIGH) mientras el valor del temporizador sea menor que el del ciclo útil. Una vez el valor del contador supera el valor del ciclo útil, la salida pasa a estado bajo (LOW). De esta forma, un valor de 255 de ciclo útil realmente genera un ciclo de trabajo del 100%, mientras que con un valor de 128, en este caso, se logra un ciclo de trabajo del 50%. La ecuación que sigue para obtener este valor es la siguiente: $B255 = Bporcentaje * 255/100$.

En este caso, no se logró que variara la velocidad de salida del vástago, es decir, entraba y se retraía de manera correcta, pero sin variar la tensión. Sin embargo, en posteriores ensayos con controladores PI sí se conseguirá.

- Código prueba 3

```
#include <math.h> // Incluimos la librería math.h para poder
usar atan2() y otras funciones matematicas

// Definición de variables globales y constantes
const int pinPotenciometro = A0; // Pin al que está conectado el
potenciómetro
const int pinControlDriver = 31; // Pin para controlar el driver
del actuador lineal
const int pinControlVoltaje = 2; // Pin para controlar el
voltaje del actuador lineal
const float longitudMaximaCarrera = 200.0; // Longitud máxima de
la carrera del actuador lineal
float valorDeseadoB = 0; // Valor deseado de B
double L1 = 0.114; // Longitud barra 1
double L2 = 0.057; // Longitud barra 2
double L3 = 0.038; // Longitud barra 3
double a = 0.01975; // Longitud vertical desde la varilla
roscada hasta el inicio del vástago
double b = 0.0201; // Longitud horizontal desde la varilla
roscada hasta el inicio del vástago
double rho = 0; // Longitud de extensión del vástago del
actuador lineal

// Definir los pines analógicos donde están conectados los
potenciómetros
const int potPin1 = A1;
const int potPin2 = A2;
// Definir el número de grados por vuelta completa del
potenciómetro
const int degreesPerRotation = 360;
// Definir el umbral para la zona muerta del potenciómetro
const int deadZoneThreshold = 51; // 5% de 1023
// Variable para almacenar la última lectura del potenciómetro 1
y 2
int lastPotValue1 = 0;
int lastPotValue2 = 0;

// Definición de variables para el control PI
const double Kp = 0.8; // Constante proporcional
```

```
const double Ki = 0.0006; // Constante integral
double setpoint = 0; // Valor deseado del control PI
double input, output; // Variables para el control PI
double error, lastError; // Variables para el control PI
double integral, derivative; // Variables para el control PI
unsigned long lastTime; // Variable para el control PI
unsigned long sampleTime = 1000; // Intervalo de tiempo para el
control PI (en milisegundos)


void setup() {
  Serial.begin(9600); // Inicializar la comunicación serial
  pinMode(pinControlDriver, OUTPUT); // Configurar el pin del
driver del actuador lineal como salida
  pinMode(pinControlVoltaje, OUTPUT); // Configurar el pin del
voltaje del actuador lineal como salida

  lastTime = millis(); // Inicializar el tiempo para el control
PI
}

void loop() {
  if (Serial.available() > 0) { // Si hay datos disponibles en
el puerto serial
    String input = Serial.readStringUntil('\n'); // Leer la
cadena de entrada desde el puerto serial
    processInput(input); // Procesar la cadena de entrada
  }

  // Lectura del valor del potenciómetro 1 y 2 (rango de 0 a
1023)
  int potValue1 = analogRead(potPin1);
  int potValue2 = analogRead(potPin2);

  // Convertir los valores del potenciómetro a grados (rango de
0 a 360 grados)
  float degrees1 = potValue1 * 360.0 / 1023.0 + 90.0; //le suma
90 para ajustar la posicion inicial del 0 a nuestro sistema, que
es en la horizontal y 10 porque hay un pequeño desfase
  float degrees2 = potValue2 * 360.0 / 1023.0;

  // Convertir los valores del potenciómetro a radianes
  float theta_2= degrees1 * PI /180.0;
  float theta_3= degrees2 * PI /180.0;

  // Calculamos K3 y K4
  double resultadoK3 = calcularK3(theta_2, theta_3);
  double resultadoK4 = calcularK4(theta_2, theta_3);
```

```
  // Calculamos phi usando atan2(K4, K3)
  double phi = atan2(resultadoK4, resultadoK3);

  // Puedes imprimir o utilizar los resultados como desees
  Serial.print("Phi en radianes: ");
  Serial.print(phi);
  Serial.print(" y phi en grados: ");
  Serial.println(phi*180.0/PI);

  // Ajustar los grados dentro del rango de 0 a 360 grados
  while (degrees1 >= 360) {
    degrees1 -= 360;
  }

  // Ajustar los grados dentro del rango de 0 a 360 grados
  while (degrees2 >= 360) {
    degrees2 -= 360;
  }

  // Verificar si se encuentra en la zona muerta
  if (abs(potValue1 - lastPotValue1) >= deadZoneThreshold) {
    // Si está dentro de la zona muerta, indicar que es la zona
muerta
    Serial.println("Zona muerta del potenciómetro 1");
  } else {
    // Imprimir el valor leído en grados en el monitor serial
    Serial.print("Grados del potenciómetro 1: ");
    Serial.print(degrees1);
    Serial.print(" y radianes del potenciómetro 1: ");
    Serial.println(theta_2);
  }

  // Verificar si se encuentra en la zona muerta
  if (abs(potValue2 - lastPotValue2) >= deadZoneThreshold) {
    // Si está dentro de la zona muerta, indicar que es la zona
muerta
    Serial.println("Zona muerta del potenciómetro 2");
  } else {
    // Imprimir el valor leído en grados en el monitor serial
    Serial.print("Grados del potenciómetro 2: ");
    Serial.print(degrees2);
    Serial.print(" y radianes del potenciómetro 2: ");
    Serial.println(theta_3);
  }

  // Actualizar la última lectura del potenciómetro 1 y 2
  lastPotValue1 = potValue1;
  lastPotValue2 = potValue2;
```

```
  // Lectura del valor del potenciómetro
  int lecturaPotenciometro = analogRead(pinPotenciometro);
  float longitudActualCarrera = map(lecturaPotenciometro, 0,
1023, 0, longitudMaximaCarrera);

  // Actualización del setpoint del control PI
  setpoint = valorDeseadoB * 200.0 / 1023.0;


  // Cálculo del error
  error = setpoint - longitudActualCarrera;

  // Cálculo de la integral
  unsigned long currentTime = millis();
  double elapsedTime = (double)(currentTime - lastTime) /
1000.0; // Convertir a segundos
  integral += (error * elapsedTime);

  // Cálculo de la salida del control PI
  output = Kp * error + Ki * integral;

  // Limitar la salida
  output = constrain(output, -11.7, 11.7);

  // Convertir la salida a un valor de voltaje B porcentaje
  output=15+(70.0/24.0*(output+12)); // Bporcentaje sin llegar a
los limites del 10% y el 90%

  // Aplicar la salida al control del voltaje del actuador
lineal
  analogWrite(pinControlVoltaje, round(output*255.0/100.0));

  // Imprimir información
  Serial.print("Longitud Actual en mm: ");
  Serial.println(longitudActualCarrera);

  //Igualar rho a la longitud de carrera en las unidades
correspondientes (m)
  rho = longitudActualCarrera / 1000;

  // Actualizar el tiempo y el error
  lastTime = currentTime;
}

// Función para procesar la entrada recibida a través de la
comunicación serial
void processInput(String input) {
  int index = input.indexOf(',');
  if (index != -1) {
```

```
    String comandoA = input.substring(0, index);
    String comandoB = input.substring(index + 1);

    int valorComandoA = comandoA.toInt();
    float nuevoValorDeseadoB = comandoB.toFloat();

    // Limitar el valor de B a 950 si es mayor que 950
    valorDeseadoB = min(nuevoValorDeseadoB, 950.0);

    // Reiniciar el valor de la integral a 0 si se recibe un
nuevo comando de B
    integral = 0;

    // Validación y control del comando A
    if (valorComandoA == 1) {
      digitalWrite(pinControlDriver, HIGH);
    } else if (valorComandoA == 0) {
      digitalWrite(pinControlDriver, LOW);
    } else {
      Serial.println("Error: Comando A debe ser 0 o 1");
      return;
    }
  }
}

// Función para calcular K1
double calcularK1(double theta_2, double theta_3) {
    return L1 + L2 * cos(theta_2) + L3 * cos(theta_2 + theta_3);
}

// Función para calcular K2
double calcularK2(double theta_2, double theta_3) {
    return L2 * sin(theta_2) + L3 * sin(theta_2 + theta_3);
}

// Función para calcular K3
double calcularK3(double theta_2, double theta_3) {
    double K1 = calcularK1(theta_2, theta_3);
    double K2 = calcularK2(theta_2, theta_3);
    return ((b + rho) * K1 - a * K2);
}

// Función para calcular K4
double calcularK4(double theta_2, double theta_3) {
    double K1 = calcularK1(theta_2, theta_3);
    double K2 = calcularK2(theta_2, theta_3);
    return (a * K1 + (b + rho) * K2);
}
```

En este código podemos encontrar las siguientes partes

1. Definición de variables y constantes:

Donde se definen los pines a los que están conectados el potenciómetro, el controlador del actuador lineal, el controlador del voltaje y se establecen otras constantes como la longitud máxima de la carrera del actuador. Se definen constantes y variables para el control PI, como las constantes de proporcionalidad (Kp) e integral (Ki), el tiempo de muestreo y las variables de error, integral y derivada.

2. Configuración inicial (*Setup*):

Se inicia la comunicación serial y se configuran los pines como entrada o salida según corresponda.

3. Bucle principal (*Loop*):

El bucle principal del programa realiza varias acciones en cada iteración:

Lee los datos recibidos por el puerto serial para ajustar la posición deseada del actuador, lee los valores de los potenciómetros para determinar la posición actual del actuador en grados y radianes, calcula la posición deseada del actuador utilizando trigonometría, ejecuta el control PI para ajustar la posición del actuador, convirtiendo la salida del controlador en un voltaje y aplicándolo al controlador de voltaje del actuador, lee la longitud actual de la carrera del actuador y actualiza el *setpoint* del control PI y obtiene la nueva longitud de la extensión del actuador y actualiza las variables del control PI.

4. Procesamiento de entrada serial (*processInput*):

Esta función se encarga de interpretar los comandos recibidos por el puerto serial, que consisten en un comando de control del actuador

(habilitar o deshabilitar) y un valor de posición deseada del actuador, valida los comandos y ajusta los valores según sea necesario.

5. Funciones de cálculo de cinemática:

Se definen funciones para calcular varios parámetros geométricos necesarios para la cinemática del sistema, como K1, K2, K3 y K4, utilizando los ángulos de los potenciómetros.

6. Definición de la zona muerta de los potenciómetros de eje:

Se verifica si los potenciómetros de eje están dentro de la zona muerta o existen saltos de una medida a la siguiente excesivamente grandes, mostrando mensajes en el monitor serial si así se confirmara.

5.3 Planos de las piezas impresas en 3D

A continuación, se añaden los planos orientativos, siguiendo el Sistema Europeo, relativos a las piezas diseñadas en Inventor que han sido impresas en 3D.

I. Plano para la pieza del agarre del actuador.

II. Plano para la pieza que evita la colisión de superficies entre el motor y el metacrilato, la arandela 3D.

III. Plano para la pieza 1 de la parte izquierda.

IV. Plano para la pieza 1 derecha.

V. Plano para la pieza 2.

VI. Plano para la pieza 3.

VII. Plano para el tope mecánico de la pieza 2.

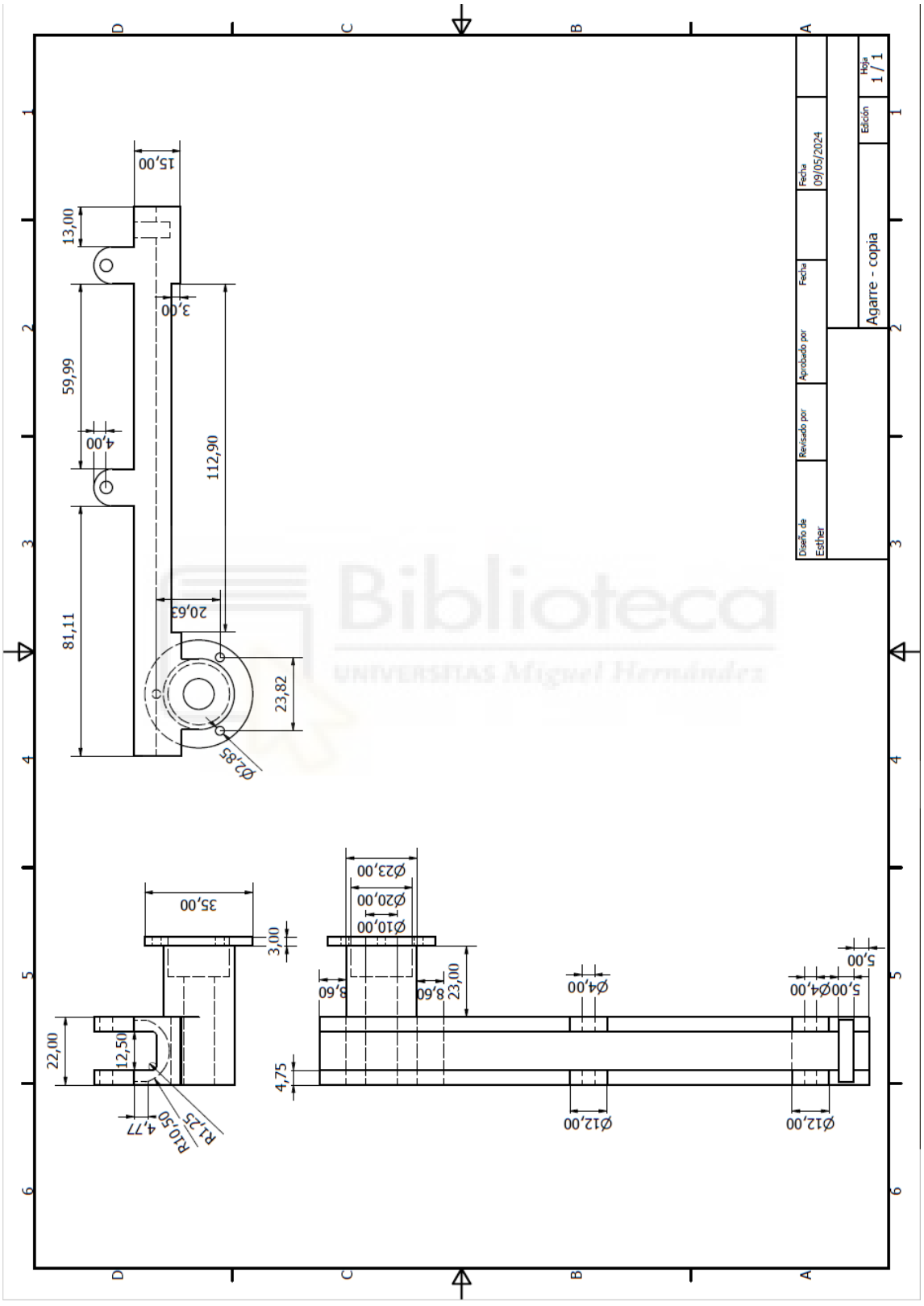
VIII. Plano para la pata del motor.

IX. Plano para la pata del metacrilato.

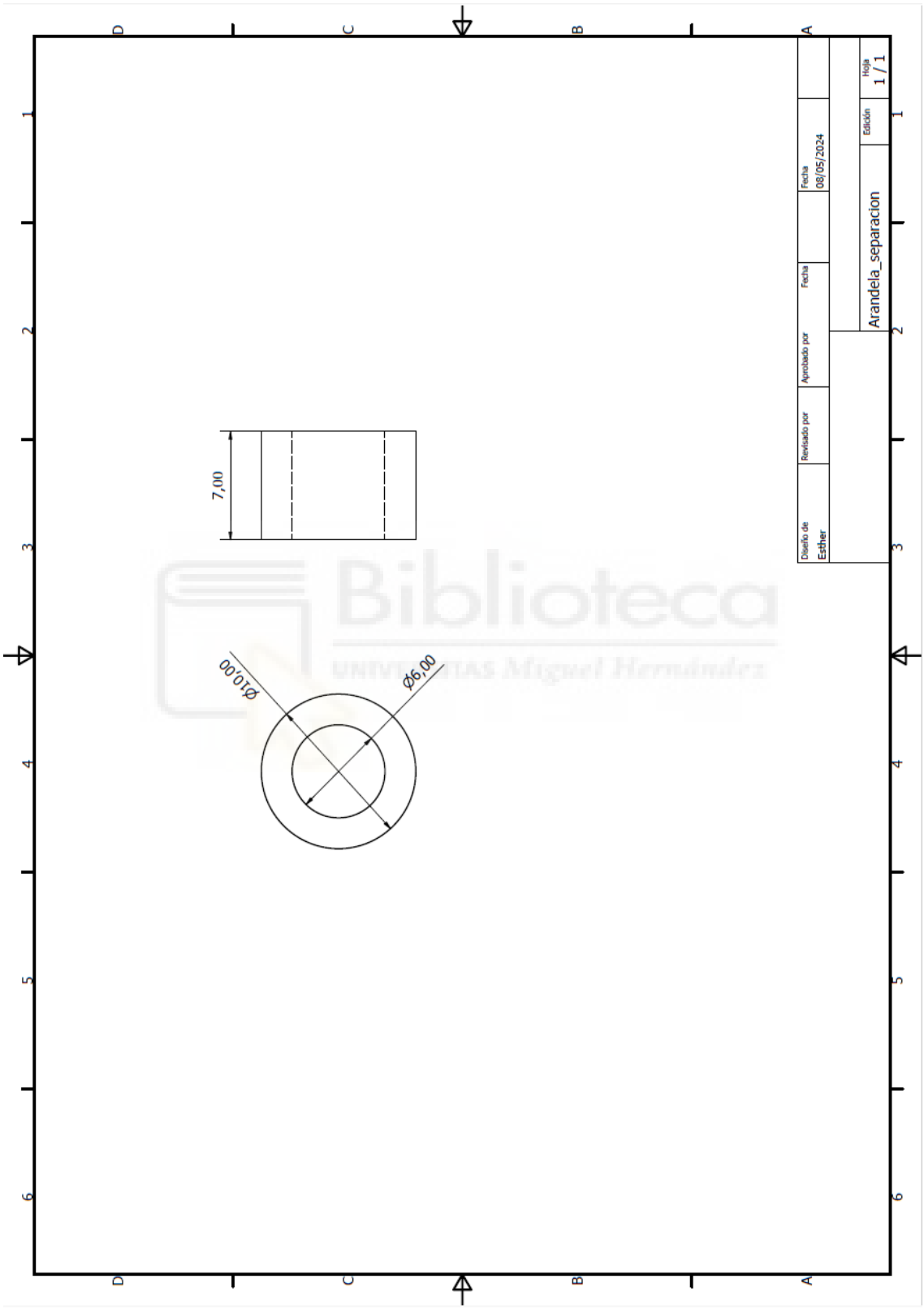
X. Plano de la zapata de las patas.

XI. Plano de la pieza de soporte del motor.

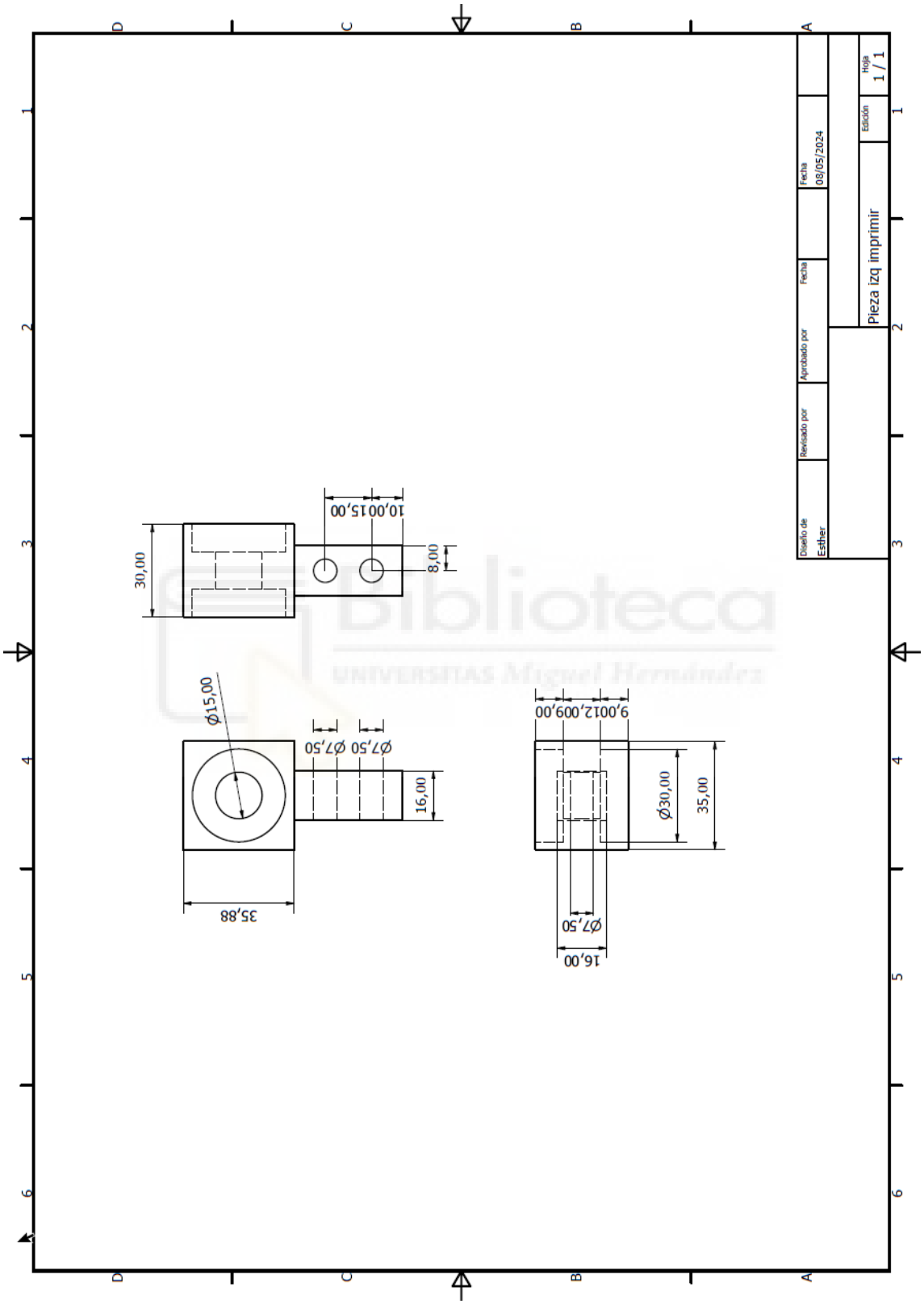
Plano de la pieza de agarre del actuador.



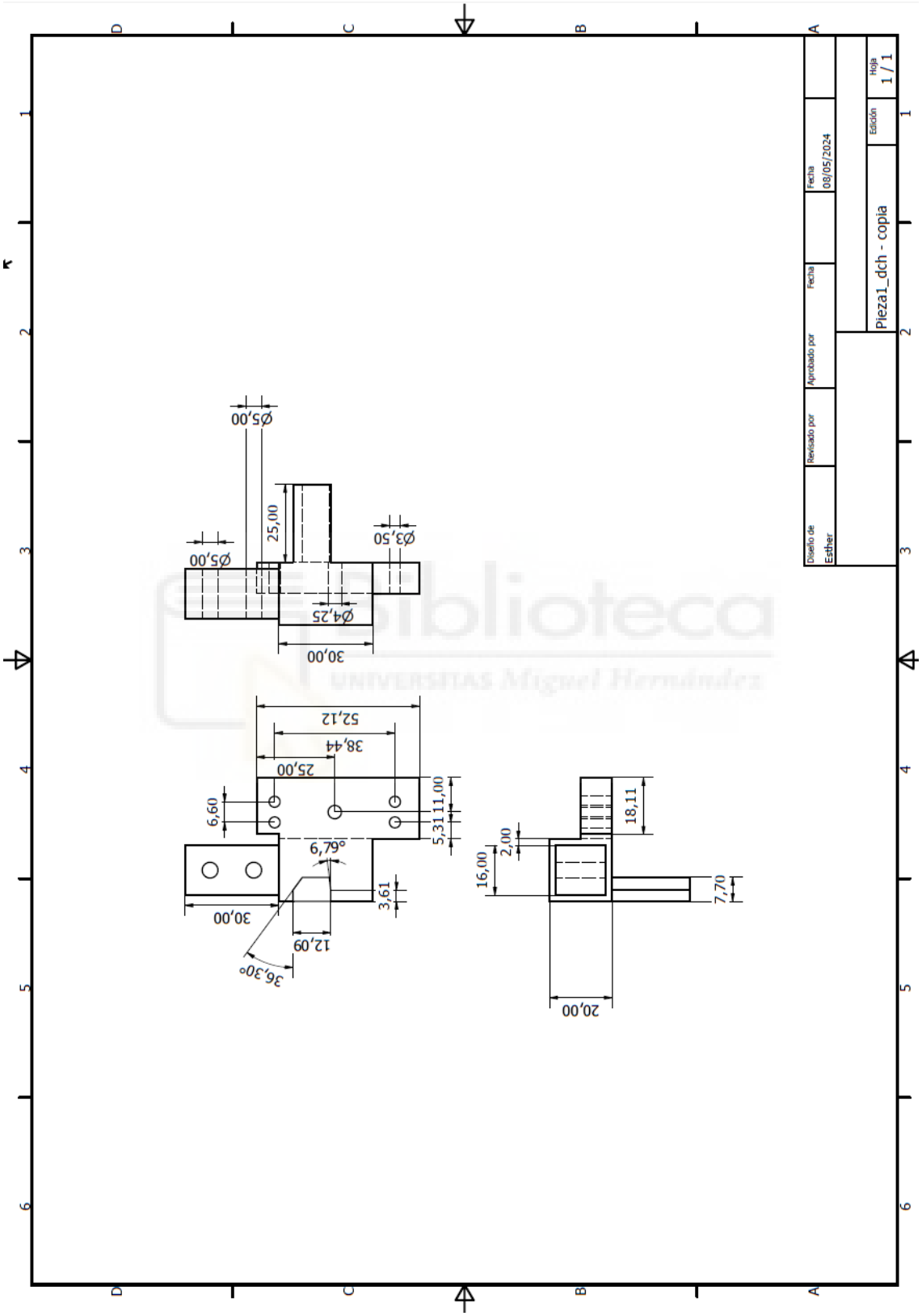
Plano de la arandela.



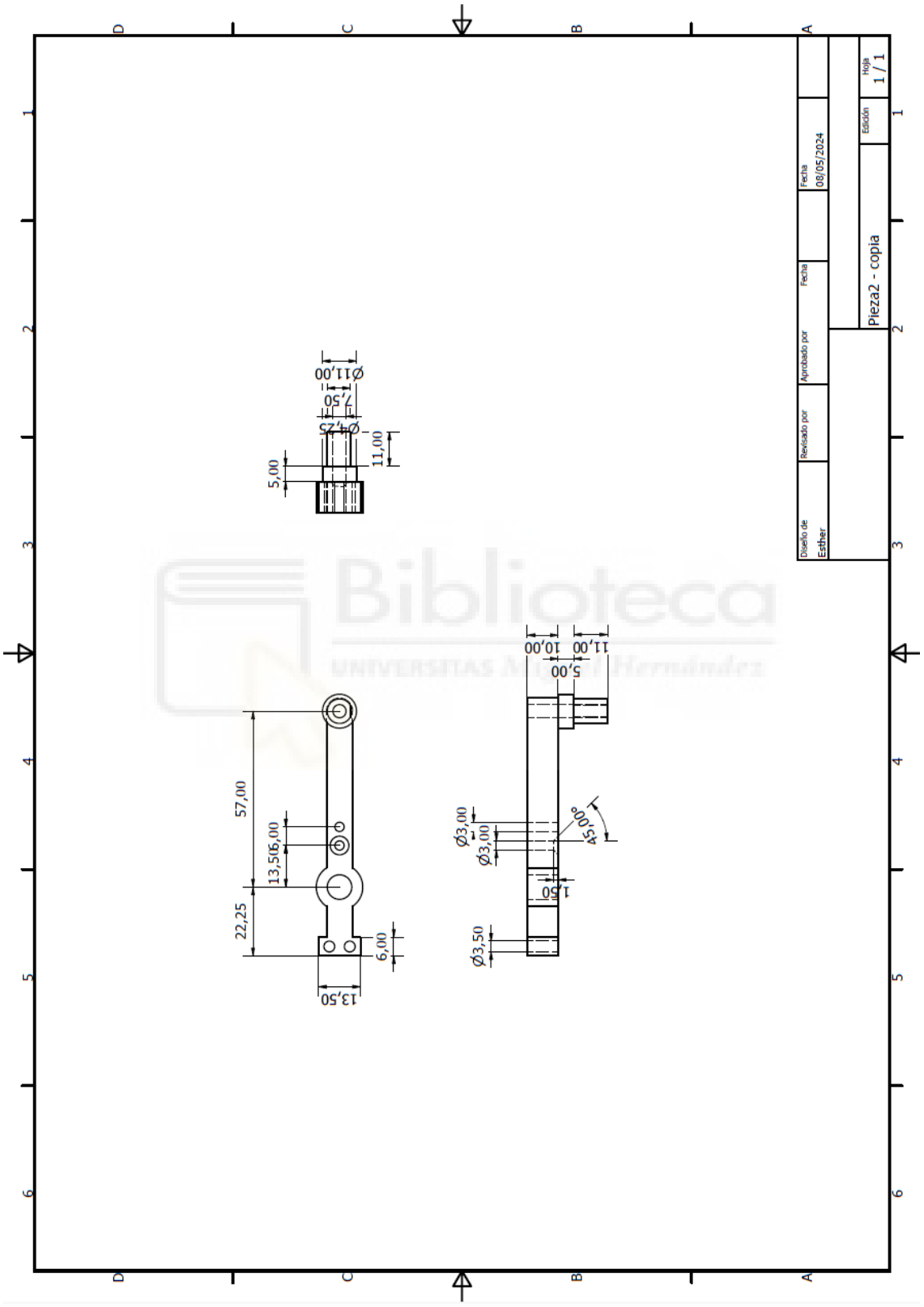
Plano de la pieza 1 izquierda.



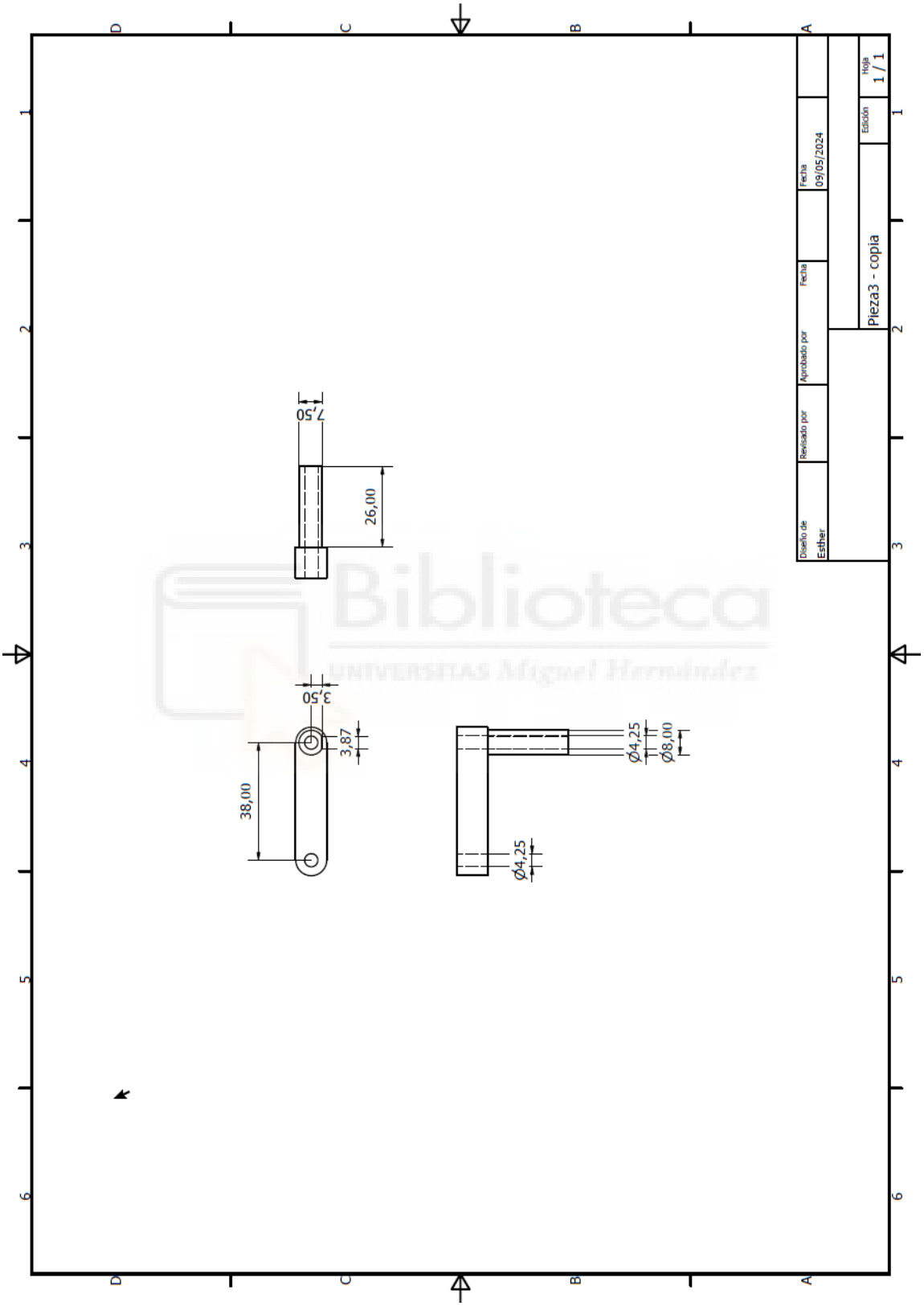
Plano de la pieza 1 derecha.



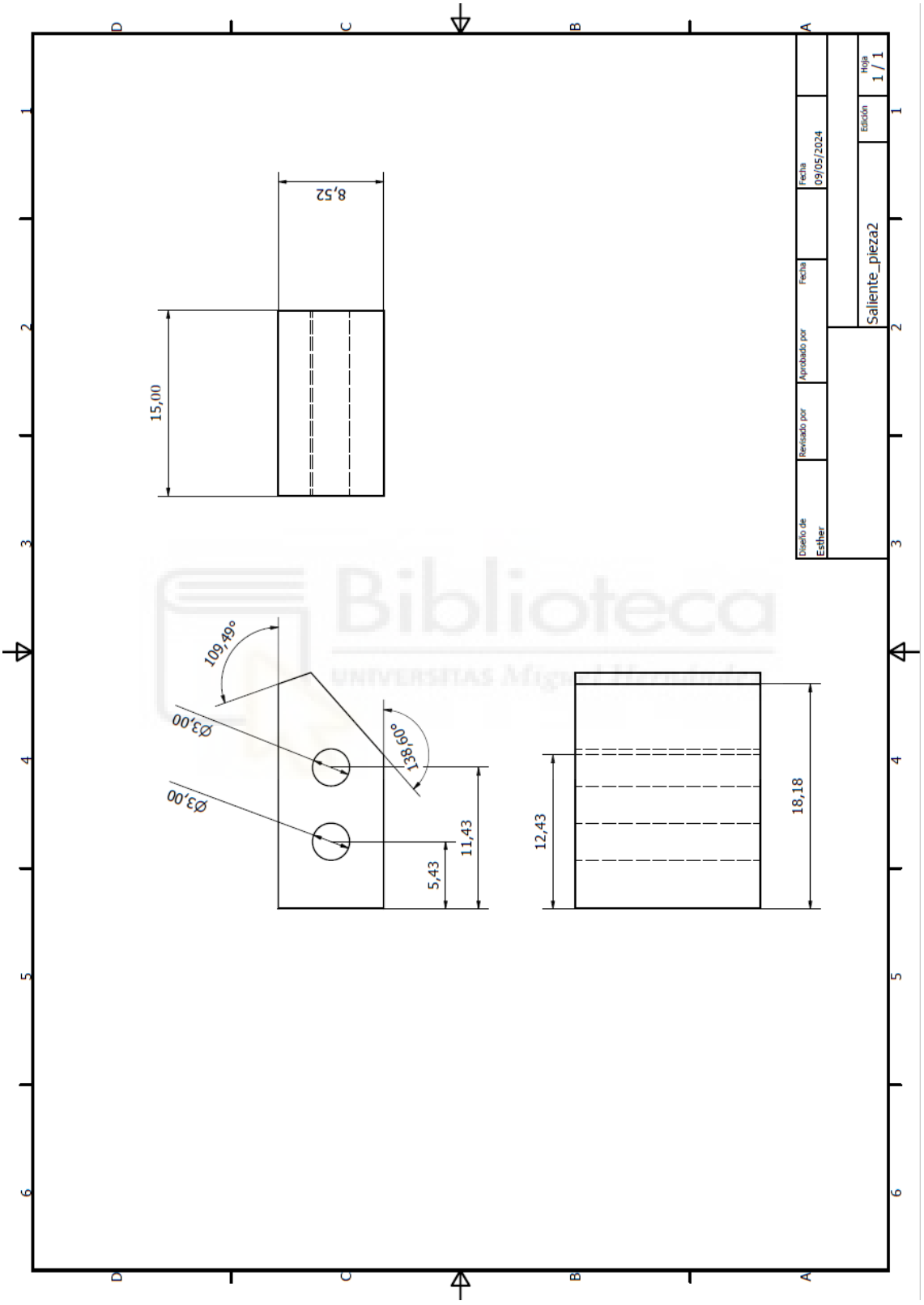
Plano de la pieza 2.



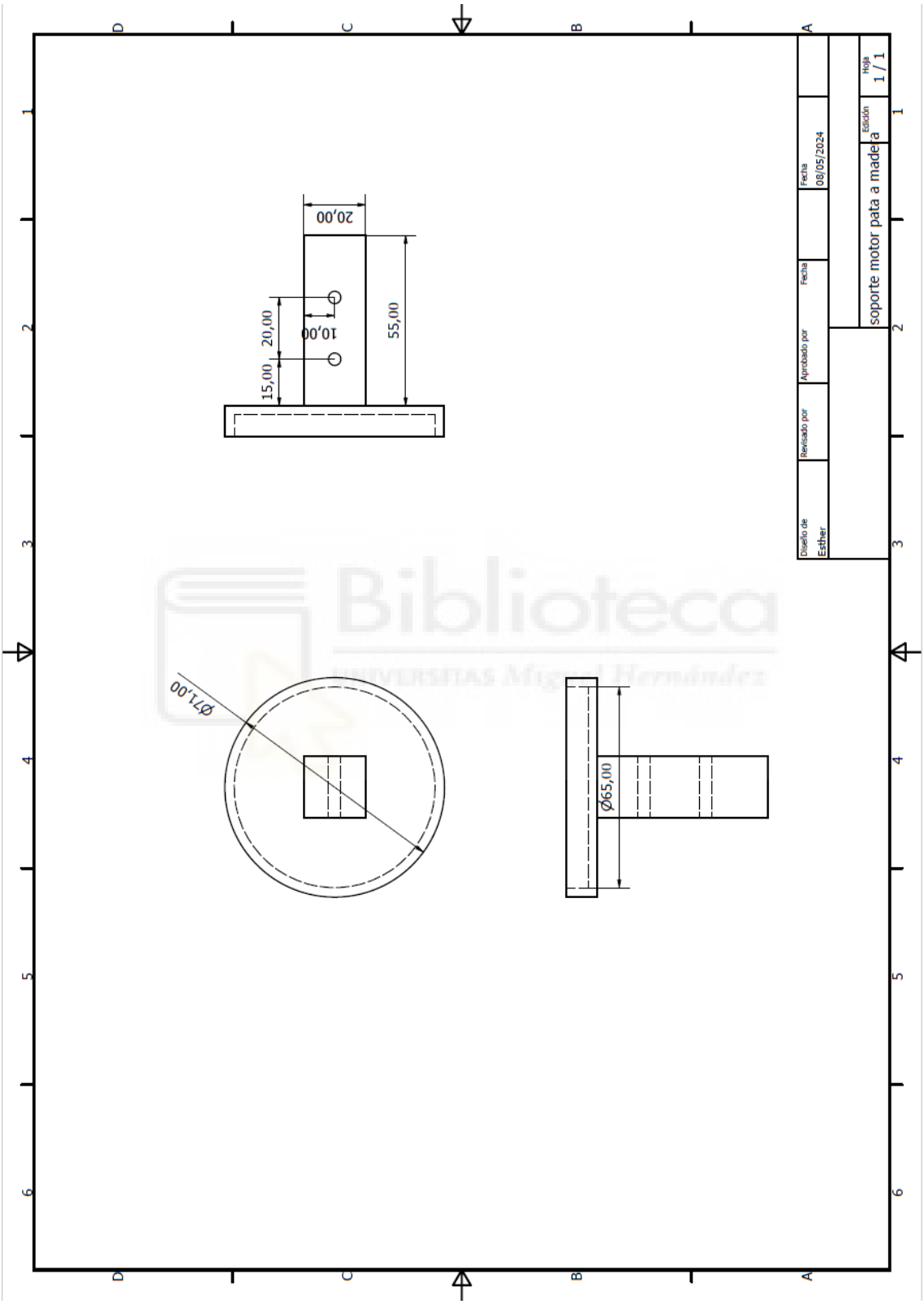
Plano de la pieza 3.



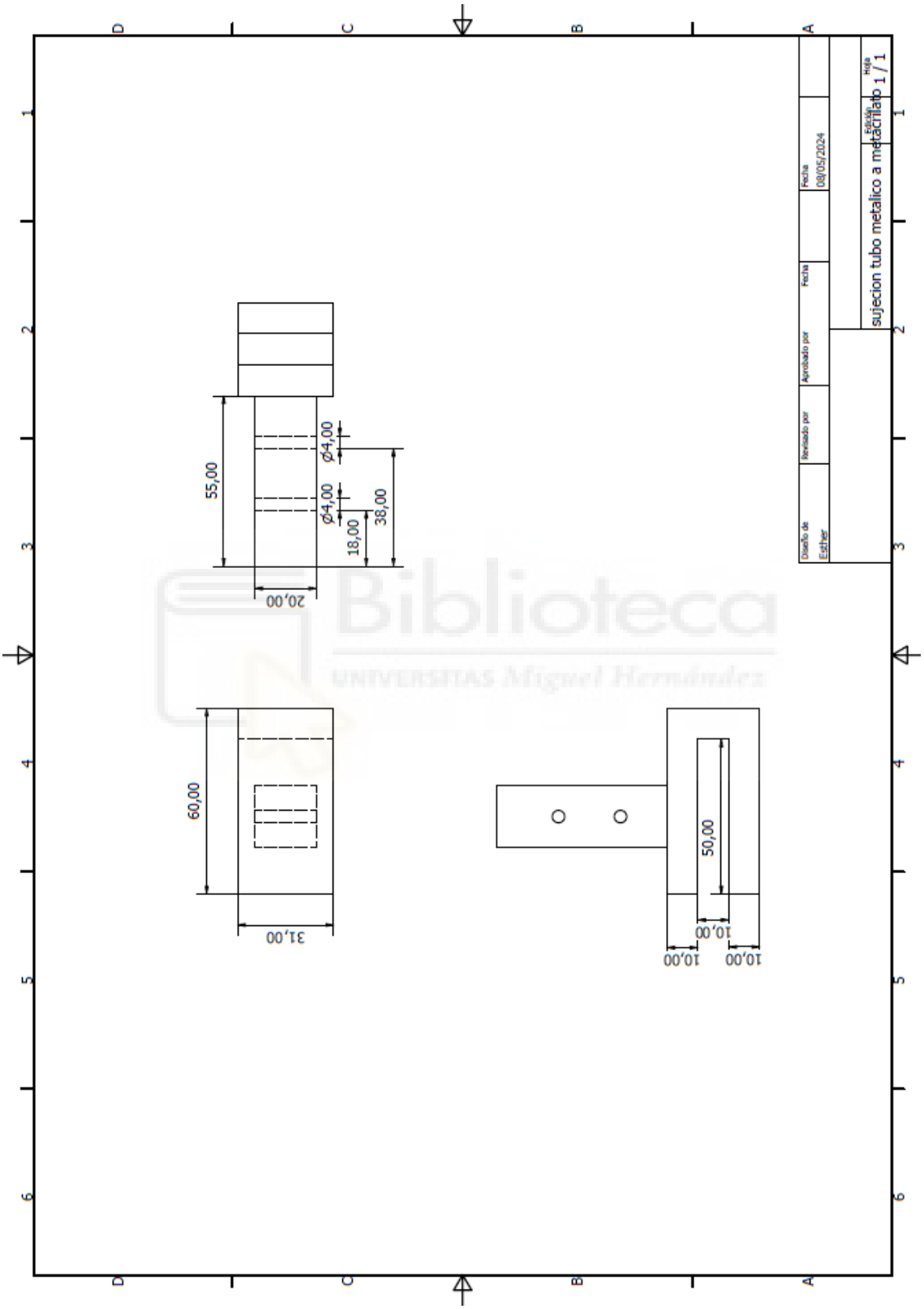
Plano del tope mecánico.



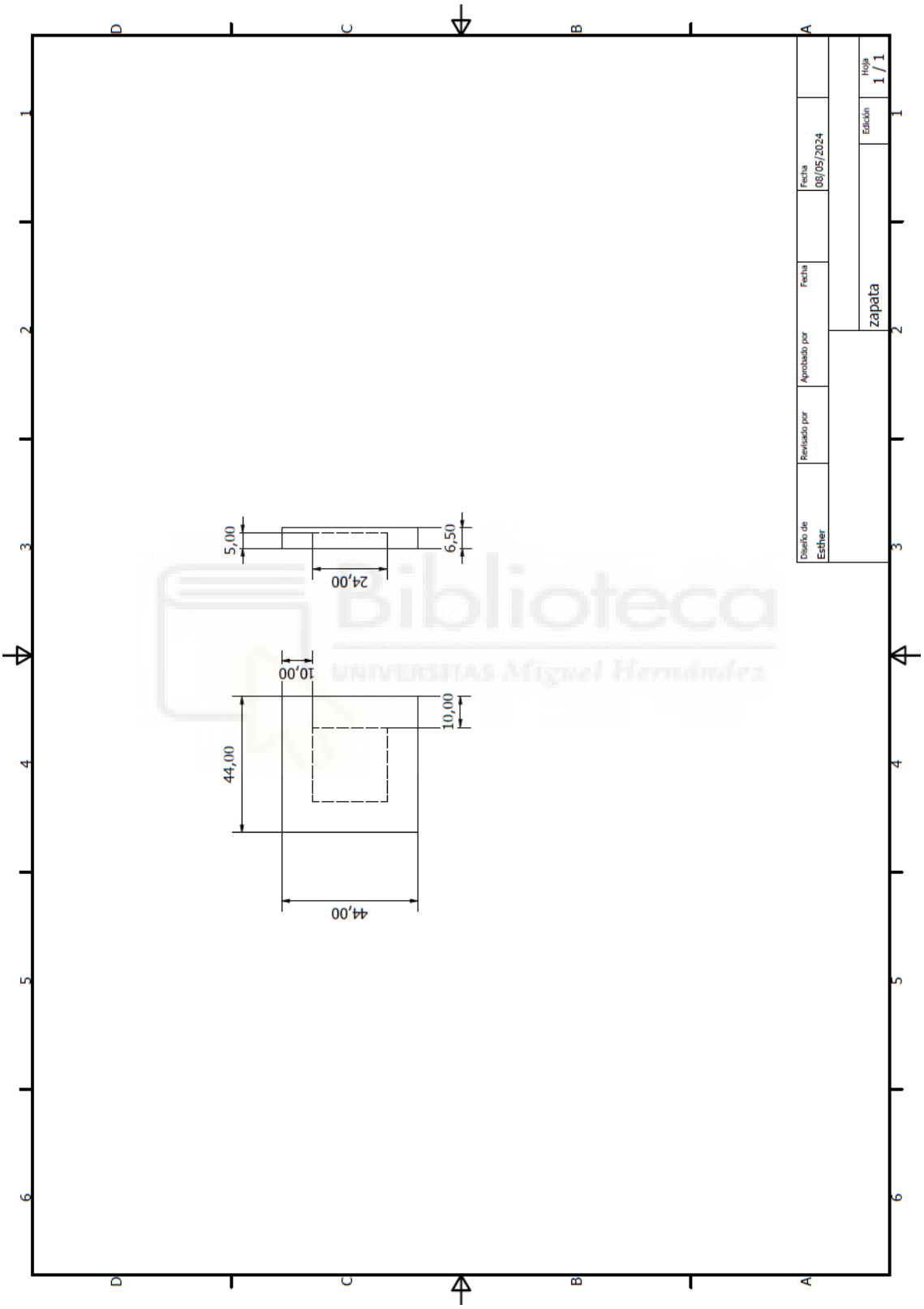
Plano de la pata del motor.



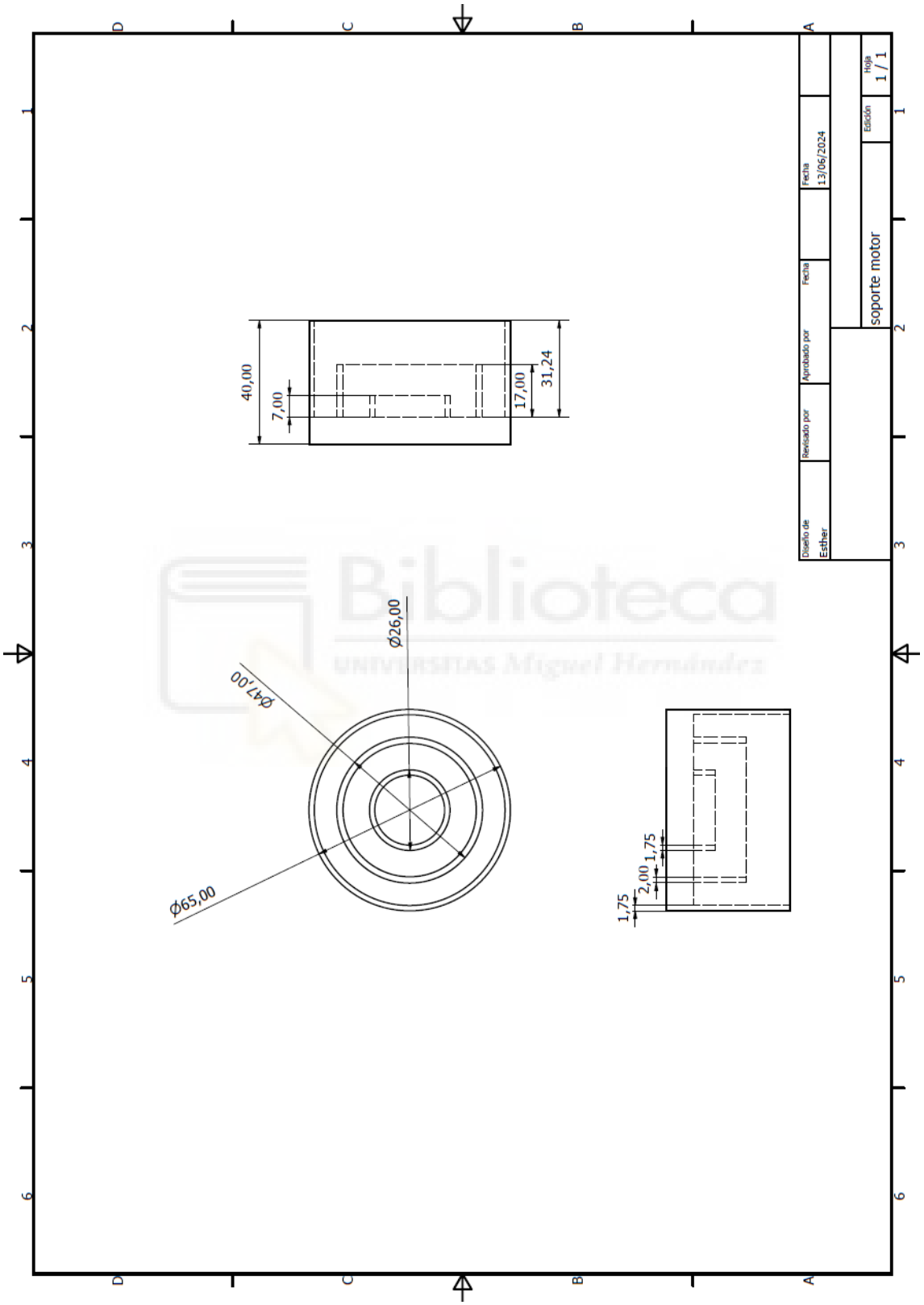
Plano de la pata del metacrilato.



Plano de la zapata.



Plano del soporte motor.



6. Referencias

[1]. A. Peidró, P. Pérez, R. Puerto, L. Payá, O. Reinoso. “Locking underactuated robots by shrinking their manifolds of free-swinging”. Mechanism and Machine Theory (2023). https://www.sciencedirect.com/science/article/pii/S0094114X2300174X#bib1

[2]. Embrague HUCO SO19 https://www.huco.com/shop/power-transmission/brakes-clutches/electromagnetic-clutches/so19

[3]. Motorreductor DOGA https://es.rs-online.com/web/p/motores-dc/7736787

[4]. Actuador lineal Actuonix P16-200-64-12-P https://www.actuonix.com/assets/images/datasheets/ActuonixP16Datasheet.pdf

[5]. Rodamiento de bolas Rodamiento de bolas de ranura profunda de fila única RS PRO de Acero, Ø int. 10mm, Ø ext. 30mm https://es.rs-online.com/web/p/rodamientos-de-bola/6189985?gb=s

[6]. Potenciómetro de 5kΩ, ±20%, 2W, 1 módulo, eje de 8 mm de Ø, Montaje en Panel https://es.rs-online.com/web/p/potenciometros/8427169?gb=s

[7]. Placa Arduino DUE https://es.rs-online.com/web/p/arduino/7697412?cm_mmc=ES-PLA-DS3A-_-google-_-CSS_ES_ES_Pmax_Test-_--_-7697412&matchtype=&&gad_source=1&gclid=CjwKCAjwouexBhAuEiwAtW_ZxwfbJxTX_rcSkiIwS-0Q6twM8VmhLe3IoPkZwxf8Zlixk7mV5s79hBoCMQwQAvD_BwE&gclsrc=aw.ds

[8]. Servocontroladora ESCON 36/2 https://www.maxongroup.com/maxon/view/product/control/4-Q-Servokontroller/403112

[9]. Web Maxon Group – servocontroladora ESCON 36/2 https://www.maxongroup.com/es-es/motores-y-sistemas/controladores/reguladores-de-corriente-y-velocidad

[10]. Generador de esquemas o circuitos electrónicos online https://www.circuitlab.com/editor/#?id=7pq5wm&from=homepage