UNIVERSITAT POLITÈCNICA DE VALÈNCIA

E.T.S. DE INGENIERÍA DEL DISEÑO

TRABAJO FIN DE MÁSTER

IMPLEMENTACIÓN Y CONTROL DE UN PROTOTIPO DE ROBOT SCARA

Autor:

JAVIER MARTÍN CALVO

Director:

VICENTE FERMÍN CASANOVA CALVO

JULIO DE 2021

Agradecimientos

En primer lugar, agradecer a mi director por aceptar este proyecto, orientarme sobre como enfocar las tareas y por ayudarme siempre que lo he necesitado durante la realización del proyecto.

Quiero agradecer a mis compañeros por el apoyo que he tenido durante la realización del máster y sobre todo durante las duras etapas de realización del proyecto.

Por último y más importante, quiero agradecer a mis padres y familia el apoyo que he recibido constantemente para poder alcanzar, la que era una de mis metas, la realización del Máster de Ingeniería Mecatrónica.

Resumen

En el presente *Trabajo Fin de Máster* (TFM) se enfocará en los robots manipuladores industriales, particularmente en los robots de *pick and place* y, más concretamente, se centra en un robot serial con una configuración muy particular usado muy frecuente en la industria denominado Robot SCARA. Con el objetivo de su implementación como un robot destinado a *pick and place* de pequeñas cargas. Se realizará un estudio matemático de la cinemática y se realizarán numerosas simulaciones con el fin de comprobar todos los cálculos desarrollados. Una vez realizado esto, se procederá a la implementación real del robot, además, será necesario la creación de una interfaz de usuario que facilite la comunicación entre el robot y el usuario. Como paso final se desarrolla el software que controla el robot, en base a los conocimientos adquiridos tras las simulaciones, para integrarse en un microcontrolador Arduino UNO.

Índice general

Índice de tablas

Índice de figuras

Lista de Acrónimos

PC *Personal Computer*.

PLA *Ácido poliláctico*.

PR *Parallel Robot*.

SCARA *Selective Compliant Assembly Robot Arm*.

SR *Serial Robot*.

TCD *Transformada Cinemática Directa*.

TCI *Transformada Cinemática Inversa*.

TFM *Trabajo Fin de Máster*.

USB *Universal Serial Bus*.

Lista de Símbolos

θ_1	Ángulo que forma el brazo1 respecto el eje x
θ_1^0	Ángulo inicial que forma el brazo1 respecto el eje x
θ_1^f	Ángulo final que forma el brazo1 respecto el eje x
θ_2	Ángulo necesario para desplazarse verticalmente una distancia z
θ_2^0	Ángulo inicial necesario para desplazarse verticalmente una distancia z
θ_2^f	Ángulo final necesario para desplazarse verticalmente una distancia z
θ_3	Ángulo que forma el brazo 2 respecto al brazo 1
θ_3^0	Ángulo inicial que forma el brazo 2 respecto al brazo 1
θ_3^f	Ángulo final que forma el brazo 2 respecto al brazo 1
θ_4	Ángulo que forma la pinza respecto el eje x
θ_4^0	Ángulo inicial que forma la pinza respecto el eje x
θ_4^f	Ángulo final que forma la pinza respecto el eje x
Ab_i	Abrazadera i
FC_i	Final de carrera i
H	Altura máxima que puede alcanzar la pinza
J_i	Acoplamiento i
L_1	Longitud del brazo 1
L_2	Longitud del brazo 2
M_i	Motor paso a paso i
P_i	Polea dentada i

P_r	Paso de la varilla roscada
RA_i	Rodamiento axial i
RR_i	Rodamiento radial i
x_e	Coordenada x del extremo del robot
x_e^0	Coordenada x inicial del extremo del robot
x_e^f	Coordenada x final del extremo del robot
y_e	Coordenada y del extremo del robot
y_e^0	Coordenada y inicial del extremo del robot
y_e^f	Coordenada y final del extremo del robot
z_e	Coordenada z del extremo del robot
z_e^0	Coordenada z inicial del extremo del robot
z_e^f	Coordenada z final del extremo del robot

Capítulo 1

Introducción

En este primer capítulo se expondrán tanto la motivación y los antecedentes, como los objetivos a lograr y la estructuración de la memoria.

1.1. Motivación y antecedentes

Como se ha podido observar en las últimas décadas, la robótica ha ido creciendo de manera muy significativa, aumentando su impacto en el sector industrial. Este crecimiento es debido a las ventajas que presenta el uso de robots para el desarrollo de tareas y/o aplicaciones como pueden ser la reducción de tiempos, reducción de costes, capacidad de repetición de tareas, etc.

La gran mayoría de robots implementados en la industria para manipulación tienen una configuración serial (del inglés,*Serial Robot* (SR)) aunque también se utilizan con configuración paralela (del inglés,*Parallel Robot* (PR)), véase Figs. 1.1 y 1.2. La principal diferencia entre éstos es la relación cinemática existente entre la base del robot y el efector final, teniendo una única relación cinemática en los robots con configuración serial y varias en los robots con configuración en paralelo.

Figura 1.1: Robot Scara SR-3iA[1]- Robot serie de Fanuc

Centrándose en los robots seriales, la resolución del problema cinemático es muy sencilla en comparación de los robots paralelos. Y más concretamente, el presente proyecto se ha centrado en un robot serial con una configuración muy específica, denominado robot *Selective Compliant Assembly Robot Arm* (SCARA), es decir, se trata de un robot expansible en el eje X-Y y rígido respecto al eje Z.

Este tipo de robots está caracterizado por tener una alta precisión y una velocidad de trabajo muy alta por lo que es empleado en numerosas lineas de producción o como robot de ensamblaje.

Se ha elegido el presente TFM siguiendo la línea principal que promueve el Máster de

[1]Fuente: `https://www.fanuc.eu/es/es`
[2]Fuente: `https://new.abb.com/es`

Figura 1.2: IRB 360 FlexPicker [2]- Robot paralelo de ABB

Ingeniería Mecatrónica que es aunar los conocimientos adquiridos de electrónica, electricidad y mecánica.

1.2. Objetivos

El objetivo principal que se ha fijado para la realización de este proyecto es llevar a cabo el estudio, control y montaje de un Robot Scara (SCARA) de cuatro grados de libertad para una aplicación *pick and place* orientada a la colocación automatizada de objetos. Para la consecución del objetivo principal del proyecto, se han planteado los siguientes objetivos intermedios:

1. **Resolución del problema cinemático directo e inverso.** Se desarrollará y resolverá la transformada cinemática directa e inversa mediante métodos geométricos.
2. **Desarrollo del simulador.** Se procederá a desarrollar un entorno de simulación en el cual se pueda observar la dinámica que rige al robot.
3. **Desarrollo de trayectorias.** Se detallarán las trayectorias espacio-temporales empleadas según las solicitaciones de la dinámica del manipulador.
4. **Montaje del robot.** Se procederá a montar físicamente el robot.
5. **Desarrollo de la Interfaz Gráfica.** Se planteará la creación de una interfaz que, a través de un protocolo de comunicación, facilite el uso del robot al usuario. También

se incluye la creación del protocolo de comunicación entre el PC y el microcontrolador Arduino, que a su vez se comunicará con los drivers de los actuadores.

1.3. Estructura de la memoria

En la siguiente subsección se hará una breve explicación de la estructuración de la memoria.

- **Capítulo 1: Introducción.** Se plantea la motivación y antecedentes que han llevado al desarrollo del presente TFM, así como los objetivos a lograr y la estructuración de la memoria.
- **Capítulo 2: Modelado matemático.** Se definen las variables y los conceptos necesarios para la resolución de la cinemática y la dinámica del robot. Posteriormente, se desarrolla el problema cinemático directo e inverso.
- **Capítulo 3: Diseño hardware.** Se explica de manera precisa todos los elementos que componen el robot.
- **Capítulo 4: Simulación.** Se reflejan los resultados obtenidos mediantes los software correspondientes como paso previo a las ensayos experimentales.
- **Capítulo 5: Implementación real.** Se muestran los pasos a seguir para la construcción del robot.
- **Capítulo 6: Conclusiones y Propuestas de mejora.** Se muestran las conclusiones a las que se ha llegado tras la realización del proyecto, y además, se muestran propuestas para mejorar el robot.
- **Capítulo 7: Presupuesto.** Se muestra el presupuesto de la realización del proyecto.
- **Capítulo 8: Pliego de condiciones.** Se detalla brevemente un pliego de condiciones para el proyecto realizado.
- **Anexo A: Código arduino.** Se explica detalladamente los programas creados para el microcontrolador arduino del proyecto.
- **Anexo B: Código Matlab.** Se detallan las funciones y scrips creados para el desarrollo del proyecto. .
- **Anexo C: Simscape.** En este anexo se muestran los ensamblajes creados para desarrollar el entorno de simulación.
- **Anexo D: Datasheets.** En este anexo se podrán visualizar las especificaciones técnicas de los componentes más relevantes utilizados en el robot
- **Anexo E: Interfaz de usuario.** En este anexo se detalla brevemente la utilización de la interfaz de usuario.

Capítulo 2

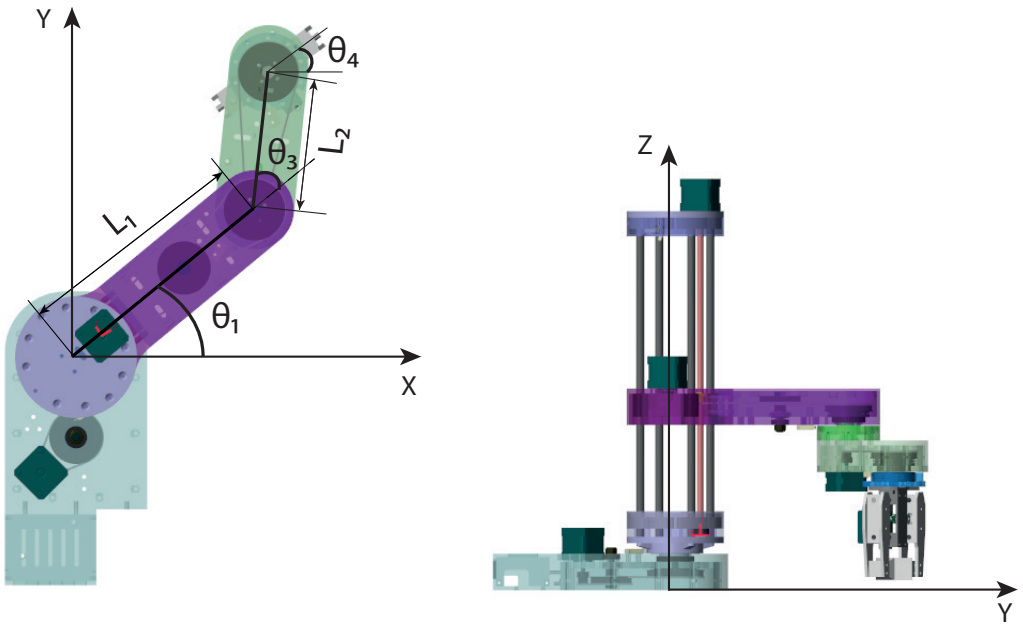
Modelado matemático

En este capítulo se muestran los desarrollos matemáticos del modelo cinemático y dinámico del robot, con el fin de su implementación en simulaciones posteriores.

2.1. Nomenclatura del robot

En el presente TFM se va a trabajar con un brazo robótico de ensamblaje selectivo compatible (del inglés, SCARA).

En la siguiente parte se introducen conceptos y referencias que se emplean en el modelo matemático con el fin de ayudar a una mejor comprensión de los cálculos que se van a realizar. Cabe destacar que se ha definido como sentido de giro rotacional de los motores un sentido levógiro (antihorario). Esto se debe tener en cuenta para la realización de los cálculos de la Sección 2.2.



(a) (b)

Figura 2.1: Cinemática del robot

siendo L_1 la longitud del brazo 1, L_2 la longitud del brazo 2, θ_1 el ángulo que forma el brazo 1 respecto el eje x, θ_2 el ángulo necesario para desplazarse verticalmente una distancia z, θ_3 el ángulo que forma el brazo 2 respecto al brazo 1 y θ_4 el ángulo que forma la pinza respecto el eje x.

Por último, se va a utilizar un sistema de referencia absoluto para el control de los motores por lo que únicamente es necesaria conocer una altura de referencia en el cálculo de la coordenada articular θ_2.

2.2. Problema cinemático

La cinemática de un robot estudia el movimiento de este respecto a uno o varios ejes de referencia, buscando obtener una descripción analítica del movimiento espacial o planar del robot en función del tiempo. Dependiendo de las relaciones que se tomen para buscar esto, se llegará al desarrollo matemático de la *Transformada Cinemática Directa* (TCD) o *Transformada Cinemática Inversa* (TCI), como se puede observar en la Fig. 2.2.

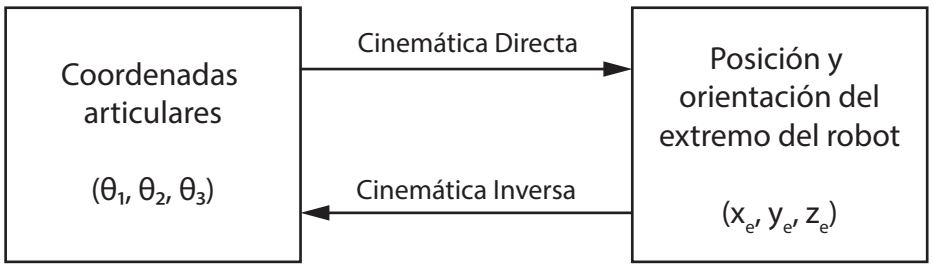


Figura 2.2: Cinemática del robot

2.2.1. Cinemática directa

La resolución de la TCD permite conocer cuál es la posición y orientación del extremo del robot a partir de sus coordenadas articulares [1]. También se ha utilizado como referencia el artículo [2].

Para resolver el problema cinemático directo se parte de una condición inicial, que es la posición del extremo del robot. Por tanto, conocida la posición inicial del extremo del robot, $Q_e^0 = [x_e^0, y_e^0, z_e^0, \theta_4^0]$, y las coordenadas articulares de los motores, $\theta = [\theta_1, \theta_2, \theta_3, \theta_4]$, se procede a calcular la posición final del extremo, $Q_e^f = [x_e^f, y_e^f, z_e^f, \theta_4^f]$, mediante la utilización de reglas trigonométricas básicas.

$$x_e^f = x_e^0 + (L_1 \cdot cos(\theta_1) + L_2 \cdot cos(\theta_1 + \theta_3)) \tag{2.1}$$

$$y_e^f = y_e^0 + (L_1 \cdot sen(\theta_1) + L_2 \cdot sen(\theta_1 + \theta_3)) \tag{2.2}$$

$$z_e^f = z_e^0 + \left(\frac{\theta_2}{360} \cdot P_r\right) \tag{2.3}$$

$$\theta_4^f = \theta_4^0 + \theta_4 \tag{2.4}$$

siendo x_e^f la coordenada x final del extremo del robot, y_e^f la coordenada y final del extremo del robot, z_e^f la coordenada z final del extremo del robot, θ_4^f la orientación final del extremo

del robot, x_e^0 la coordenada x inicial del extremo del robot, y_e^0 la coordenada y inicial del extremo del robot, z_e^0 la coordenada z inicial del extremo del robot y θ_4^0 la orientación inicial del extremo del robot.

2.2.2. Cinemática inversa

La resolución de un problema cinemático inverso permite conocer las coordenadas articulares que el robot debe adoptar para que el extremo del robot tenga una posición y orientación determinadas [1]. También se ha utilizado como referencia el artículo [3] y [4].

Para resolver el problema cinemático inverso se parte de una condición inicial, que es la posición del extremo del robot al igual que en la Sección 2.2.1. Por tanto, conocida la posición inicial del extremo del robot, $Q_e^0 = [x_e^0, y_e^0, z_e^0, \theta_4^0]$, y, la posición y orientación deseada del extremo del robot, $Q_e^f = [x_e^f, y_e^f, z_e^f, \theta_4^f]$, se procede a calcular las coordenadas articulares de los motores, $\theta = [\theta_1, \theta_2, \theta_3, \theta_4]$, mediante la utilización de reglas trigonométricas básicas.

Se comienza con el cálculo de las coordenadas articulares de los motores 1 y 3 que afectan al desplazamiento del robot sobre el plano XY. En la Fig. 2.3 se pueden observar dos triángulos que relacionan el origen de coordenadas con la posición del extremo del robot. Haciendo uso del teorema del coseno sobre el triángulo amarillo se obtiene (2.5) y aplicando el teorema de Pitágoras sobre el triángulo rojo se obtiene (2.6).

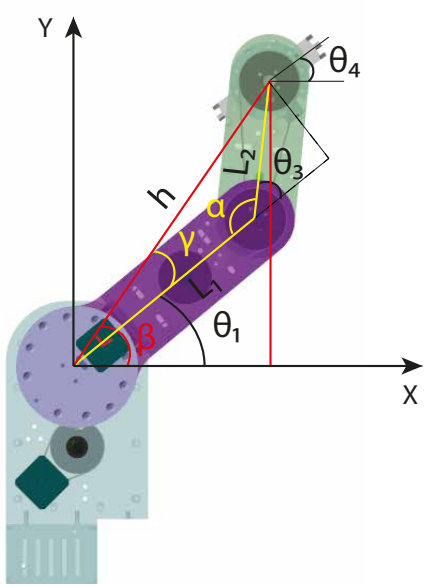


Figura 2.3: Cinemática inversa

$$h^2 = L_1^2 + L_2^2 - 2 \cdot L_1 \cdot L_2 \cdot cos(\alpha) \tag{2.5}$$

$$h^2 = x_e^2 + y_e^2 \tag{2.6}$$

Despejando el coseno del ángulo α de (2.5) junto (2.6) se obtiene (2.7).

$$cos(\alpha) = \frac{x_e^2 + y_e^2 - L_1^2 - L_2^2}{-2 \cdot L_1 \cdot L_2} \tag{2.7}$$

Utilizando el teorema de Pitágoras se tiene (2.8) que junto (2.7) se obtiene el valor del ángulo α (2.9).

$$sen(\alpha) = \pm\sqrt{1 - cos^2(\alpha)} \tag{2.8}$$

$$tan(\alpha) = \frac{sen(\alpha)}{cos(\alpha)} \Rightarrow \alpha = tan^{-1}\left(\frac{\pm\sqrt{1 - cos^2(\alpha)}}{cos(\alpha)}\right) \tag{2.9}$$

Por último, se obtiene la coordenada articular del motor 3, θ_3, (2.10).

$$\theta_3 = \pi - \alpha \tag{2.10}$$

$$\theta_3^f = \theta_3 - \theta_3^0 \tag{2.11}$$

Para el cálculo de θ_1 es necesario calcular antes el ángulo β y el ángulo γ. Para el cálculo del ángulo β se utiliza el triángulo rojo (2.12) y, para el ángulo γ el triángulo marcado con línea discontinua negra (2.13).

$$\beta = tan^{-1}\left(\frac{y_e}{x_e}\right) \tag{2.12}$$

$$\gamma = tan^{-1}\left(\frac{L_2 \cdot sen(\theta_3)}{L_1 + L_2 \cdot cos(\theta_3)}\right) \tag{2.13}$$

Haciendo uso de los ángulos calculados se obtiene el valor de la coordenada articular del motor 1, θ_1, (2.14).

$$\theta_1 = \beta - \gamma = tan^{-1}\left(\frac{y_e}{x_e}\right) - tan^{-1}\left(\frac{L_2 \cdot sen(\theta_3)}{L_1 + L_2 \cdot cos(\theta_3)}\right) \tag{2.14}$$

$$\theta_1^f = \theta_1 - \theta_1^0 \tag{2.15}$$

Para el calculo de la coordenada articular θ_2 es necesario comparar la coordenada z de la posición deseada con la altura máxima que puede alcanzar la pinza, H.

$$\theta_2 = \frac{(H - z_e)}{P_r} \cdot 360 \tag{2.16}$$

$$\theta_2^f = \theta_2 - \theta_2^0 \tag{2.17}$$

siendo P_r el paso de la varilla roscada y H la altura máxima que puede alcanzar la pinza.

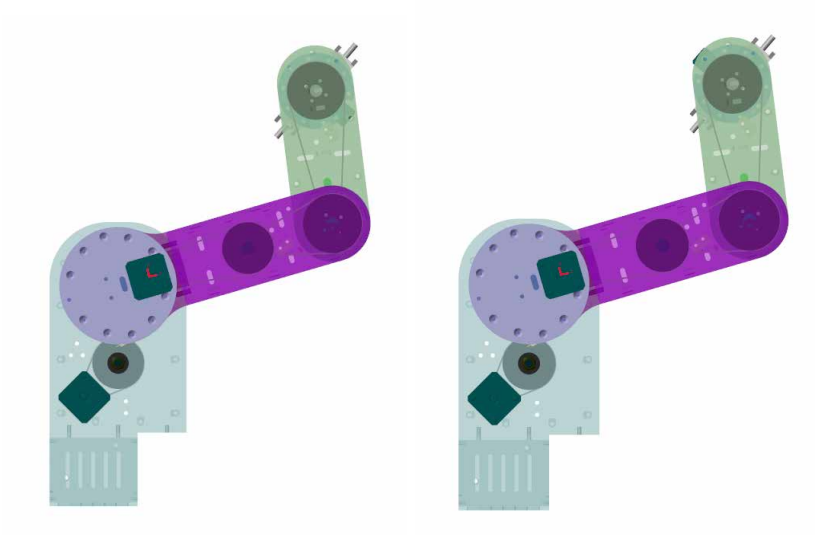
2.2.3. Configuración

Debido a al diseño de los robots SCARAs se puede alcanzar una posición y orientación determinada mediante cuatro configuraciones distintas.

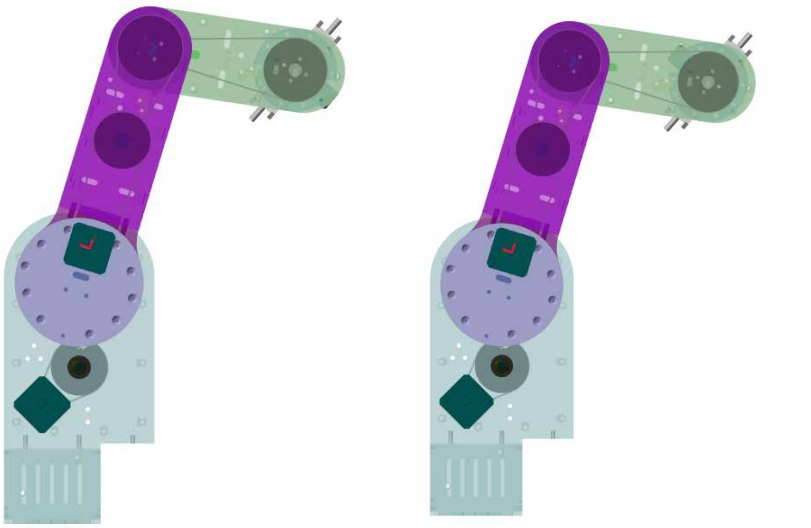
- **Codo arriba.**
- **Codo abajo.**

Las configuraciones codo arriba y codo abajo son combinaciones de las coordenadas articulares θ_1 y θ_3 que permiten al robot alcanzar una posición determinada en el espacio. En la Fig. 2.4 (a) ó (b) se puede observar la configuración codo abajo, y en la Fig. 2.4 (c) ó (d) se puede observar la configuración codo arriba.

Las dos configuraciones mencionadas (codo arriba y codo abajo) se pueden combinar con las dos posibilidades de la coordenada articular θ_4 que permiten alcanzar la misma orientación de la pinza, dando lugar a las cuatro combinaciones de coordenadas articulares que permiten alcanzar una posición y orientación determinada. Se pueden observar las cuatro configuraciones distintas para un punto del espacio en la Fig. 2.4.



(a) Codo abajo (b) Codo abajo rot.



(c) Codo arriba (d) Codo arriba rot.

Figura 2.4: Configuraciones

Capítulo 3

Diseño hardware

En este capítulo se expone la elección del diseño y, además, se detalla la utilidad y la funcionalidad de cada una de las piezas que componen el robot.

3.1. Introducción

El objetivo principal de este capítulo es obtener los archivos .step de las piezas que componen el robot. Este tipo de archivos son propios de los programas de CAD y, concretamente en este caso, se obtienen dichos archivos mediante el programa SolidWorks. La elección de estos archivos se debe a la futura implementación del robot en el Capítulo 4.

Para la elección del diseño se ha recopilado información de diversos diseños de este tipo de robot en diferentes páginas de acceso y uso público. El diseño del robot que finalmente se ha seleccionado es el del siguiente proyecto [5]. Se ha seleccionado debido a la robustez del diseño y a la complejidad del mismo ya que supone un buen reto su implementación hardware. Aunque se ha echo uso de dicho diseño, cabe destacar que se han realizado modificaciones de las dimensiones de numerosas piezas que componen el robot.

El robot va a ser divido en cinco partes que en su conjunto forman el robot scara del proyecto: base, conjunto de elevación, brazo 1, brazo 2 y pinza, como se puede observar en la Fig. 3.1.

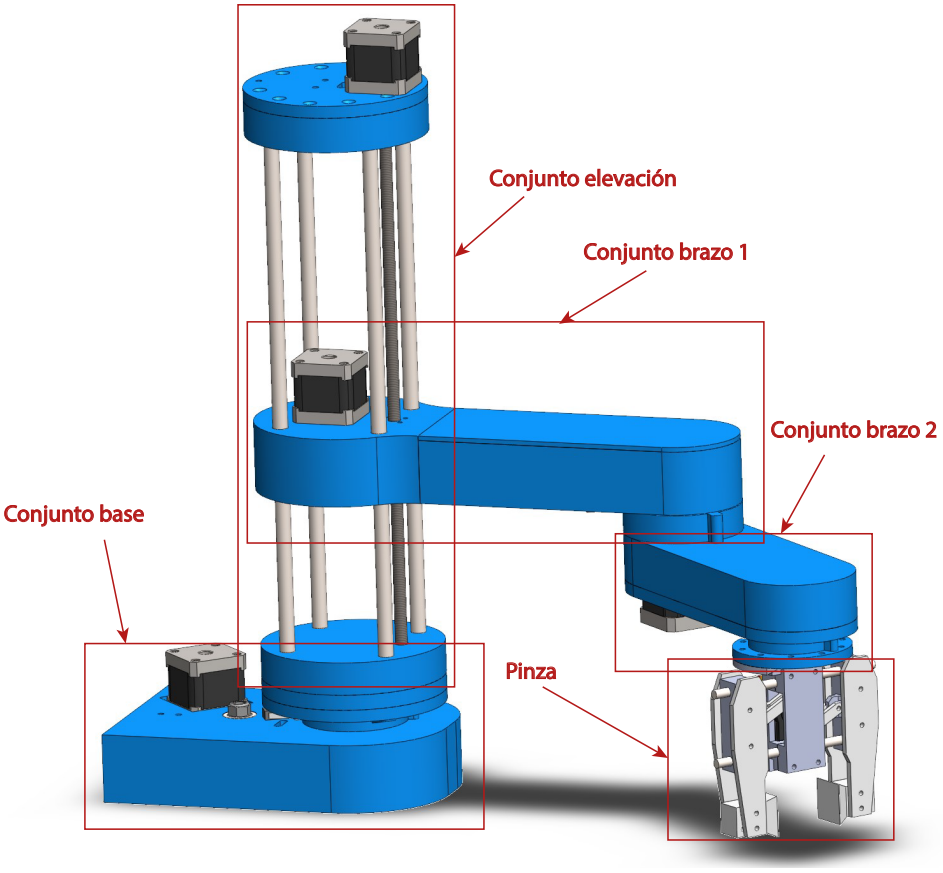
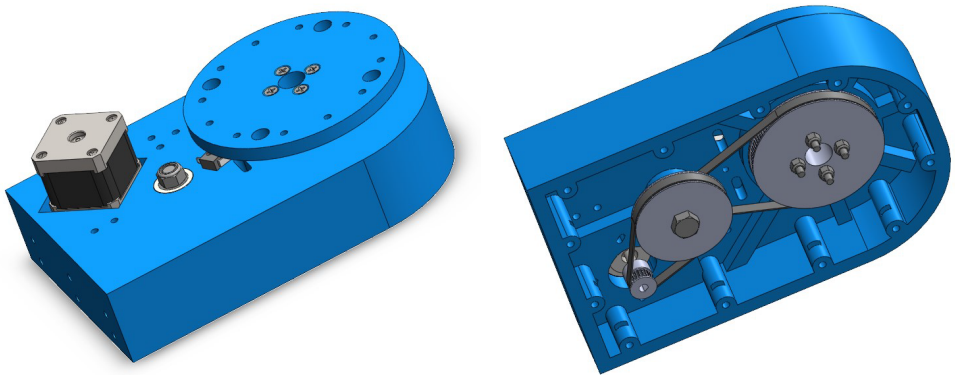


Figura 3.1: Robot Scara

3.2. Conjunto base

El conjunto base esta formado por el grupo de piezas que forman los cimientos del robot, es decir, sirve como sujeción del robot al suelo. Además, en este conjunto se encuentra el primer motor del robot que, mediante una etapa de reducción, permite al brazo robótico un movimiento de rotación respecto al eje z, véase Fig. 3.2.

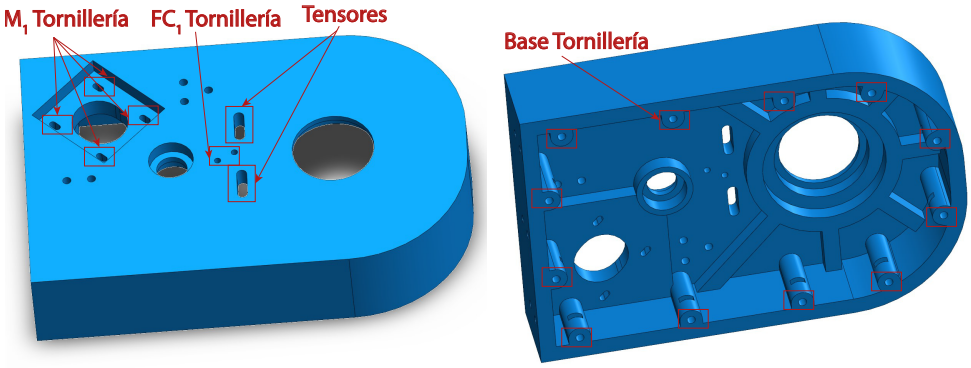


(a) (b)

Figura 3.2: Conjunto base

3.2.1. Base

La pieza denominada como base sirve como sustento del brazo robótico a una plataforma, véase Fig. 3.3. La base es fijada a la plataforma mediante tornillería. Además, ha sido diseñada para el acoplamiento del motor 1 , M_1, el final de carrera 1, FC_1, y un conjunto de transmisión por correas que permite al brazo robótico la rotación sobre el eje z.



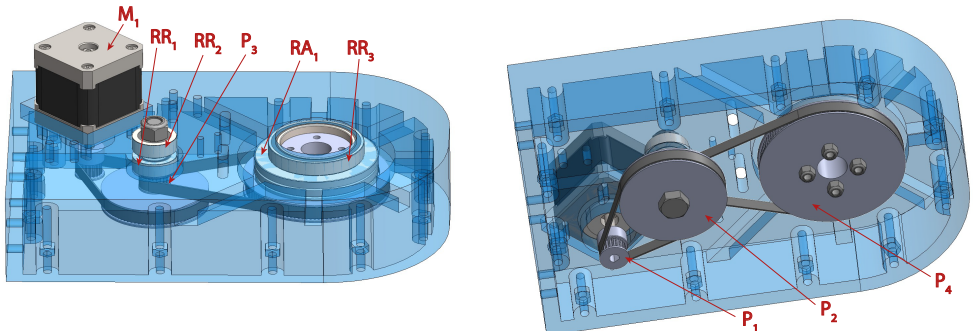
(a) (b)

Figura 3.3: Base

En la Fig. 3.3 (a) se puede observar los orificios dejados para los tornillería del motor M_1, del final de carrera FC_1 y para añadir tensores en la transmisión en el caso de que fuese necesario. En la Fig. 3.3 (b) se puede observar la posición de los orificios que se utilizan para fijar la base a una estructura mediante tornillería.

3.2.2. Transmisión 1

El motor 1, M_1, tiene acoplado a su eje una polea dentada de 20 dientes, P_1, que a su vez transmite el movimiento a otra polea dentada de 80 dientes, P_2, mediante una correa. Esta segunda polea dentada tiene acoplada en su mismo eje otra polea dentada de 22 dientes, P_3, que transmite el movimiento a una polea dentada de 110 dientes, P_4. El conjunto de poleas dentadas forman una etapa de reducción de 1:20. Se puede observar la nomenclatura utilizada en la Fig. 3.4.



(a) (b)

Figura 3.4: Transmisión 1

La polea dentada P_4 está acoplada mediante un rodamiento axial[1], RA_1, para soportar cargas axiales, y un rodamiento radial[2], RR_3, para soportar cargas radiales, al acoplamiento 1. Las poleas dentadas P_2 y P_3 están acopladas mediante dos rodamientos radiales, RR_1 y RR_2.

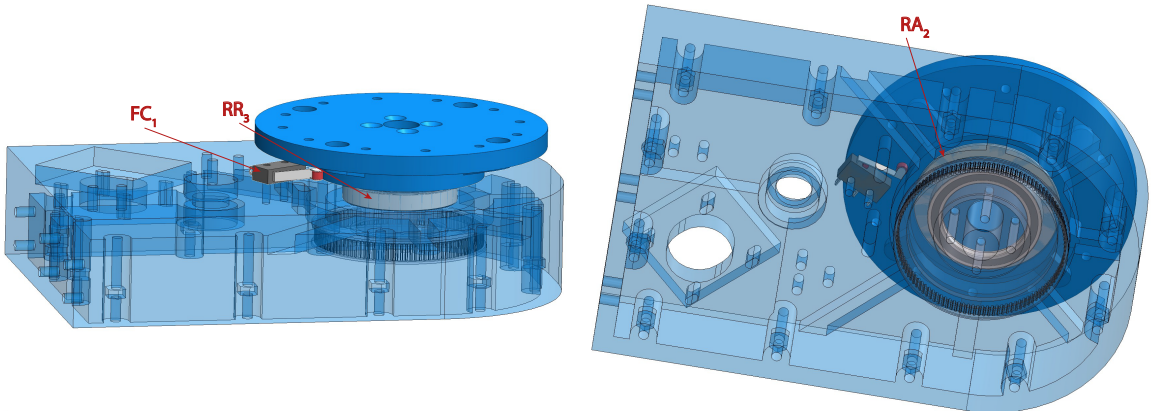
3.2.3. Acoplamiento 1

La siguiente pieza denominada como acoplamiento 1, J_1, es usada como unión entre la base y el conjunto elevación, se puede observar en la Fig. 3.5. Esta pieza rota alrededor del eje z porque está unida al motor M_1 mediante un conjunto de transmisión de movimiento rotatorio (Sección 3.2.2). En la Fig. 3.5 (a) se puede observar la posición del final de carrera 1, FC_1, y la posición del rodamiento radial al que está acoplado J_1 (mismo que al que esta

[1]Rodamiento axial: está diseñado para resistir esfuerzos en la misma dirección que el eje.

[2]Rodamiento radial: está diseñado para resistir esfuerzos de dirección normal a la dirección que pasa por el centro de su eje.

acoplado la polea dentada P_4). En la Fig. 3.5 (b) se puede observar la posición del rodamiento axial al que está acoplado J_1.



(a) (b)

Figura 3.5: Acoplamiento 1

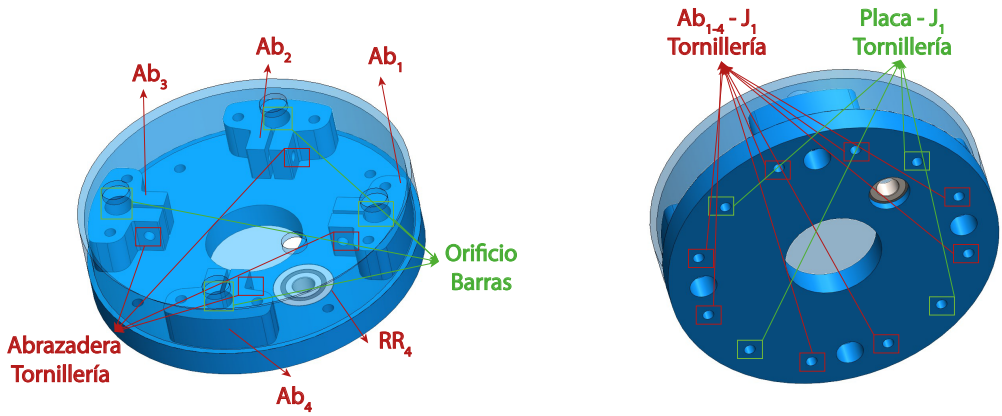
3.3. Conjunto elevación

El conjunto elevación esta formado por el grupo de piezas que sirven de guías y provocan que el brazo 1 realice un movimiento lineal ascendente o descendente a lo largo del eje z. Este conjunto puede dividirse en diferentes piezas de las que cabe destacar cuatro: placa inferior, barras guía, barra roscada y placa superior, véase Fig. 3.6.

Figura 3.6: Conjunto elevación

3.3.1. Placa inferior

La placa inferior es un elemento que sirve como elemento de unión entre el acoplamiento J_1 y las barras guía. Las barras guía son sujetadas en posiciones específicas mediante abrazaderas, AB_i, siendo i el número de la abrazadera. Las abrazaderas son fijadas a la placa inferior y al acoplamiento J_1 mediante tornillería observada en la Fig. 3.7 (b). Se puede ajustar la presión que ejerce la abrazadera a la barra guía mediante tornillería, véase Fig. 3.7 (a). Se puede observar un rodamiento radial, RR_4, que evita la oscilación de la barra roscada pero además permite su rotación, véase Fig. 3.7 (a).



(a) (b)

Figura 3.7: Placa inferior

Cabe destacar que la tapa de la placa inferior es meramente un elemento estético que sirve para ocultar las abrazaderas.

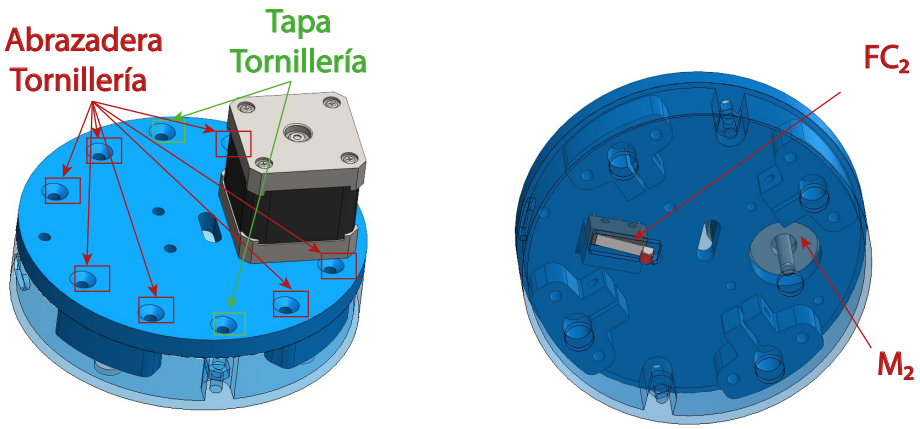
3.3.2. Barras guía y barra roscada

Las barras guía sirven para asegurar que el movimiento lineal ascendente o descendente a lo largo del eje z del brazo 1 se realice correctamente, repartiendo la inercia generada por el brazo 1 y brazo 2 entre las diferentes barras guía. La barra roscada es la pieza que junto a la plataforma de montaje transforma el movimiento de rotación proveniente del motor 2 en un movimiento lineal que hace ascender/descender el brazo 1.

3.3.3. Placa superior

La placa superior es utilizada como soporte para el motor 2, M_2, y el final de carrera 2, FC_2, véase Fig. 3.8. Esta placa superior es unida a la placa inferior mediante las barras guías que como se ha detallado en la Sección 3.3.1 van fijadas mediante cuatro abrazaderas y, estas,

a su vez, van fijadas a la placa superior mediante tornillería. El motor M_2 esta fijado a la placa superior y su eje está acoplado a la barra roscada que produce el movimiento vertical del brazo robótico.



(a) (b)

Figura 3.8: Placa superior

También se utiliza la placa superior como soporte del final de carrera FC_2 fijado mediante tornillería. Al igual que en la placa inferior, la placa superior cuenta con una tapa cuya finalidad es ocular las abrazaderas.

3.4. Conjunto brazo 1

El conjunto brazo 1, se puede observar en la Fig. 3.9, se acopla al conjunto de elevación mediante las barras guía y la barra roscada.

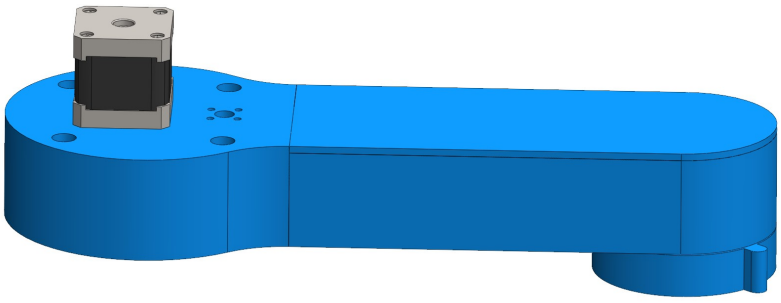
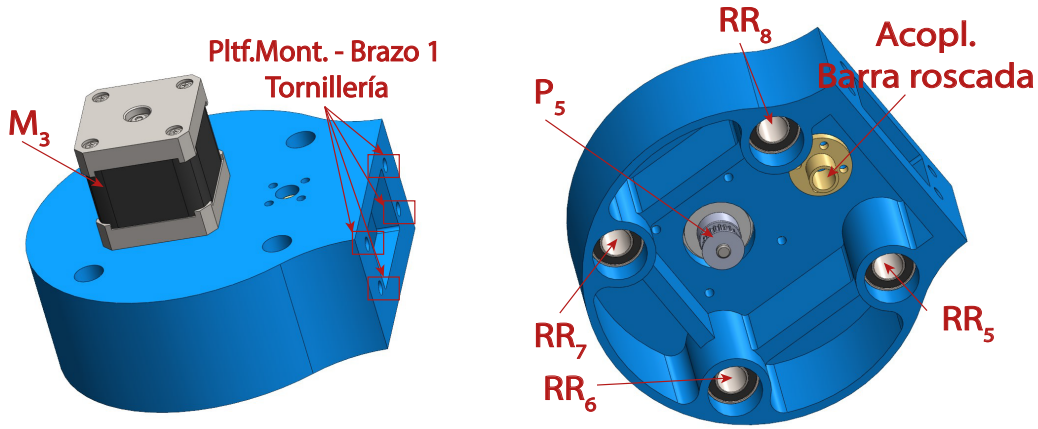


Figura 3.9: Conjunto brazo 1

Este conjunto tiene fijado un motor que mediante una transmisión provoca un movimiento de rotación alrededor del eje z sobre el acoplamiento 2. Se puede dividir en cuatro partes según su finalidad: la plataforma de montaje, el brazo 1, la transmisión 2 y el acoplamiento 2.

3.4.1. Plataforma de montaje

La plataforma de montaje esta acoplada a la barra roscada mediante la unión vista en la Fig. 3.10 (b) que permite el movimiento ascendente/descendente del brazo robótico cuando la barra rota mediante el movimiento del motor M_2. Además, la plataforma tiene cuatro orificios para acoplar cuatro rodamientos radiales: RR_5, RR_6, RR_7 y RR_8 donde van insertadas las barras guía.



(a) (b)

Figura 3.10: Plataforma de montaje

El motor 3, M_3, está situado en esta plataforma para reducir la inercia que generaría si se situará en el brazo 1. Por último, en la Fig. 3.10 (a) se puede observar los orificios para la tornillería qe sirve de unión entre la plataforma de montaje y el brazo 1.

3.4.2. Brazo 1

El brazo 1 tiene como finalidad principal la fijación de la transmisión 2 que produce el movimiento del brazo 2 (codo de este tipo de robots). Además, es la estructura donde van acoplados los tensores si fuesen necesarios, el final de carrera 3, FC_3 y los rodamientos que se utilizan en la transmisión 2. Todo ello se puede observar en la Fig. 3.11.

3.4.3. Transmisión 2

El motor 3, M_3, tiene acoplado a su eje una polea dentada de 20 dientes, P_5, que a su vez transmite el movimiento a otra polea dentada de 80 dientes, P_6, mediante una correa. Esta segunda polea dentada tiene acoplada en su mismo eje otra polea dentada de 23 dientes, P_7, que transmite el movimiento a una polea dentada de 92 dientes, P_8. El conjunto de poleas

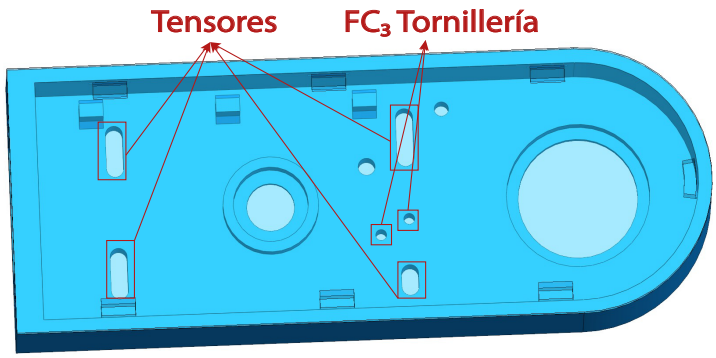


Figura 3.11: Brazo 1

dentadas forman una etapa de reducción de 1:16. Se puede observar la nomenclatura utilizada en la Fig. 3.12.

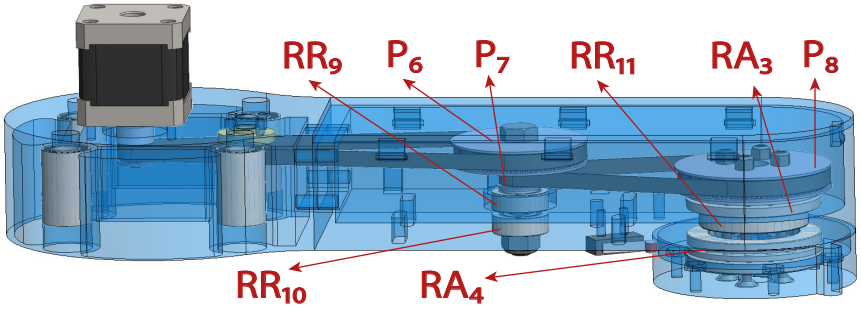


Figura 3.12: Transmisión 2

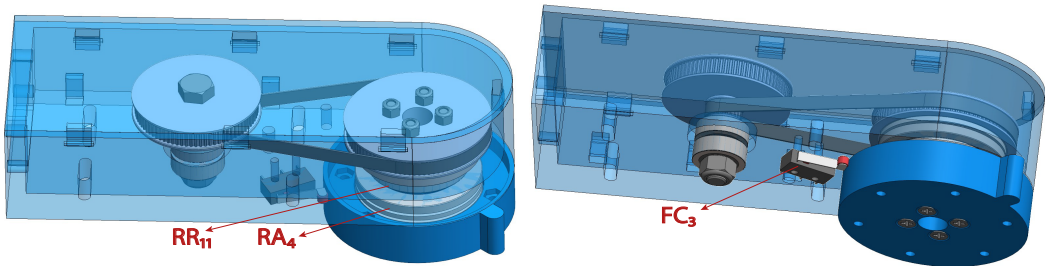
La polea dentada P_8 está acoplada mediante un rodamiento axial, RA_3, para soportar cargas axiales, y un rodamiento radial, RR_{11}, para soportar cargas radiales, al acoplamiento 2. Las poleas dentadas P_6 y P_7 están acopladas mediante dos rodamientos radiales, RR_9 y RR_{10}.

3.4.4. Acoplamiento 2

La siguiente pieza denominada como acoplamiento 2, J_2, es usada como unión entre el brazo 1 y el conjunto brazo 2, se puede observar en la Fig. 3.13. Esta pieza rota alrededor del eje z porque está unida al motor M_3 mediante un conjunto de transmisión de movimiento rotatorio (Sección 3.4.3).

En la Fig. 3.13 (a) se puede observar la posición del rodamiento axial al que está acoplado J_2, RA_4, y la posición del rodamiento radial al que está acoplado J_2 (mismo que al que esta acoplado la polea dentada P_8). En la Fig. 3.13 (b) se puede observar la posición del final de

carrera 3, FC_3.



(a) (b)

Figura 3.13: Acoplamiento 2

3.5. Conjunto brazo 2

El conjunto brazo 2, se puede observar en la Fig. 3.14, se acopla al conjunto brazo 1, más concretamente al acoplamiento J_2, mediante tornillería.

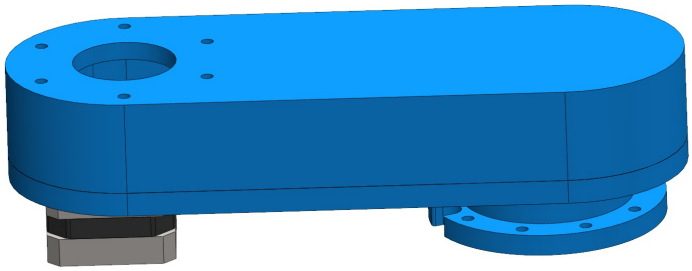
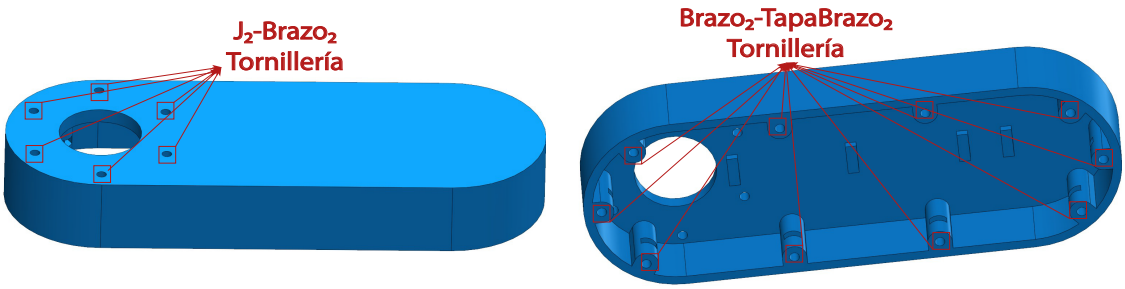


Figura 3.14: Conjunto brazo 2

Este conjunto tiene fijado el motor 4, M_4, que mediante una transmisión provoca un movimiento de rotación alrededor del eje z sobre el acoplamiento 3. Se puede dividir en cuatro partes según su finalidad: la tapa brazo 2, el brazo 2, la transmisión 3 y el acoplamiento 3.

3.5.1. Brazo 2

El brazo 2 es una pieza que se utiliza como unión entre el acoplamiento J_2 y la tapa del brazo 2, es decir, sirve como elemento de unión entre el acoplamiento 2 y el conjunto del brazo 2. En la Fig. 3.15 (a) se puede observar la posición de la tornillería que une esta pieza al acoplamiento 2 mientras que en la Fig. 3.15 (b) se observar la posición de la tornillería de unión entre el brazo 2 y la tapa del brazo 2.



(a) (b)

Figura 3.15: Brazo 2

3.5.2. Tapa brazo 2

A continuación, se observa la tapa del brazo 2 en la Fig. 3.16. Esta pieza sirve de acoplamiento al motor 4, M_4, al final de carrera 4, FC_4 y a la transmisión 3.

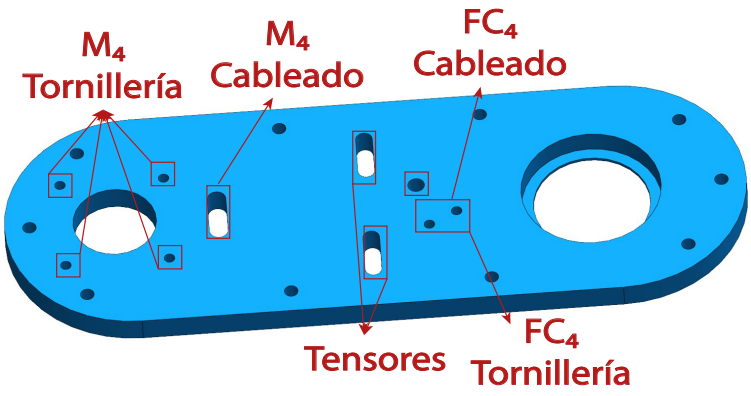


Figura 3.16: Tapa brazo 2

Además, se puede contemplar la posición de la tornillería que fija el motor M_4 a la pieza, la posición de los tensores y la posición de la tornillería de final de carrera FC_4.

3.5.3. Transmisión 3

El motor 4, M_4, tiene acoplado a su eje una polea dentada de 20 dientes, P_9, que a su vez transmite el movimiento a otra polea dentada de 90 dientes, P_10, mediante una correa. El conjunto de poleas dentadas forman una etapa de reducción de 1:4.5. Se puede observar la nomenclatura utilizada en la Fig. 3.17.

La polea dentada P_{10} está acoplada mediante un rodamiento axial, RA_5, para soportar

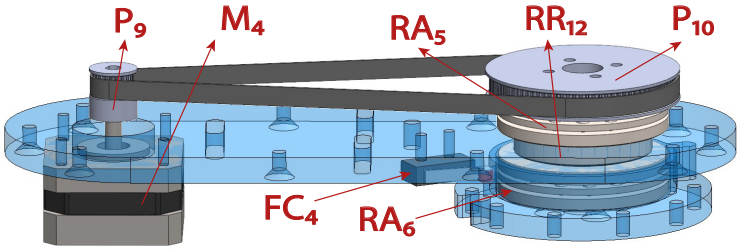
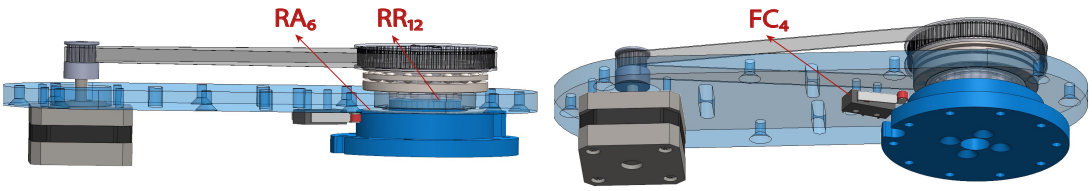


Figura 3.17: Transmisión 3

cargas axiales, y un rodamiento radial, RR_{12}, para soportar cargas radiales, al acoplamiento 3.

3.5.4. Acoplamiento 3

La siguiente pieza denominada como acoplamiento 3, J_3, es usada como unión entre el brazo 2 y la pinza, se puede observar en la Fig. 3.18. Esta pieza rota alrededor del eje z porque está unida al motor M_4 mediante un conjunto de transmisión de movimiento rotatorio (Sección 3.5.3).



(a) (b)

Figura 3.18: Acoplamiento 3

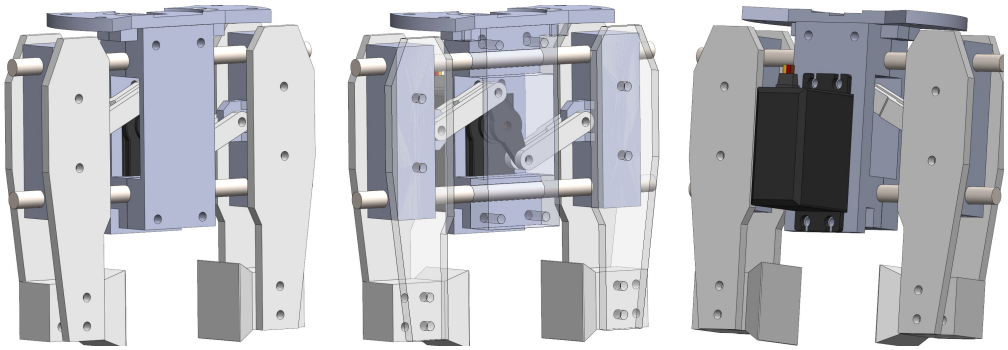
En la Fig. 3.18 (a) se puede observar la posición del rodamiento axial al que está acoplado J_2, RA_6, y la posición del rodamiento radial al que está acoplado J_2 (mismo que al que esta acoplado la polea dentada P_{10}). En la Fig. 3.18 (b) se puede observar la posición del final de carrera 3, FC_4.

3.6. Pinza

El *end-effector*[3] del presente brazo robótico es una pinza como se puede observar en la Fig. 3.19. La pinza tiene acoplado un servomotor a la parte fija que realiza un movimiento rotatorio y, además, el eje del servomotor está unido a los brazos de la pinza mediante

[3]End-effector: extremo final del robot.

eslabones. Los brazos de la pinza tienen el movimiento restringido por unas barras guía que sólo permiten que se desplacen con un movimiento lineal de translación a lo largo del eje de simetría de estas. Asimismo, los brazos de la pinza tienen limitado el movimiento por los eslabones que unen los brazos con el eje del servomotor. Por tanto, mediante las barras guía y los eslabones se consigue transformar el movimiento rotatorio del servomotor en un movimiento de translación lineal de los brazos de la pinza.



(a) (b) (c)

Figura 3.19: Pinza

Capítulo 4

Simulación

En este capítulo se exponen los resultados simulados del funcionamiento del movimiento del robot.

4.1. Preliminares

Se van a desarrollar simulaciones con el software matemático Matlab para confirmar que todos los cálculos explicados en el Capítulo 2 se cumplen de manera correcta.

Para la implementación de la simulación se utilizará una herramienta de Matlab denominada Simulink y , más concretamente, una toolbox denominada Simscape que sirve para simular el comportamiento de sistemas dinámicos de manera gráfica antes de su implementación hardware. Con Simscape se va a modelar los componentes físicos que conforman el robot con sus respectivas conexiones físicas para la creación de un entorno de simulación en el que poder comprobar el comportamiento del robot ante diversas condiciones.

4.2. Simscape

La herramienta Simscape tiene gran cantidad de componentes y conexiones físicas para poder modelar el robot. A continuación, se explican los elementos predefinidos por esta toolbox que se utilizan para poder crear el entorno de simulación.

Cabe señalar que esta herramienta tiene ciertas particularidades respecto a las piezas que ya están predefinidas como puede ser que el eje de rotación de las piezas es siempre el eje z de referencia de la misma pieza. Por tanto, es necesario entender bien estas particularidades antes de empezar a desrerollar el entorno de simulación.

4.2.1. Cables y poleas

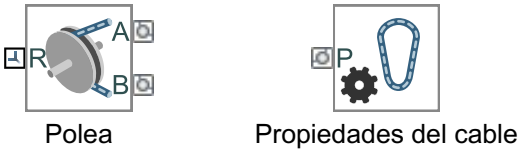
En la siguiente subsección se van a explicar las propiedades de los elementos: polea y propiedades del cable. En este caso, se utilizan para simular las transmisiones por correa que mueven las articulaciones del robot.

- Polea. Es un elemento que simula una polea la cual rota sobre su eje z de referencia. El cable que se conecte a dicha polea debe encontrarse sobre el plano x-y de la pieza. Este elemento tiene como entrada, R, el eje de rotación de la polea y como salidas dos puertos, A y B, que se unen inversamente a otros dos puertos de otra polea, es decir, representarán los puntos de tangencia de el cable con la polea, véase Fig. 4.1 (a). Además, es necesario definir el radio de la polea.

- Propiedades del cable. Este elemento se utiliza para otorgar propiedades al cable. Este elemento tiene una salida, P, que se debe unir al cable al que se le quiere aplicar las propiedades como se puede observar en la Fig. 4.1 (b). Además, es completamente necesario usar este elemento para que no se produzcan errores en las posteriores simulaciones.

Definidos estos elementos, para el desarrollo del proyecto en el cual se utilizan poleas dentadas se definen las relaciones de los radios de las poleas para que cumplan la relación real

de dientes de las poleas dentadas. De esta manera, se puede simular de manera aproximada el comportamiento de las transmisiones existentes en el robot.



(a) (b)

Figura 4.1: Cables y poleas

4.2.2. Elementos cuerpo

En la siguiente subsección se detallan los elementos más importantes para poder llevar a cabo la simulación que son los elementos cuerpo que constituyen las piezas físicamente dentro de este entorno de simulación.

- Sólido extruido. Consiste en un elemento que sirve para simular un objeto físico mediante el método de extrusión, es decir, definiendo una base y una longitud se crea una pieza.
- Sólido de revolución. Consiste en un elemento que sirve para simular un objeto físico mediante el método de revolución, es decir, definiendo una sección y un ángulo se crea la pieza. Cabe señalar que la revolución se realiza respecto al eje z de la pieza.
- Sólido de archivo. Elemento indispensable para este proyecto ya que permite exportar archivos creados mediante softwares de diseño a Simscape.

(a) (b) (c)

Figura 4.2: Elementos cuerpo

Los elementos cuerpos tienen por norma general un marco de referencia, R, que se utiliza para unir sólidos, véase Fig. 4.2. Cabe señalar que en todos estos elementos se puede calcular la inercia la geometría o la masa. Si selecciona por la geometría, además, se ha de definir si

se calcula la inercia basándose en la densidad de la pieza o en la masa total de la pieza. Por último, destacar que se pueden crear ejes locales de referencia dentro de un elemento cuerpo y también se puede modificar el aspecto visual.

4.2.3. Curvas y planos

Los elementos que se van a explicar a continuación se van a utilizar para mejorar las simulaciones dentro del entorno de Simscape.

- Spline. El siguiente elemento representa una curva en 2D o 3D obtenida de la interpolación de puntos definidos que puede ser abierta o cerrada. Como entrada, R, se tiene un marco de referencia y como salida, G, la geometría de la curva creada, véase Fig. 4.3 (a).
- Plano infinito. Es un elemento que se utiliza para crear planos con dimensiones especificadas. El plano es definido con una entrada, R, que constituye el origen y orientación del plano, ya que este es siempre perpendicular al eje de coordenadas z del marco de referencia de la entrada. También se tiene una salida, G, que representa la geometría del plano creado como se observa en la Fig. 4.3 (b).

Estos elementos van a ser utilizados en simulaciones más avanzadas como puede ser el seguimiento de una trayectoria generada mediante una spline.

(a) (b)

Figura 4.3: Curvas y planos

4.2.4. Fuerzas y pares

En la siguiente subsección se va a explicar un elemento absolutamente necesario para la realización de simulaciones de pick and place.

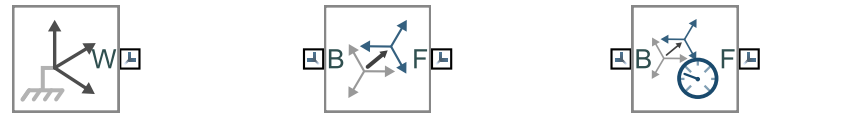
- Fuerza espacial de contacto. Es un elemento que aplica una fuerza de contacto entre dos elementos. En función de como se definan sus parámetros se pueden simular contactos rígidos o con cierta penetración entre objetos. Por tanto, se tienen dos puertos de entrada, B y F, que deben conectarse a las dos piezas que se quiere que tengan un contacto (véase Fig. 4.4)

Figura 4.4: Fuerzas

4.2.5. Ejes y transformadas

A continuación, se detallan tres elementos imprescindibles para crear el entorno de simulación que se encuentran dentro de la librería de ejes y transformadas de Simscape.

- Eje mundo. Este elemento representa el eje de la tierra, es decir, un eje sin movimiento que es la base de un modelo mecánico. Puede haber varios elementos *Eje mundo* pero todos representan el mismo. Tiene como puerto, W, un eje que se identifica como el eje conectado al eje mundo, véase Fig. 4.5 (a).
- Transformación rígida. Sirve para definir una transformación entre dos ejes. Existen dos transformaciones, de rotación y de translación. Este elemento tiene dos puertos: el eje de la base, B, y el eje seguidor, F, el cual se traslada y/o rota respecto al eje de la base. Estos puertos pueden observarse en la Fig. 4.5 (b).
- Sensor de transformación. Es un elemento que se utiliza para medir relaciones entre dos ejes, véase Fig. 4.5 (c)



(a) (b) (c)

Figura 4.5: Ejes y transformadas

4.2.6. Articulaciones

En la siguiente subsección se va explicar el funcionamiento de los bloques de articulaciones de Simscape que se han utilizado para el desarrollo del proyecto.

- Articulación prismática. Esta articulación tiene un grado de libertad que permite la translación del eje seguidor respecto al eje z de la base, véase Fig. 4.6 (a).

- Articulación de revolución. Esta articulación tiene un grado de libertad que permite la rotación del eje seguidor respecto al eje z de la base, véase Fig. 4.6 (b).
- Articulación cilíndrica. La articulación cilíndrica tiene dos grados de libertad que permiten la translación y rotación del eje seguidor respecto al eje z de la base, véase Fig. 4.6 (c).
- Articulación cartesiana. Esta articulación tiene tres grados de libertad que permite la translación del eje seguidor respecto al eje z-y-z de la base, véase Fig. 4.6 (d).
- Articulación 6DOF. Esta articulación tiene seis grados de libertad que permite la translación y rotación del eje seguidor respecto al eje z-y-z de la base, véase Fig. 4.6 (e).
- Articulación husillo. La articulación husillo tiene un grado de libertad que permite la translación del eje seguidor respecto al eje z de la base cuando rota el eje seguidor respecto al eje z, véase Fig. 4.6 (f).

Todos los bloques de articulaciones cuenta con una entrada, B, que referencia a un eje base y una salida, F, que representa el eje seguidor. El eje seguidor es el que recibe la acción de la articulación respecto al eje z de coordenadas de la base. Como ya se comentó al principio de la Sección 2.2, los movimientos que provocan las articulaciones son siempre respecto al eje z. Todas las articulaciones se pueden controlar mediante los parámetros que las definen como puede ser un actuador.

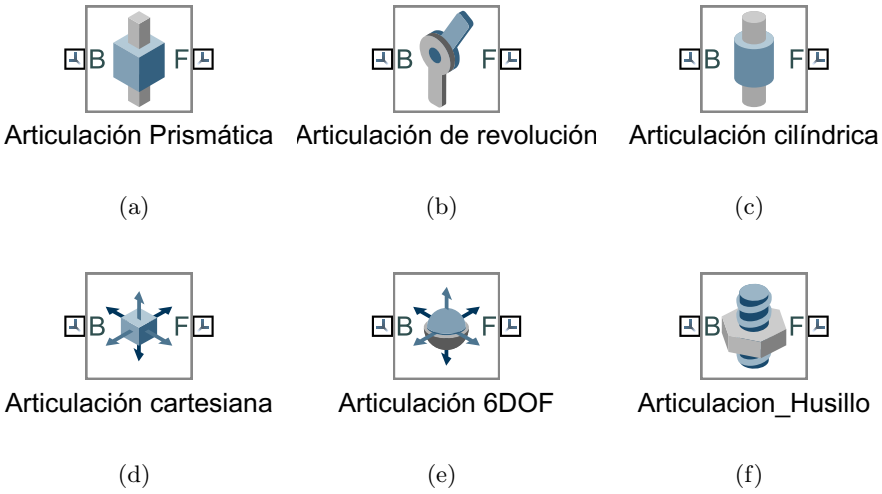


Figura 4.6: Articulaciones

4.2.7. Configuración

En la siguiente subsección se detalla un elemento imprescindible para poder realizar simulaciones en el entorno de Simscape.

- Configuración del mecanismo. Este elemento es necesario para establecer parámetros que afectan a todas las piezas de la simulación. Este elemento tiene un puerto, C, que se conecta a un eje, generalmente, al eje mundo. En la Fig. 4.7 se puede observar dicho elemento.

En este proyecto se va a utilizar este elemento para ejercer la fuerza de la gravedad en todos los bloques de piezas.

Figura 4.7: Configuración

4.3. Implementación de entorno de simulación

En esta sección se va a detallar como se ha creado el entorno de simulación en Simscape para la realización de las simulaciones. Para ello es necesario conocer el funcionamiento del programa y la utilidad de los elementos a utilizar (vistos en las subsecciones 4.2).

Para poder implementar el entorno de simulación es necesario tener archivos de tipo .step de todas las piezas que forman el robot y, además, se debe tener claro por que pieza se va a empezar la implementación del robot.

Se comienza con la pieza *Base* y se procederá a añadir pieza a pieza con sus respectivas relaciones para construir un entorno de simulación fiable. En la Fig. 4.11 se puede observar el entorno de simulación finalizado. Para una fácil comprensión se han creado grupos parecidos a los detallados en el Capítulo 3. No son exactamente iguales a los mencionados en dicho capítulo ya que sería más complicado entender el funcionamiento de todos los bloques dentro del entorno. Además, para una mayor facilidad de comprensión se han creado máscaras para poder visualizar fotos de las piezas o conjunto de estas en cada bloque.

En la Fig. 4.8 se puede observar la forma del elemento sólido de archivo que, en este caso, exporta la pieza denominada como *Base* con la máscara que permite visualizar una foto de esta pieza. Este elemento está unido a diferentes piezas mediante diferentes transformaciones, véase Fig. C.1.

A continuación, en la Fig. 4.9 se puede observar el contenido dentro de la pieza *Base*. Se puede apreciar que se han creado ocho ejes de referencia con los cuales relacionar dicha pieza con otras piezas. Además, se puede percibir que cada unos de estos ejes de referencia creados dentro de la pieza esta definido con el nombre de la pieza a la que se debe unir para facilitar la comprensión. Si se entra dentro de la configuración del elemento *Base* y se

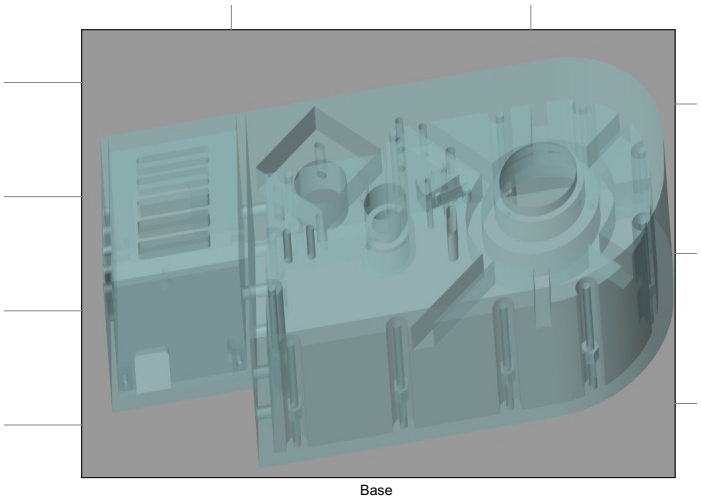


Figura 4.8: Base (máscara)

implementa al opción de visualizar los ejes, se podrían observar las posiciones de los ejes con su nomenclatura en la pieza, véase Fig. 4.10.

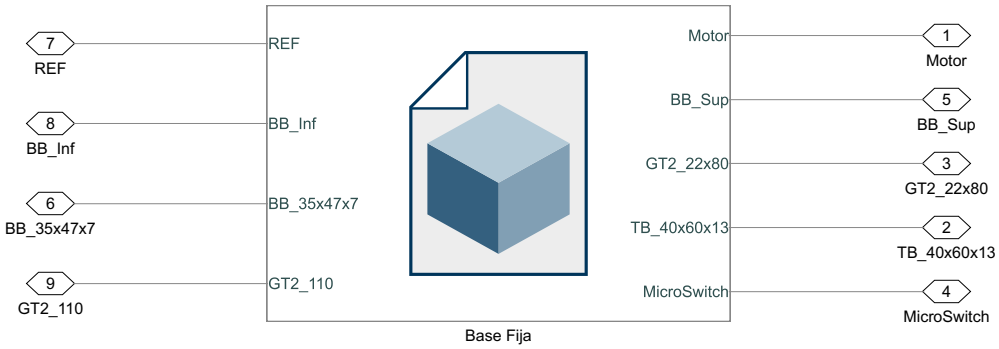


Figura 4.9: Base (elemento)

De esta manera, se puede observar claramente como está definido cada eje de una pieza y como se relaciona con el eje de otra pieza mediante esta nomenclatura. Por tanto, siguiendo el modelo de esta pieza se procede a implementar el entorno de simulación completo, tendiendo en cuenta que la pieza *Base* es la que está relacionada directamente con el *Eje mundo* y, por consiguiente, es la pieza de la que se parte en el proceso de creación del entorno.

En el Anexo C se encuentran todos los grupos que forman el entorno de simulación:

- **Grupo Base.** Se puede observar su contenido en la Fig. C.1.
- **Grupo Barras.** Se puede observar su contenido en la Fig. C.2.
- **Grupo Barra Movil.** Se puede observar su contenido en la Fig. C.3.
- **Grupo Brazo 1 - Parte 1.** Se puede observar su contenido en la Fig. C.4.
- **Grupo Brazo 1 - Parte 2.** Se puede observar su contenido en la Fig. C.5.
- **Grupo Brazo 2.** Se puede observar su contenido en la Fig. C.6.
- **Grupo Pinza.** Se puede observar su contenido en la Fig. C.7.

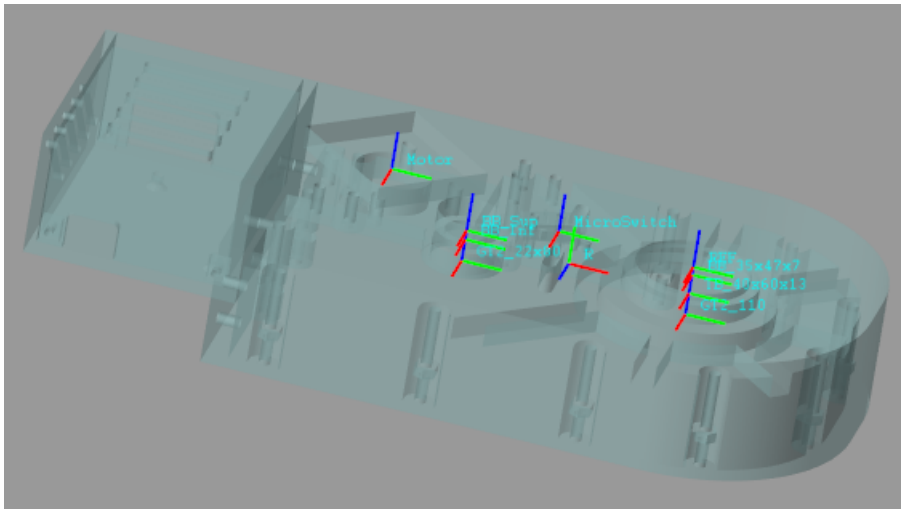


Figura 4.10: Base (ejes)

En la Fig. 4.12 se puede apreciar como se vería el robot completo en el entorno de simulación. Cabe destacar que se ha modificado la opacidad de algunas piezas para poder observar más detalladamente los mecanismos internos. Esto es sólo una modificación visual que no influye en la dinámica del robot.

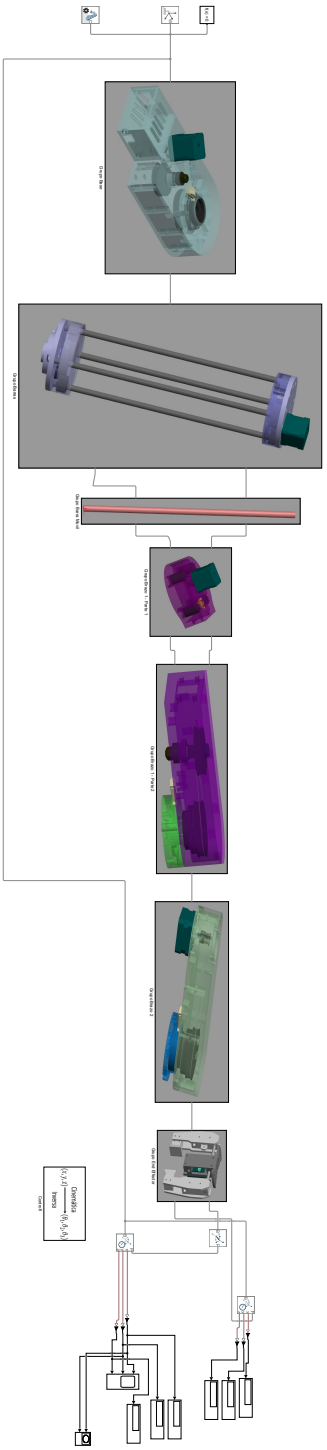


Figura 4.11: Entorno de simulación

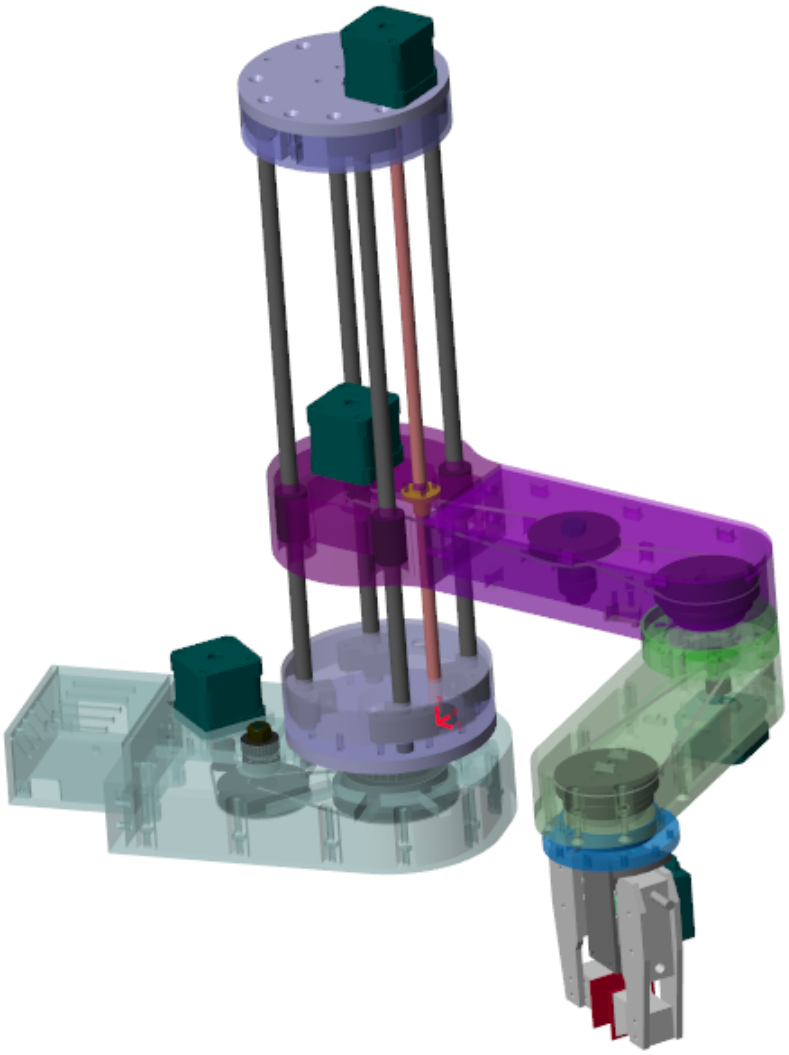
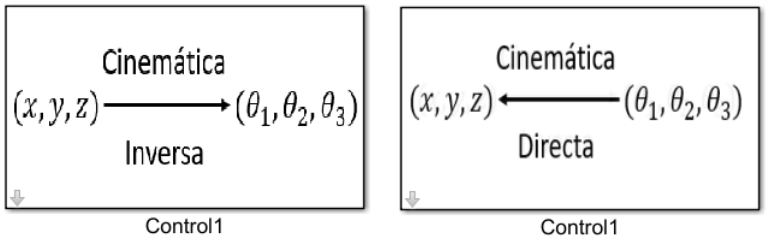


Figura 4.12: Entorno de simulación - Robot

4.4. Interfaz de simulación

Tras crear el entorno de simulación en la sección anterior surge la necesidad de crear una pequeña interfaz de interacción para la facilitación de la utilización del entorno por parte del usuario. Para ello, se procede a crear una máscara que contenga los bloques que definen los movimientos a realizar por parte del robot. Esta máscara no sólo se utiliza para mostrar una imagen en el bloque (utilizada en la sección anterior para mostrar fotos de las piezas) sino que además se usa para implementar una pequeña interfaz.

En la Fig. 4.13 se puede observar como varía la imagen de la interfaz en función de que cinemática se quiera utilizar: directa Fig. 4.13 (b) e inversa Fig. 4.13 (a).



(a) (b)

Figura 4.13: Control

Si se hace hace doble click sobre el bloque de control se puede acceder directamente a la interfaz con lo que se puede observar una imagen del robot con sus respectivas coordenadas articulares, véase Fig. 4.14. Además se puede observar los dos modos que existen con la interfaz: cinemática y trayectorias. En el modo cinemática se puede hacer movimientos del robot con diferentes parámetros ajustables en función de si se selecciona la cinemática directa o inversa. En el modo trayectorias se realizan trayectorias predefinidas para observar el funcionamiento del robot.

En la parte inferior de esta interfaz se encuentra la parte donde el usuario puede interactuar para modificar las simulaciones a realizar, véase Fig. 4.15. En la Fig. 4.15 (a) se puede observar los parámetros modificables para la cinemática directa que son las coordenadas articulares de los motores. En la Fig. 4.15 (b) se puede observar los parámetros modificables para la cinemática inversa que son las coordenadas de la posición final del robot.

A continuación, se procede a detallar el funcionamiento interno del la interfaz de control. En la Fig. 4.16 se puede observar el esquema de control del entorno. Mediante una variable, Var_Modo, se controla el modo de funcionamiento de la simulación, seleccionando entre trayectorias y cinemáticas. En ambos modos se obtienen las coordenadas articulares de los motores en grados que se suman a las condiciones iniciales para cada articulación. Una vez hecho esto, se pasa de grados a radianes mediante un bloque y se multiplica cada coordenada articular por la inversa de la etapa de reducción específica de cada coordenada. Por último, se limita el cambio de cada coordenada articular mediante dos bloques para evitar cambios

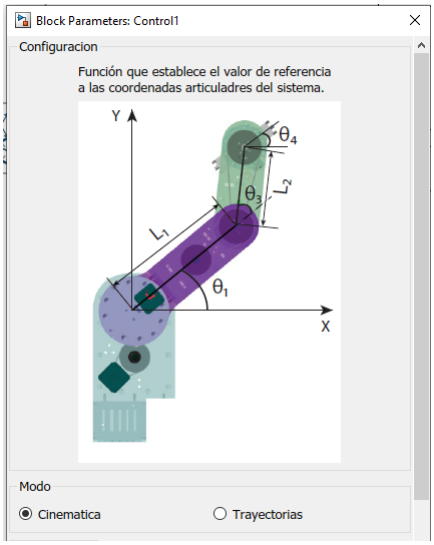


Figura 4.14: Interfaz

instantáneos que en la realidad no pueden producirse.

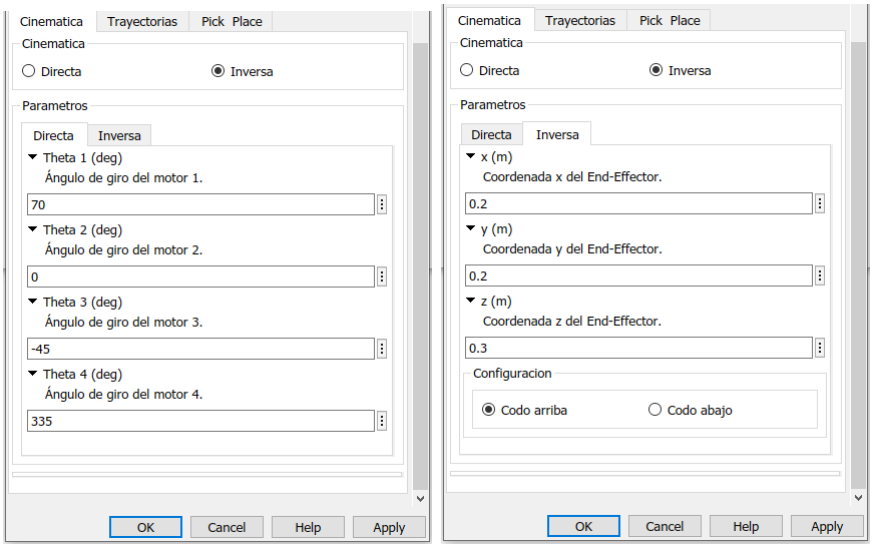
4.4.1. Trayectorias

En la Fig. 4.17 se puede apreciar el contenido del bloque trayectorias comentado en la sección anterior. A continuación, se procede a explicar detalladamente su funcionamiento.

Por un lado, se generan una señales senoidales discretas mediante el periodo de muestreo del sistema. Estas señales se desplazan del origen de coordenadas mediante el incremento con unas constantes. Cada señal senoidal representa una coordenadas del sistema de referencia, x e y. Por la otra parte, se generan dos señales senoidales discretas iguales a las anteriores pero adelantadas un periodo de muestreo. Las cuatro señales senoidales son introducidas en una función CI que calcula la orientación que ha de adquirir la pinza a lo largo de la trayectoria.

Por otro lado, mediante las primeras señales senoidales descritas se calcula las coordenadas articulares de los motores. En esta simulación se ha restringido el desplazamiento vertical del brazo robótico, es decir, se mueve por el plano X-Y.

En la Fig. 4.18 se puede observar una serie de imágenes que muestran al robot siguiendo la trayectoria predefinida. A continuación, en la Fig. 4.19 se observan los valores que adquieren las coordenadas articulares durante la realización del seguimiento de la trayectoria. Por último. en la Fig. 4.20 se puede observar una comparativa de las posiciones reales generadas con las ondas senoidales y los valores reales de posición final de la pinza obtenidos. En todos ellos se puede ver claramente que el error cometido es prácticamente nulo. Además, se ver la forma de la trayectoria recorrida por la pinza durante en la simulación, véase Fig. 4.20 (d).



(a) (b)

Figura 4.15: Cinemática

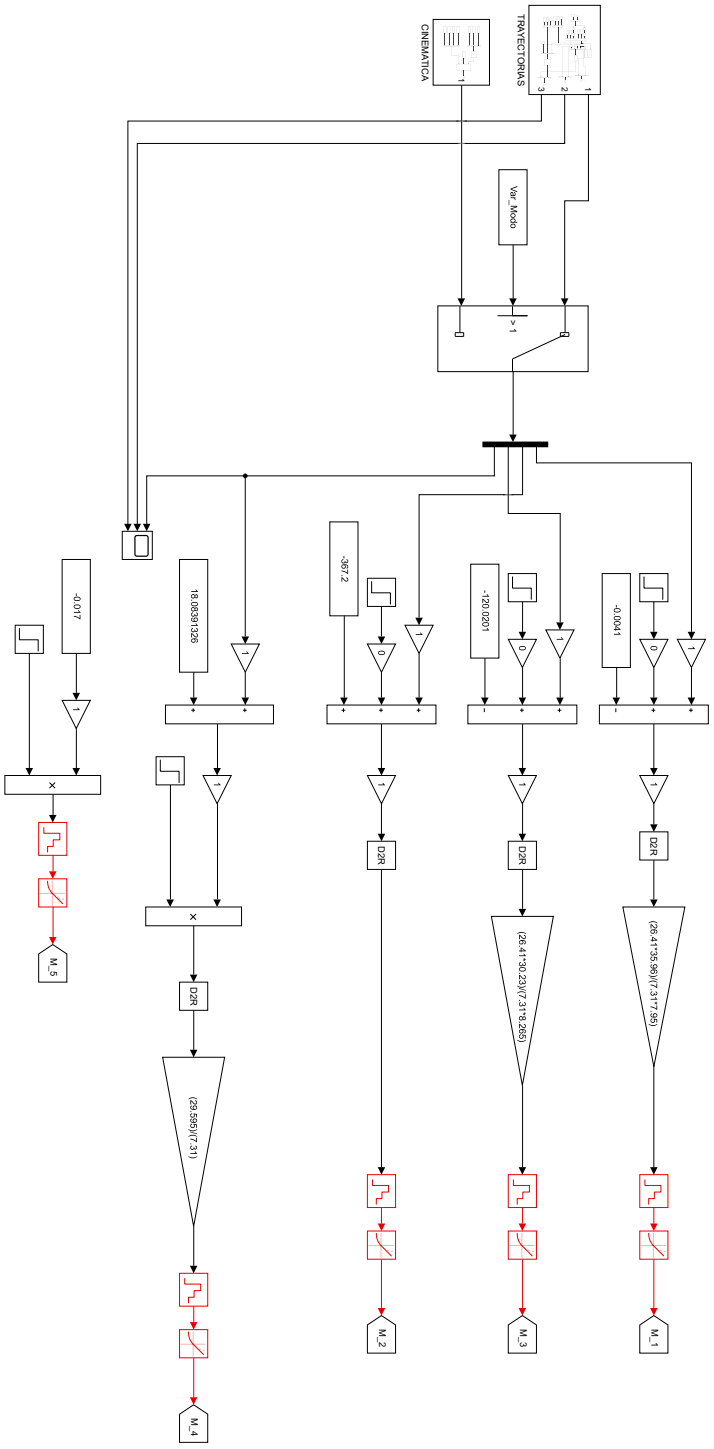


Figura 4.16: Interfaz - Control

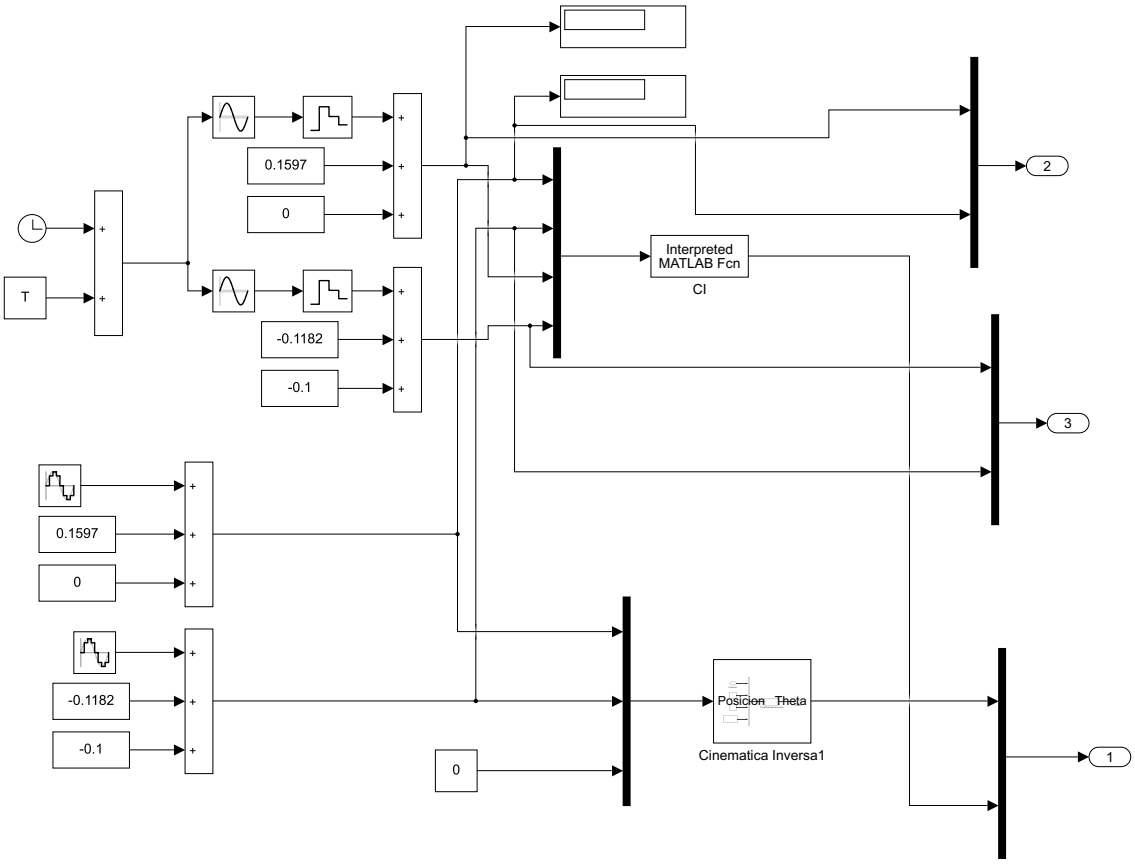
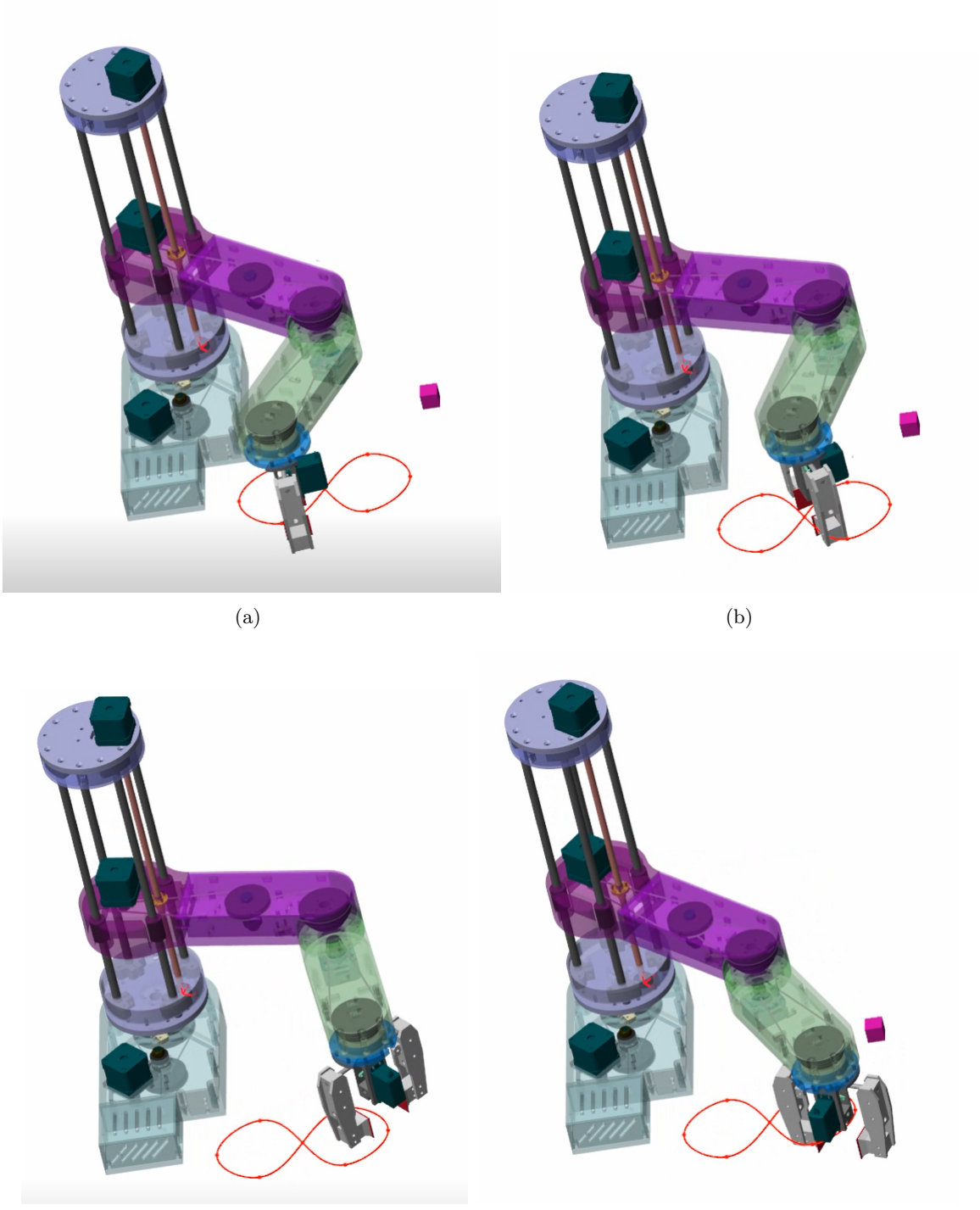


Figura 4.17: Interfaz - Trayectorias



(a) (b)

(c) (d)

Figura 4.18: Simulación Trayectorias

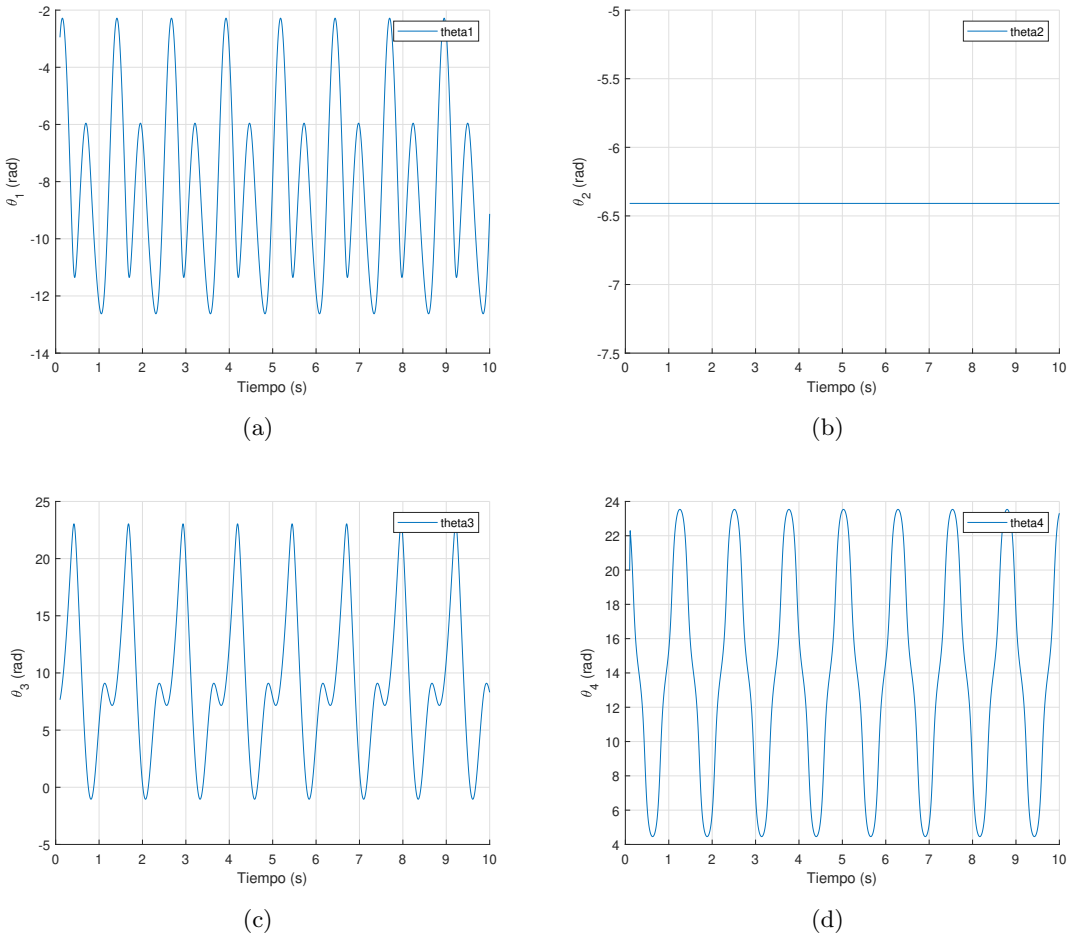


Figura 4.19: Simulación Trayectorias - Coordenadas articulares

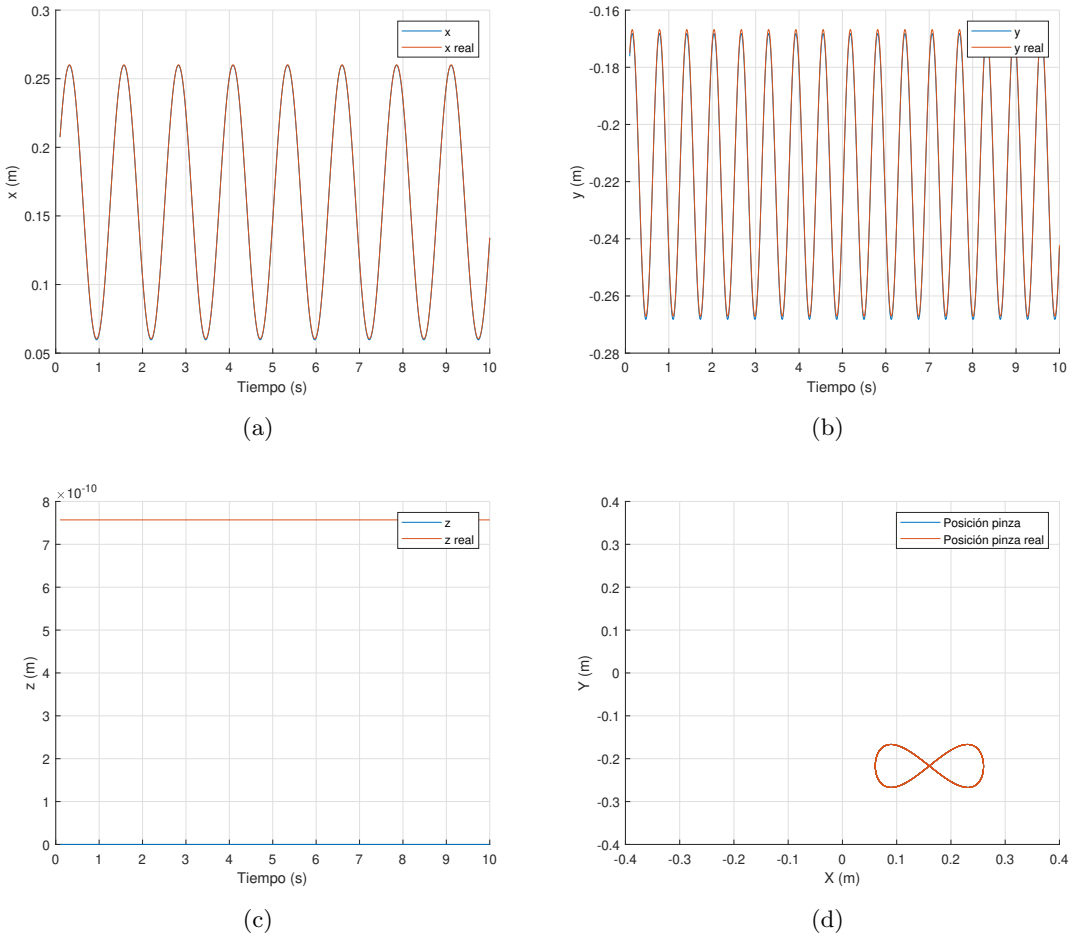


Figura 4.20: Simulación Trayectorias - Posiciones

4.5. Cinemática

En la siguiente sección se detalla el funcionamiento de la cinemática. Primeramente, se observa el bloque cinemática, Fig. 4.21, dentro del bloque de control de la interfaz, véase Fig. 4.16. En la Fig. 4.21 se puede ver que mediante un selector el usuario es capaz de seleccionar el tipo de control que se va a realizar.

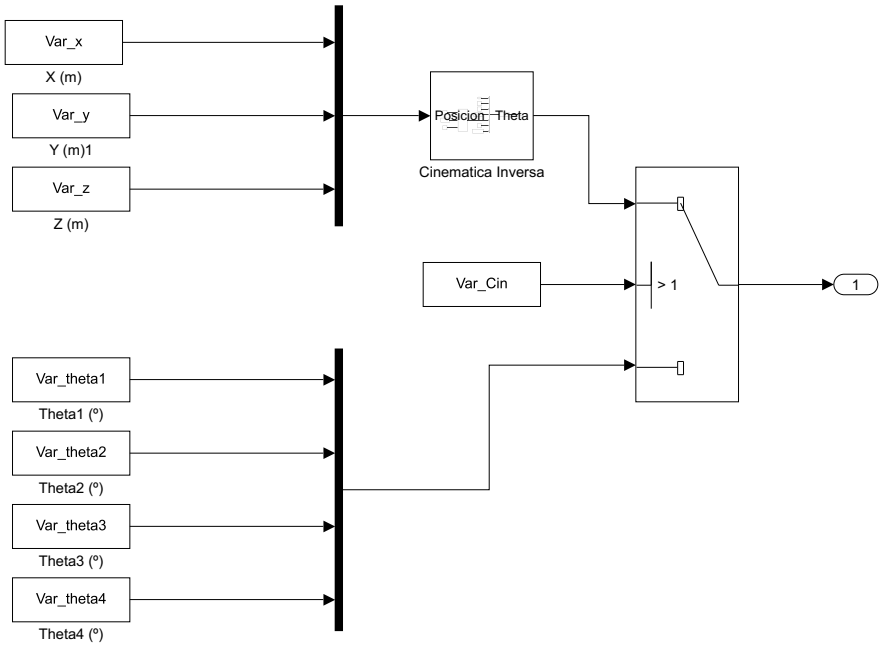


Figura 4.21: Interfaz - Cinemática

Si el usuario ha seleccionado la cinemática inversa es necesario trasformar las coordenadas (x, y, z) en coordenadas articulares de los motores. Para ello, se llama al bloque de cinemática inversa, véase Fig. 4.22. En este bloque se aprecia como se agrupan las variables necesarias para resolver el problema cinemático inverso. Además, se puede apreciar también un selector en el que el usuario selecciona el tipo de configuración que quiere para que el robot alcance la posición especificada. Cabe señalar que se puede ver la función que resuelve el problema cinemático inverso en el Anexo B.2.1.

A continuación, se procede a realizar un movimiento mediante la cinemática inversa ya que es la cinemática utilizada para aplicaciones de pick and place. El usuario introduce

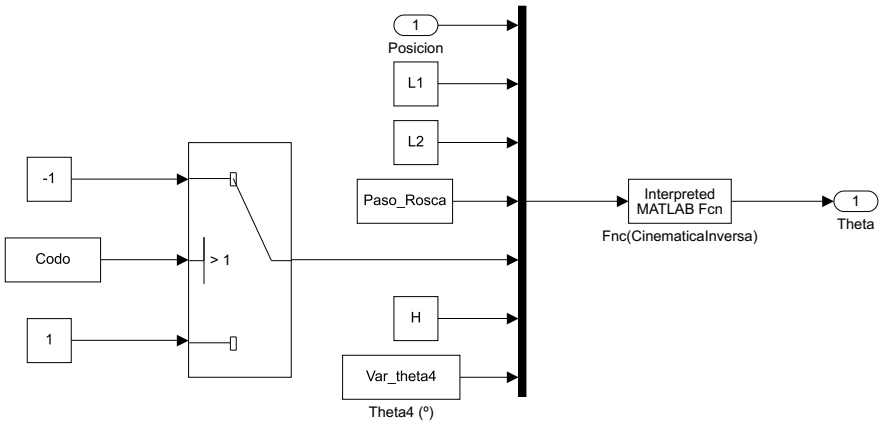
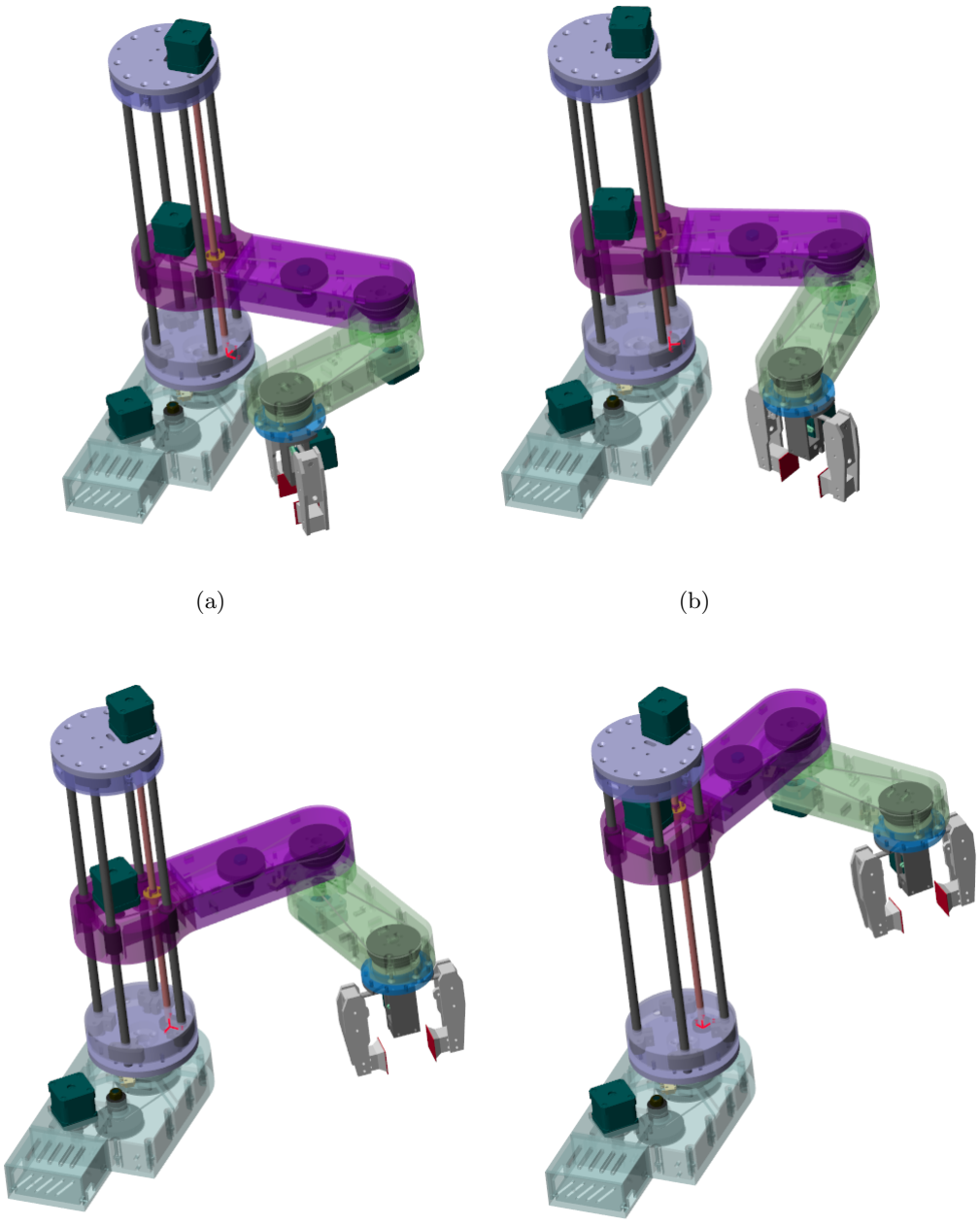


Figura 4.22: Interfaz - Cinemática inversa

los parámetros de la posición a la que se quiere llegar con la pinza y se pulsa para inicial la simulación. En la Fig. 4.23 se puede observar la consecución del movimiento que ha efectuado el robot tras la simulación.

En la Fig. 4.24 se observan los resultados de las coordenadas articulares tras la simulación. Las coordenadas articulares van evolucionando en el tiempo según las limitaciones dispuestas hasta alcanzar el valor de cada uno calculado.

En la Fig. 4.25 se puede observar la evolución de las variables que realmente interesa apreciar que son las variables de las posiciones de referencia absolutas de la pinza. Se puede apreciar que la coordenada x (4.25 (a)) e y (4.25 (b)) dependen de las articulaciones θ_1 y θ_3 y, por tanto, se observa como por ejemplo la coordenada x comienza ascendiendo de valor en vez de aproximarse directamente al valor de la coordenada especificado. En contraposición a lo comentado respecto a las coordenada x e y, la coordenada z (4.25 (c)) sólo depende únicamente de la coordenada articular θ_2 por lo que se puede apreciar una evolución más lineal que la de las otras dos coordenadas. En la (Fig. 4.25 (d)) se muestra la evolución de la pinza en el sistema tridimensional. Como se puede apreciar, durante los primeros instantes



(a) (b)

(c) (d)

Figura 4.23: Simulación Inversa

de simulación, la posición de la pinza varía respecto a los tres ejes hasta aproximadamente los 0.2s. Tras este tiempo, únicamente varía la coordenada articular θ_2, es decir, en este caso se desplaza verticalmente en sentido ascendente.

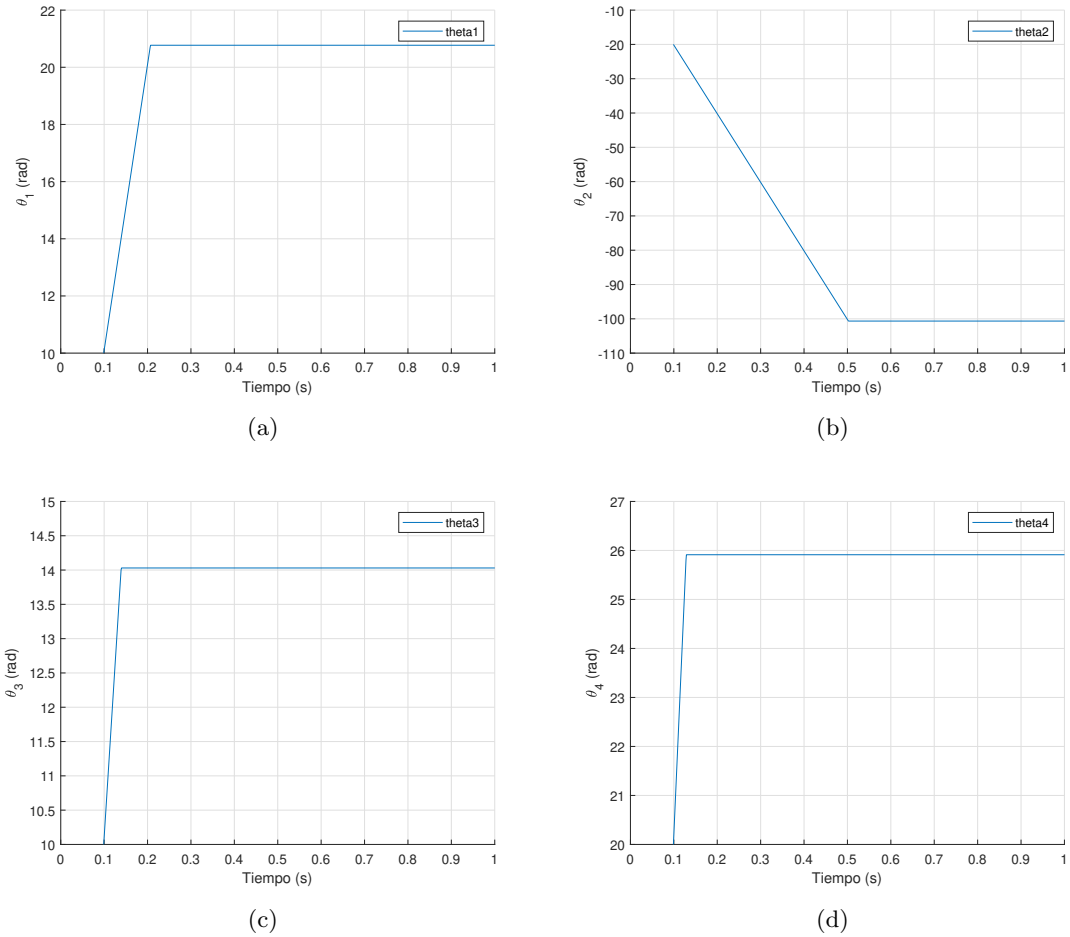
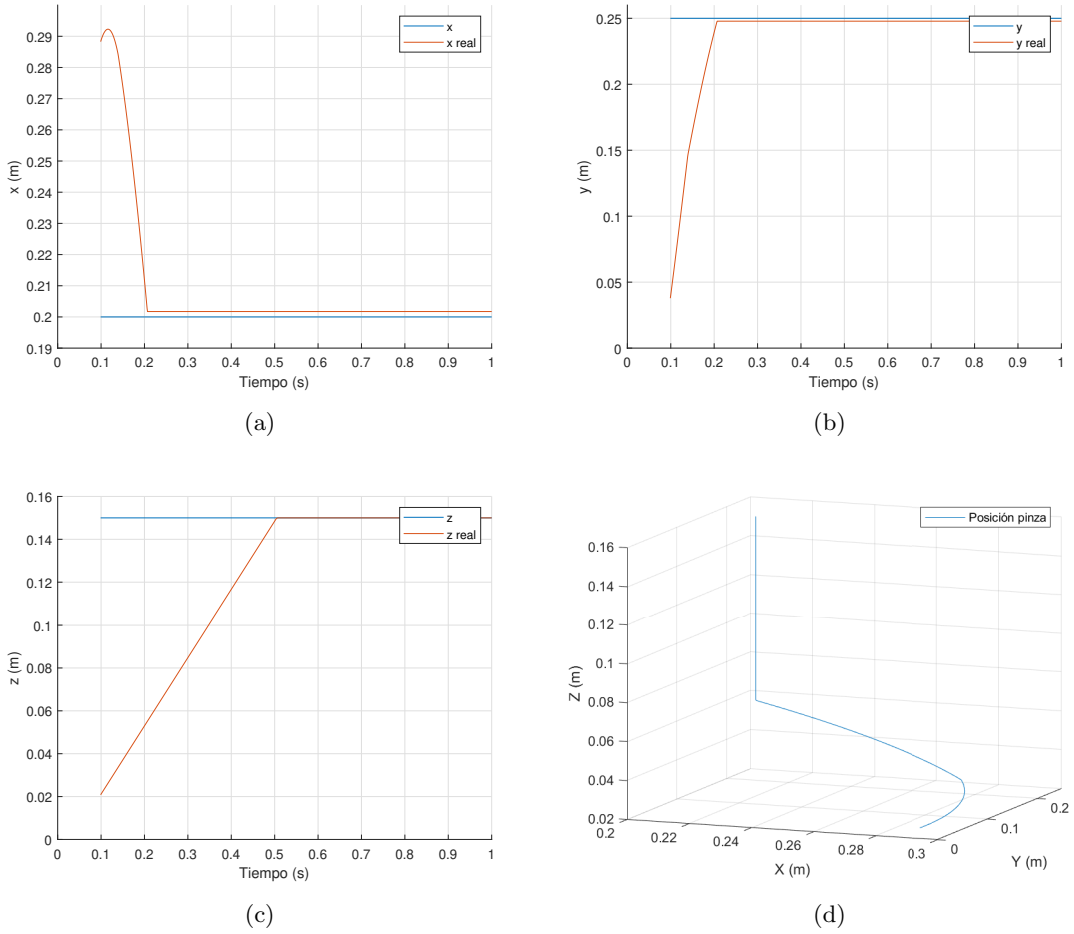


Figura 4.24: Simulación Inversa - Coordenadas articulares



(a) (b)

(c) (d)

Figura 4.25: Simulación Inversa - Posiciones

Capítulo 5

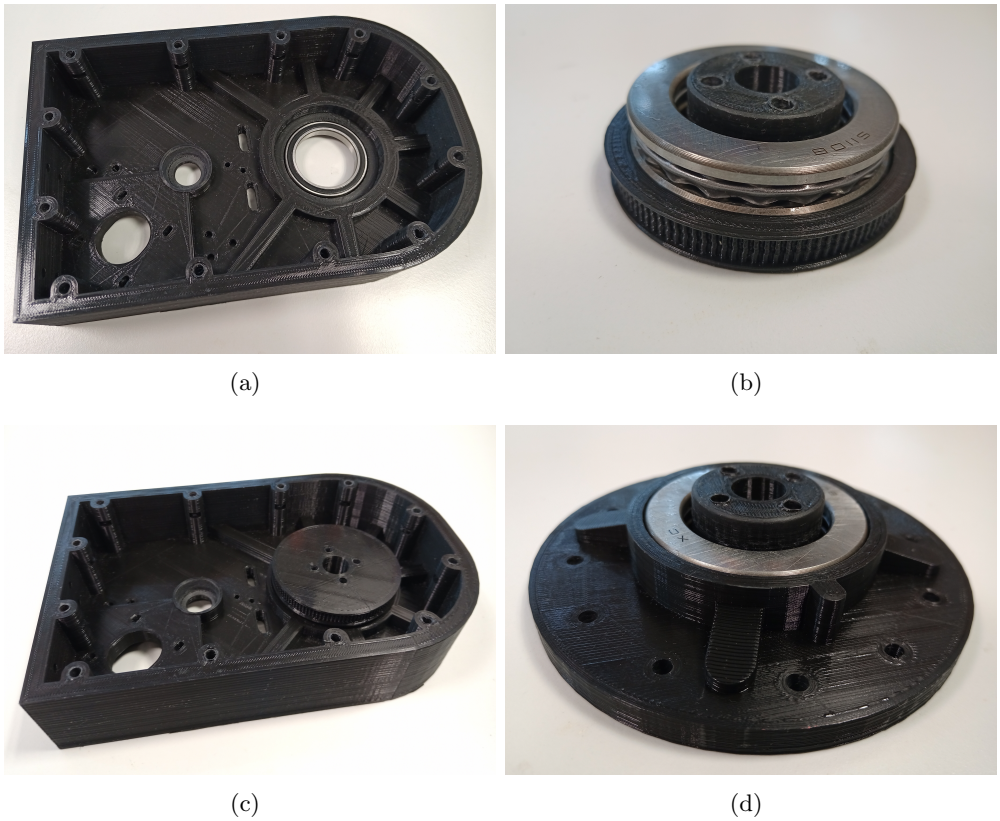
Implementación real

En este capítulo se desarrolla el montaje mecánico del brazo robótico. Posteriormente, se procede a realizar movimientos del mismo.

5.1. Montaje mecánico

En la siguiente sección se procede a desarrollar paso a paso el montaje del brazo robótico. Para ello se hace referencia a la nomenclatura detallada en el Capítulo 3.

En la Fig. 5.1 (a) se observa como se ha acoplado mediante presión el rodamiento radial RR_3 a la base. Tras esto, se junta la polea dentada P_4 con el rodamiento axial RA_1 (véase Fig. 5.1 (b)) y se acopla mediante presión al rodamiento RR_3 por la parte inferior de la base, véase 5.2 (a). A continuación, se junta el acoplamiento J_1 con el rodamiento axial RA_2 (Fig. 5.2 (b)) y se acopla por la parte superior de la base al rodamiento RR_3 mediante presión. En este último paso es necesario alinear los orificios del acoplamiento J_1 con los de la polea P_4 ya que ambas piezas son unidas mediante unos tornillos de métrica 5 de 55mm y tuercas autoblocantes.



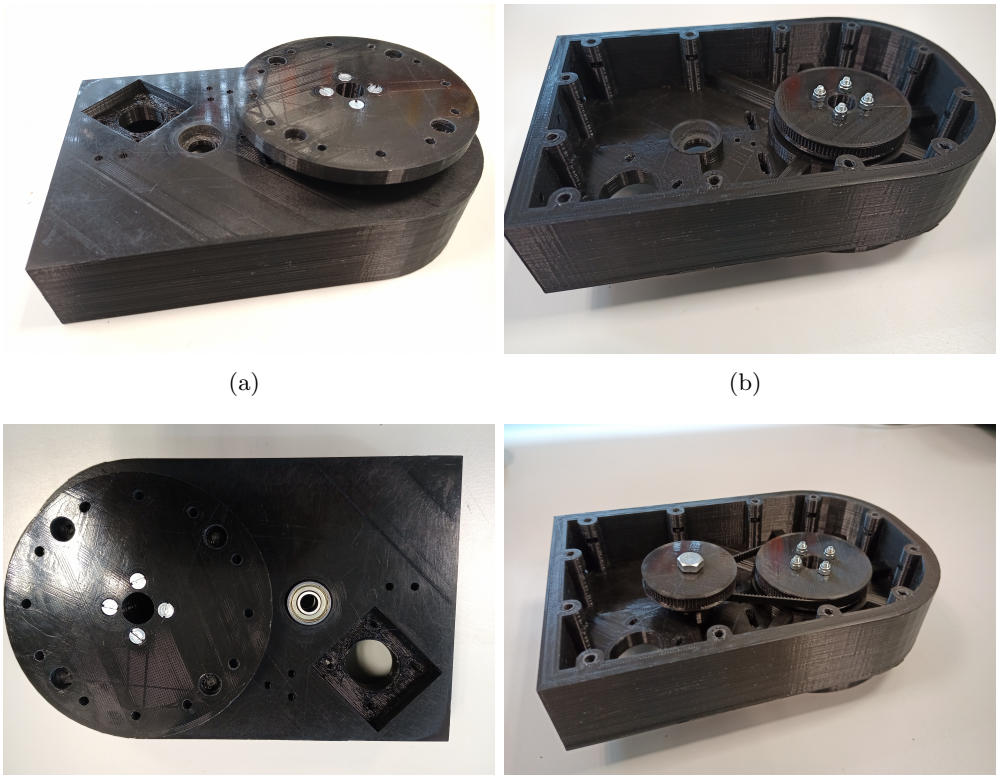
(a) (b)

(c) (d)

Figura 5.1: Montaje - Paso 1

Se continua con el acoplamiento de los rodamientos radiales RR_1 y RR_2 mediante presión, véase Fig. 5.2 (c). Se une el conjunto de poleas P_2-P_3 a un tornillo de métrica 8 de 45mm. Se utiliza una correa GT2 de 300mm para establecer la unión de la polea P_2 a la polea P_4. Tras esto, se procede a introducir el tornillo de métrica 8 por el eje de los rodamientos RR_1 y RR_2 y se fija mediante una tuerca autoblocante, véase Fig. 5.2 (d). Durante este paso, se

debe asegurar que las poleas queden sobre el mismo plano horizontal, para ello se hace uso de arandelas que modifican la altura de la polea P_2. Cabe señalar que se pueden utilizar tensores para ajustar la tensión de la correa.



(a) (b)

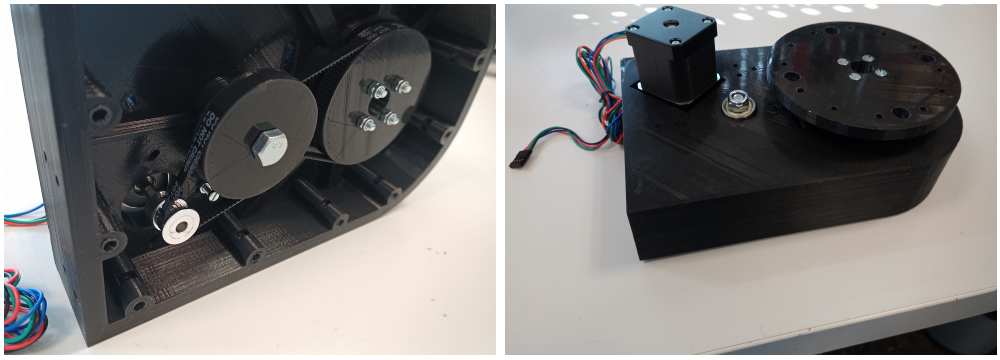
(c) (d)

Figura 5.2: Montaje - Paso 2

A continuación, se acopla la polea P_1 al eje del motor M_1 de forma que quede en el mismo plano horizontal que la polea P_3. Ambas poleas son unidas mediante una correa GT2 de 200mm, véase Fig. 5.3 (a). El motor M_1 es fijado mediante tornillos de métrica 3 y tuercas. Además, el final de carrera FC_1 es fijado en su posición mediante tornillos de métrica 2,5. Se puede observar el conjunto descrito en la Fig. 5.3 (b).

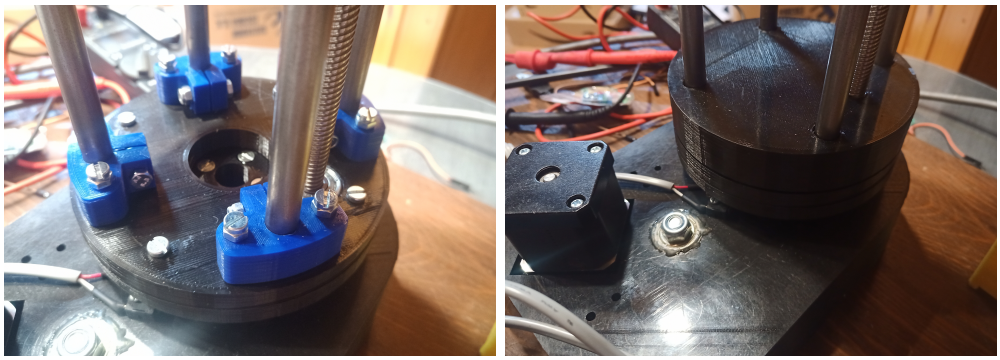
El siguiente paso es fijar la placa inferior y las abrazaderas Ab_{1-4} al acoplamiento J_1 mediante tornillos de métrica 4 y tuercas autoblocantes. Tras esto, se acopla el rodamiento radial RR_4 al orificio de la placa inferior mediante presión. Además, se fija las cuatro barras guías a las abrazaderas mediante tornillos de métrica 4 y tuercas autoblcantes, véase la Fig. 5.4 (c). Después de fijar las barras se procede a colocar la tapa de la placa inferior que se trata de una pieza meramente decorativa como se puede observar en la Fig. 5.4 (d).

Una vez llegado a esta parte del montaje se procede a realizar los agujeros de los tornillos que fijan la base a una plataforma, en este caso, se ha seleccionado una de madera. Para realizar los agujeros con precisión se imprime el plano de la base y se adhiere a la paltaforma,



(a) (b)

Figura 5.3: Montaje - Paso 3



(a) (b)

Figura 5.4: Montaje - Paso 4

véase Fig. 5.5 (e). Tras esto, se realizan los taladros en las posiciones de los tornillos quedando los orificios en sus respectivas posiciones como se puede observar en la Fig. 5.5 (f). A continuación, se taladran los orificios de la base con una broca macho de rosco de métrica 5. Una vez hecho esto, se fija la base a la plataforma mediante tornillos de métrica 5 de 40mm. Se puede observar en la Fig. 5.5 el estado del brazo robótico en este punto del montaje.

La siguiente etapa consiste en el montaje del brazo 1, brazo 2 y pinza del brazo robótico. Se comienza con el conjunto brazo 1, se une la plataforma de montaje con el brazo 1 mediante tornillos de métrica 5 de 30mm. Además, se acoplan mediante presión los rodamientos radiales RR_5, RR_6, RR_7 y RR_8 en la plataforma de montaje y, los rodamientos radiales RR_9, RR_10 y RR_11 en sus respectivos huecos del brazo 1 como se puede observar en la Fig. 5.14 (a). A continuación, se acopla la polea dentada conjunta P_6 y P_7 de manera análoga a como se hizo con las poleas P_2 y P_3. Tras esto, se fija al eje del motor M_2 la polea dentada P_5 de tal manera que al acoplar el motor en la plataforma de montaje el conjunto de poleas $P_5 - P_6$ queden en el mismo plano horizontal. Ambas poleas son unidas mediante una correa GT2 de 400mm, véase Fig. 5.14 (b).

(a) (b)

Figura 5.5: Montaje - Paso 5

(a) (b)

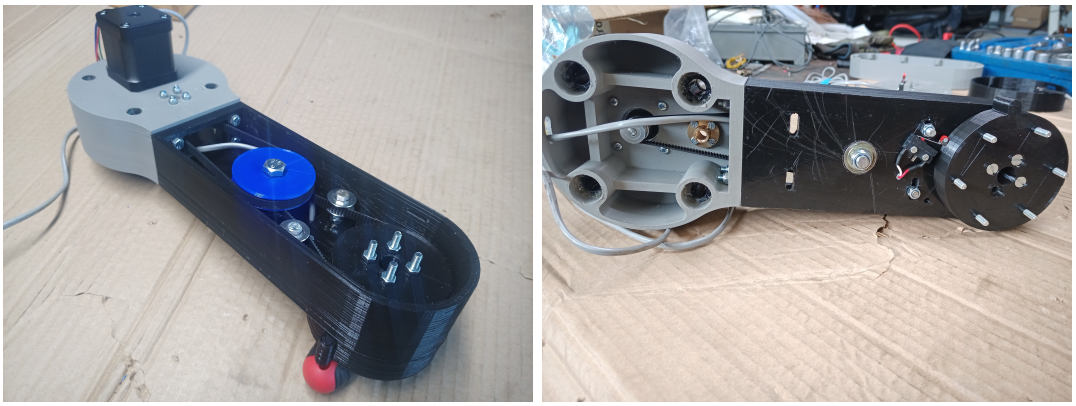
Figura 5.6: Montaje - Paso 7

El siguiente paso consiste en el acoplamiento de la polea P_8 junto al rodamiento axial RA_3 mediante presión en el rodamiento radial RR_{11}, véase Fig. 5.7 (c). Después, se fijan tornillos de métrica 4 de 25mm al acoplamiento J_2 y se une al rodamiento axial RA_4 como se puede observar en la Fig. 5.7 (d). Señalar que se utiliza una correa GT2 de 300mm para transmitir el movimiento de la polea P_7 a P_8. Posteriormente, se une el acoplamiento J_2 al rodamiento radial RR_{11}, teniendo especial cuidado de alinear los orificios de los tornillos del acoplamiento J_2 con los orificios de los tornillos de la polea P_8. Tras esto, se unen ambas piezas mediante tornillos de métrica 5 de 50mm y tuercas autoblocantes como se puede observar en la Fig. 5.8 (e). Seguidamente se procede a fijar el final de carrera FC_3 en su posición mediante tornillos de métrica 2.5mm y sus respectivas tuercas, véase Fig. 5.8 (f). Cabe destacar que en este conjunto de brazo robótico se han usado tensores para aumentar la tensión de las correas (véase Fig. 5.8 (e)). Por último, se fija el brazo 2 al acoplamiento J_2 mediante los tornillos fijados en dicho acoplamiento y tuercas autoblocantes como se puede observar en la Fig. 5.9.

Se procede al montaje de la transmisión del último motor M_4. Se introduce mediante presión el rodamiento radial RR_{12} en el hueco correspondiente de la tapa del brazo 2 como

(a) (b)

Figura 5.7: Montaje - Paso 8



(a) (b)

Figura 5.8: Montaje - Paso 9

se puede observar en la Fig. 5.10 (a). Se une el rodamiento axial RA_5 con la polea P_{10} y el rodamiento axial RA_6 con el acoplamiento J_3, véase Fig. 5.11 (c) y (d). Ambos conjuntos se fijan mediante presión en el rodamiento radial RR_{12}, teniendo especial cuidado de alinear los orificios de los tornillos que unen las piezas P_10 y J_3 como se observa en la Fig. 5.10 (b) . Los tornillos utilizados son de métrica 4 de 45mm. A continuación, se fija la polea P_9 al eje del motor M_4 de tal forma que quede en el mismo plano horizontal a la polea P_{10}. Ambas poleas se unen mediante una correa GT2 de 400mm, véase Fig. 5.12 (e). También se acopla el final de carrera FC_4 en su respectiva posición. A continuación, se acopla la pinza con tornillos de métrica 3 de 20mm al acoplamiento $J3$, véase Fig. 5.13 (b).

Por último se une el conjunto de la tapa del brazo 2 y la pinza con el brazo 2 mediante tornillos de métrica 3 de 30mm. Hay que tener especial cuidado en la colocación de los cables, véase Fig. 5.13. Una vez completada esta parte, se puede colocar la plataforma de montaje

sobre las barras guía como se observa en la Fig. 5.16. Se puede apreciar en ambas figuras como se han colocado unas regletas para el posicionamiento del cableado con el fin de evitar rozamientos con las partes mecánicas móviles del robot.

Sólo queda acoplar a la placa superior el motor, M_2, y el final de carrera FC_2 para que junto a las abrazaderas se unan estas últimas piezas a las barras guía y el eje del motor se una a la barra roscada mediante un acoplamiento, como se puede observar en la Fig. 5.17. Se puede observar la apariencia del brazo robótico mecánicamente terminado en la Fig. 5.18.

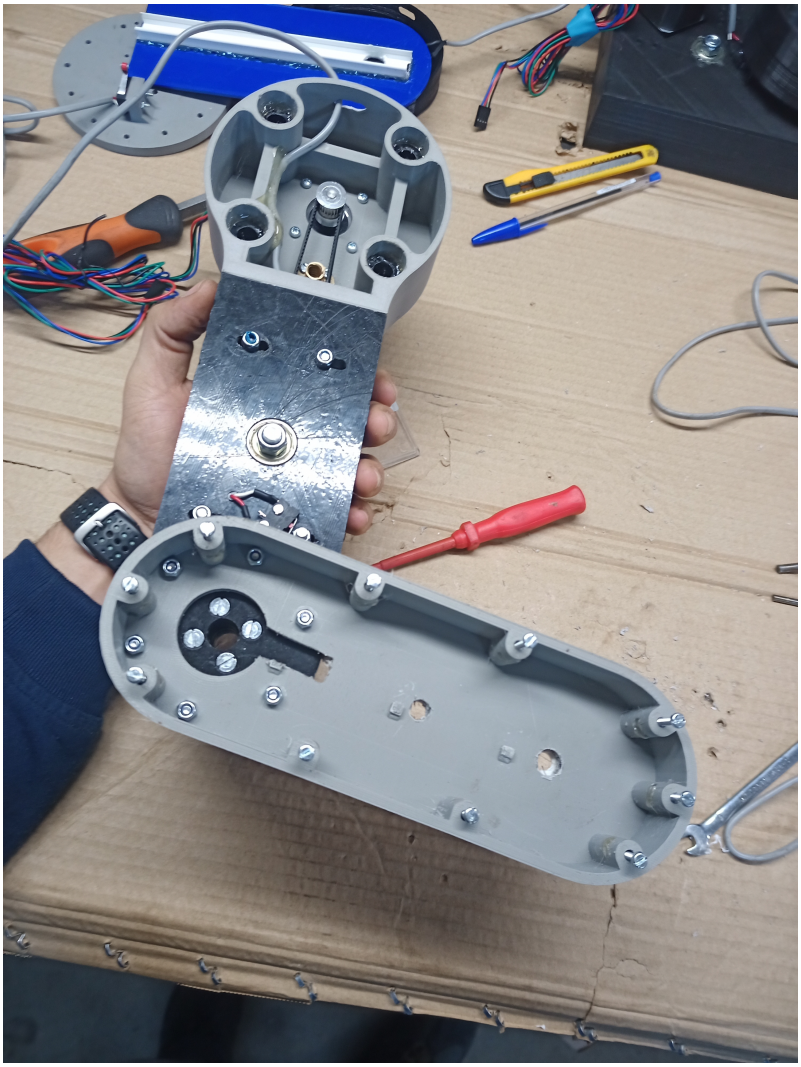
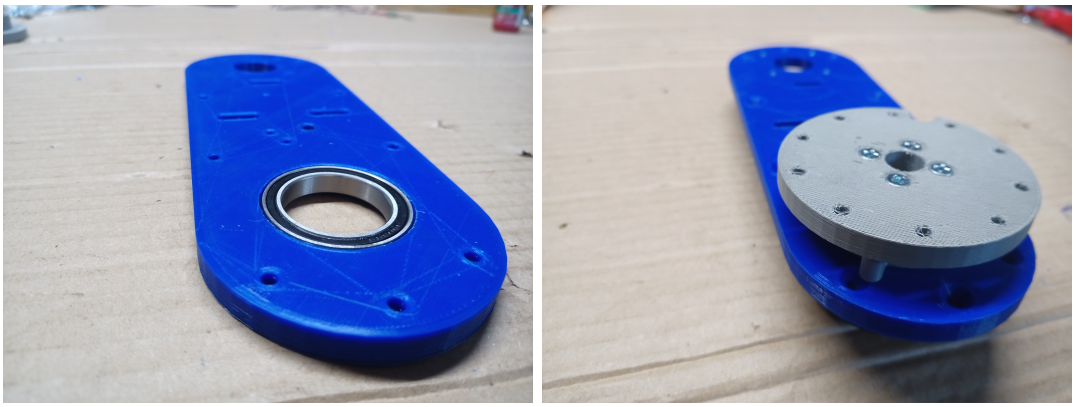


Figura 5.9: Montaje - Paso 10

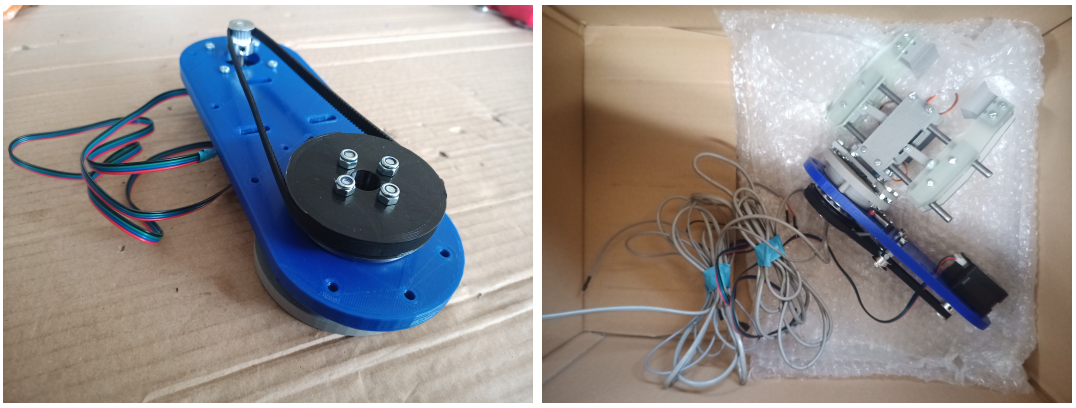


(a) (b)

Figura 5.10: Montaje - Paso 11

(a) (b)

Figura 5.11: Montaje - Paso 12



(a) (b)

Figura 5.12: Montaje - Paso 13

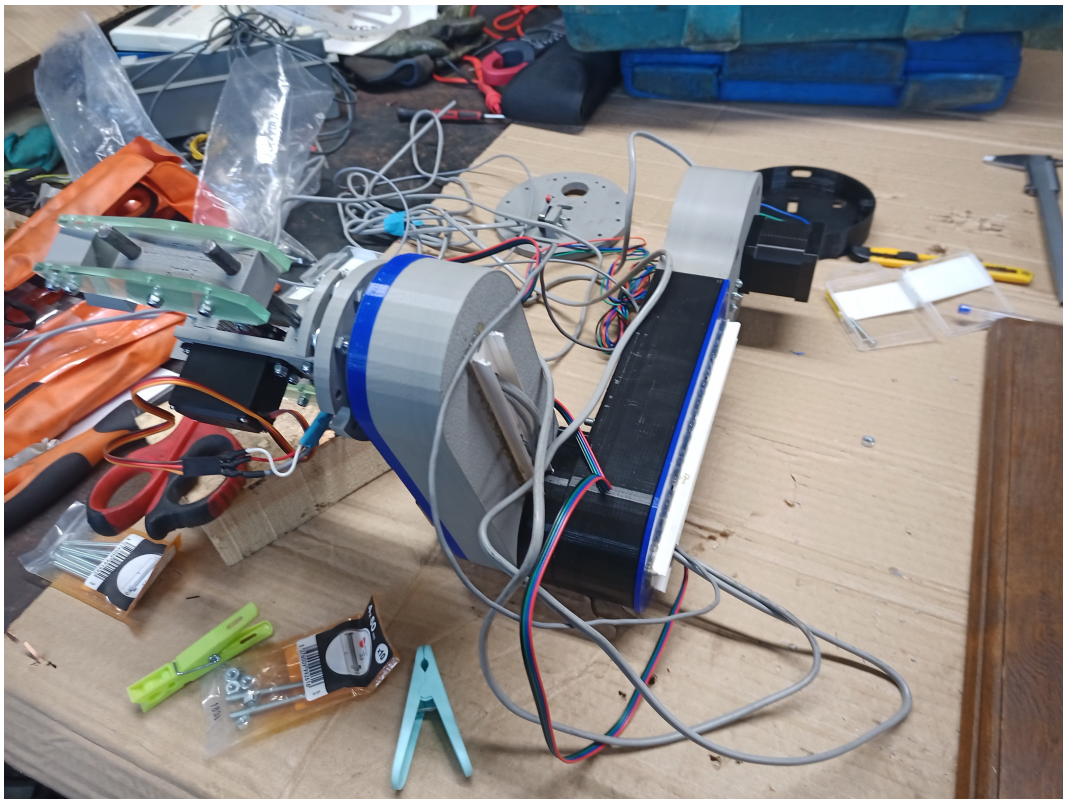


Figura 5.13: Montaje - Paso 14

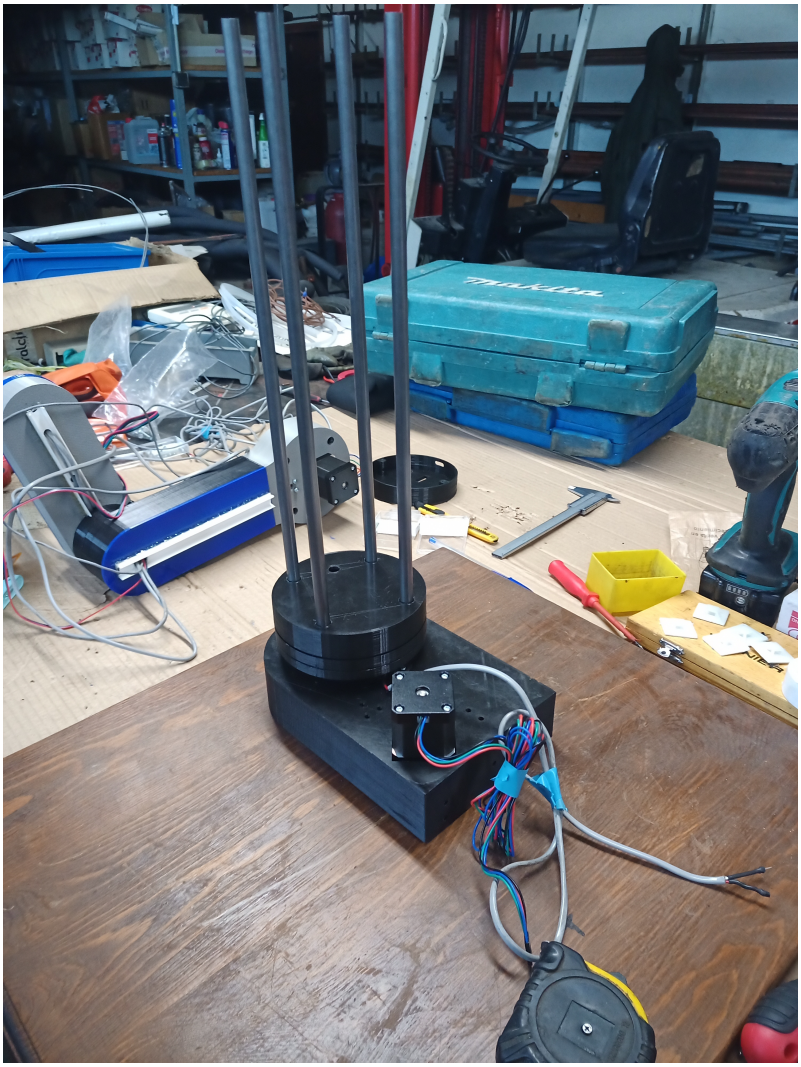


Figura 5.14: Montaje - Paso 15

Figura 5.15: Montaje - Paso 16

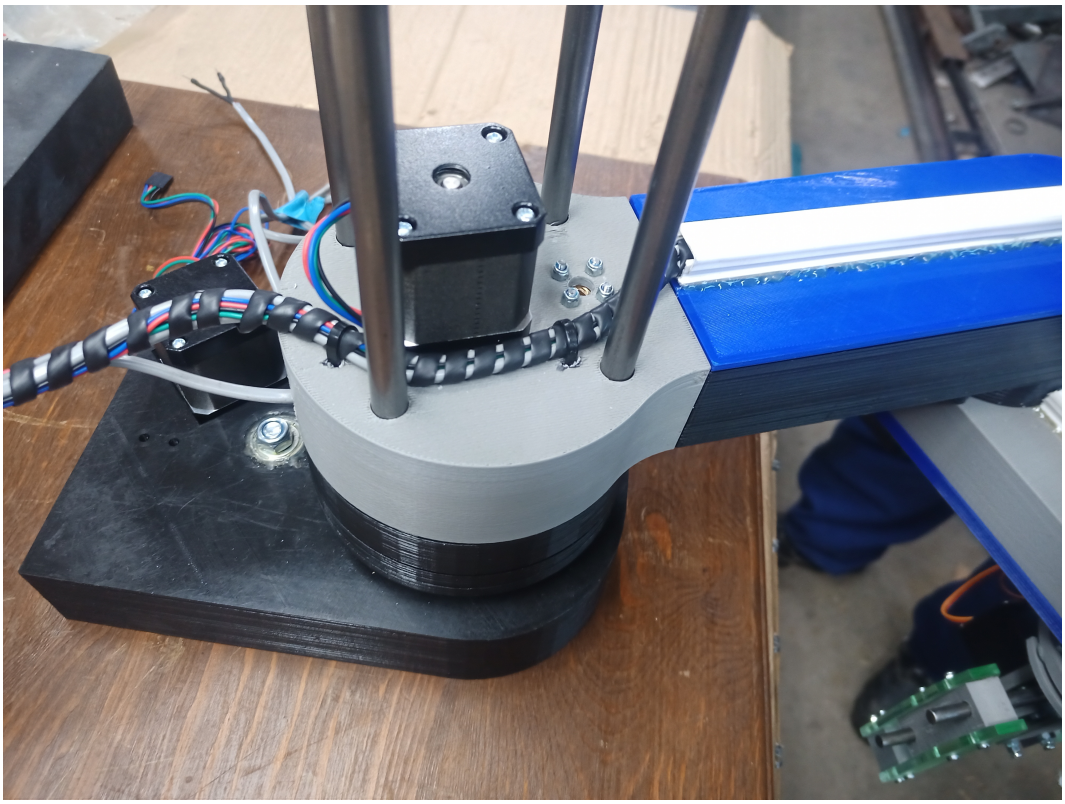
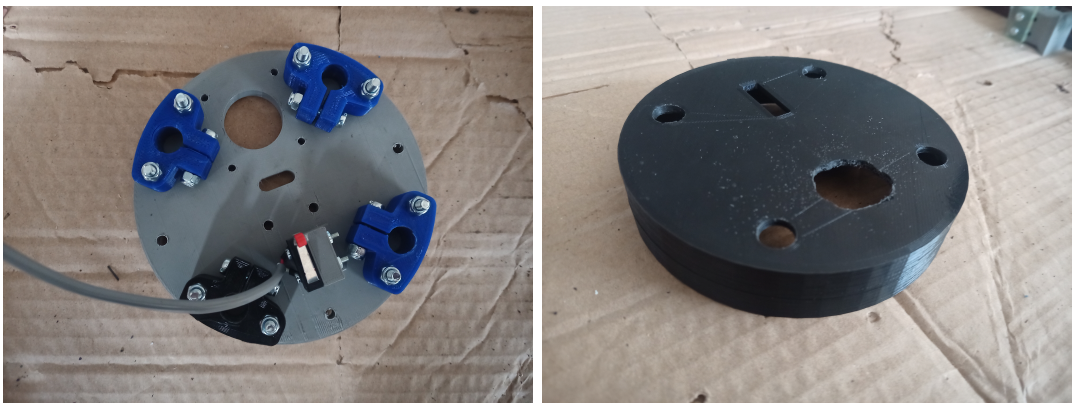


Figura 5.16: Montaje - Paso 17



(a) (b)

Figura 5.17: Montaje - Paso 18

Figura 5.18: Montaje - Paso 19

5.2. Montaje eléctrico/electrónico

En la siguiente sección se va a detallar el montaje eléctrico/electrónico que se ha necesitado realizar para el buen funcionamiento del brazo robótico. En primer lugar, se va a utilizar un Arduino UNO como microcontrolador del robot, véase Fig. 5.19. El microcontrolador Arduino UNO es alimentado a 5V mediante un cable USB al ordenador, además, se puede observar la hoja de especificaciones del componente en el Anexo D Sección D.1.

Figura 5.19: Arduino UNO

Como se van a utilizar motores paso a paso NEMA 17 (17HD48002-22B y 17HS2408) es necesario utilizar un adaptador para este microcontrolador que facilite el control de dichos motores. Por ello, se ha decidido utilizar una CNC Shield v3, véase Fig. 5.20, que se acopla en la parte superior del microcontrolador. Este componente tiene que ser alimentado externamente por una fuente de corriente continua de entre 12-36V, por la necesidad del brazo robótico, se utiliza una fuente de corriente continua de 12V 6A. El amperaje de la fuente a utilizar es importante debido a que debe alimentar a todos los motores y tiene que tener la capacidad de suministrar la corriente a cada uno de los motores. Para la selección de dicha fuente se ha recurrido a las especificaciones de los motores que se puede observar en el Anexo D Sección D.3 y D.4. En el Anexo D Sección D.2 se puede observar la relación de los pines de la CNC Shield v3 con las entradas tanto analógicas como digitales del microcontrolador Arduino UNO.

Por otro lado, es necesario acoplar sobre la CNC Shield v3 unos drivers que regulen el amperaje que se debe suministrar a los motores, más concretamente, se va a utilizar los DRV8825, véase Fig. 5.21. Estos drivers se utilizar para la etapa de arranque de los motores en la que es necesario suministrar unas corrientes superiores a las nominales y que podrían dañar los motores. Estos drivers lo que hacen es chopear la alimentación de los motores en el arranque para evitar el daño en los motores. Estos controladores deben ser regulados para los motores en concreto que se van a utilizar. Para ello, se debe alimentar la placa sin estar los motores conexionados y se debe medir la tensión en el driver. Esta tensión debe cumplir la ecuación (5.1), para esto, los drivers tienen un potenciómetro incorporado que permite el

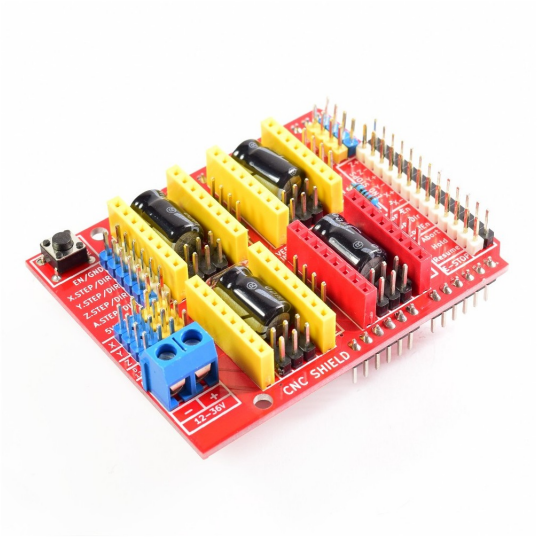


Figura 5.20: CNC Shield v3

control de esta tensión.

$$V_{ref} = I_{ref} \cdot 0{,}5 \tag{5.1}$$

siendo I_{ref} la corriente máxima admisible de los motores y V_{ref} la tensión que se debe medir en el driver.

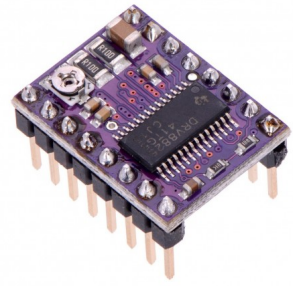
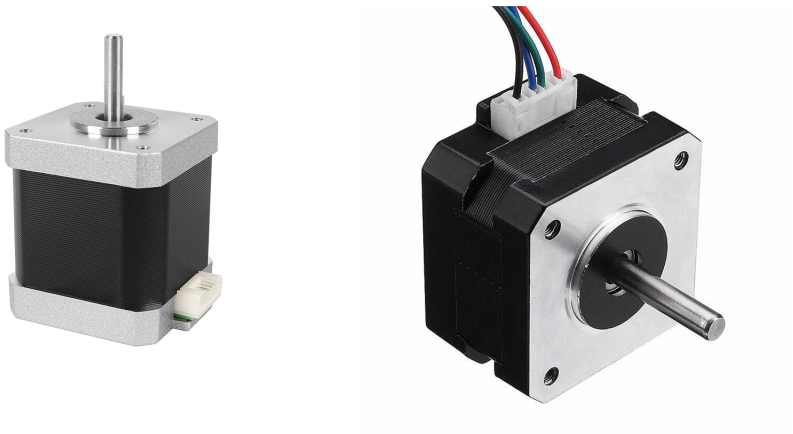


Figura 5.21: DRV8825

Por último, por seguridad se ha añadido una seta de emergencia cuyos contactos sean normalmente cerrados para cortar la alimentación de los motores sin que se corte la alimentación del microcontrolador. Esta acción puede resultar de especial interés para poder obtener información el microcontrolador cuando la seta de emergencia es pulsada.



(a) 17HD48002-22B (b) 17HS2408

Figura 5.22: Motores Nema 17

Todo el esquema de montaje se puede observar en la Fig. 5.23. En el siguiente diagrama se puede apreciar fácilmente la relación entre los distintos componentes.

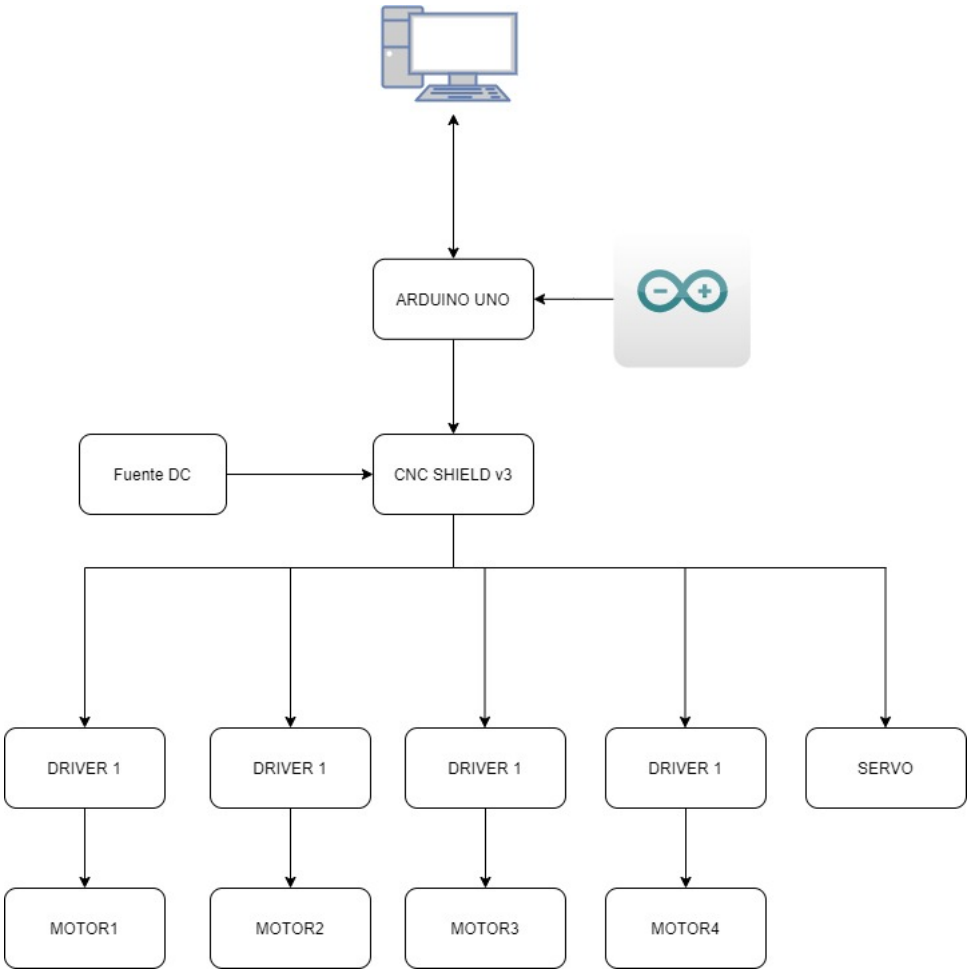


Figura 5.23: Diagrama de conexiones

5.3. Pruebas de movimiento

En la siguiente sección se van a realizar pruebas de movimiento del brazo robótico para ver si todo se comporta acorde a los cálculos y simulaciones realizadas. Para poder realizar dichas pruebas es necesario realizar tres comprobaciones en orden:

1. Comprobar que la fuente de continua esta conectada a la red y que el interruptor se encuentra abierto.

2. Comprobar que el microcontrolador se encuentra alimentado por el cable USB al PC de control.

3. Iniciar la interfaz de usuario en el PC de control y comprobar que se inicia de forma correcta.

4. Cerrar el interruptor.

Estos pasos son muy importantes ya que la interfaz debe encenderse siempre que el microcontrolador este alimentado ya que sino pueden surgir problemas de conexión serie entre el PC y el microcontrolador.

En las simulaciones experimentales que se van a realizar se parte de la posición de HOME, es decir, de una posición conocida. Esta posición viene definida por los finales de carrera, sensores mecánicos, del robot y se puede observar en la Fig. 5.24.

(a) (b)

Figura 5.24: Simulación Experimental 0 - Robot SCARA

5.4. Primer movimiento

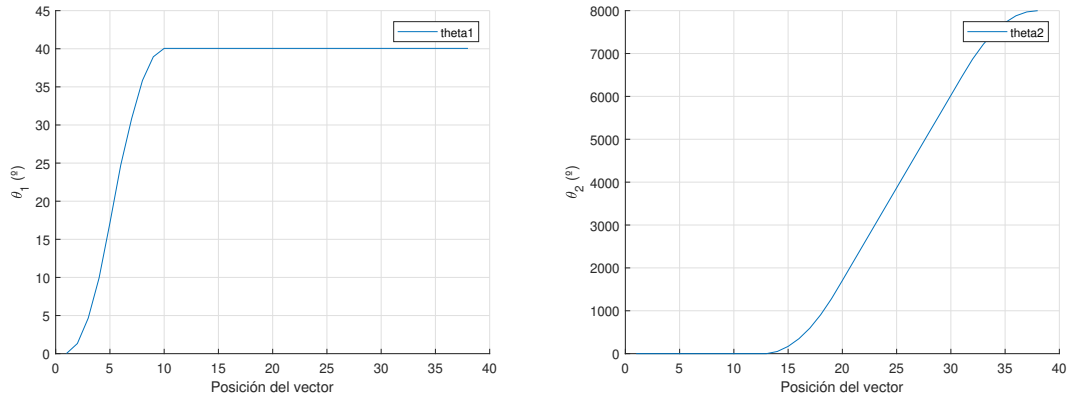
Se comienza realizando un primer movimiento para comprobar el correcto funcionamiento tanto de la interfaz, la comunicación entre la interfaz y el microcontrolador y, el programa del propio microcontrolador. En la Fig. 5.25 se pueden observar dos imágenes de la posición final que alcanza el robot.

En las Fig. 5.26 se puede ver la evolución de las coordenadas articulares a lo largo del movimiento del robot. Se puede apreciar como evolucionan siguiendo un perfil trapezoidal que se consigue utilizando una librería particular para los motores paso a paso (vista en el Anexo A). en la Fig. 5.27 se pueden observar los valores de las coordenadas xyz del robot evolucionando a lo largo del movimiento.

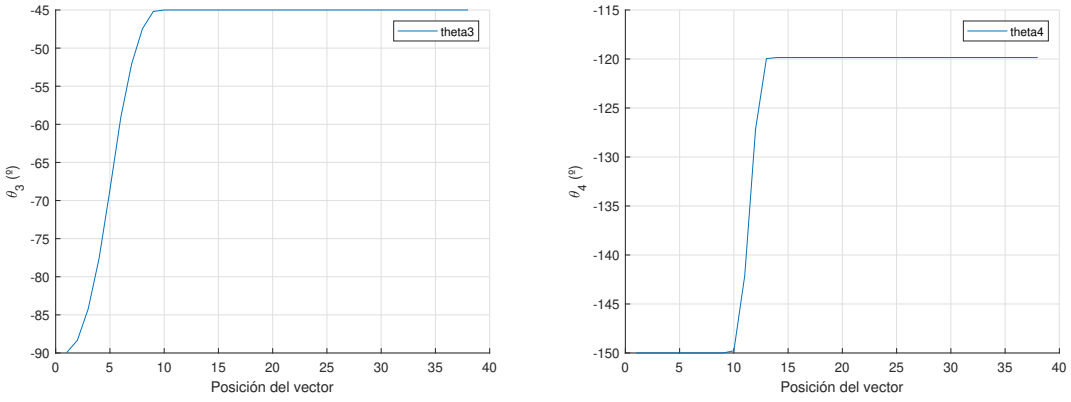
En una primera impresión de los resultados de este primer movimiento se puede apreciar que el robot se comporta de acuerdo a todo lo especificado y los resultados son favorables a lo desarrollado.

(a) (b)

Figura 5.25: Simulación Experimental 1 - Robot SCARA



(a) (b)



(c) (d)

Figura 5.26: Simulación Experimental 1 - Coordenadas articulares

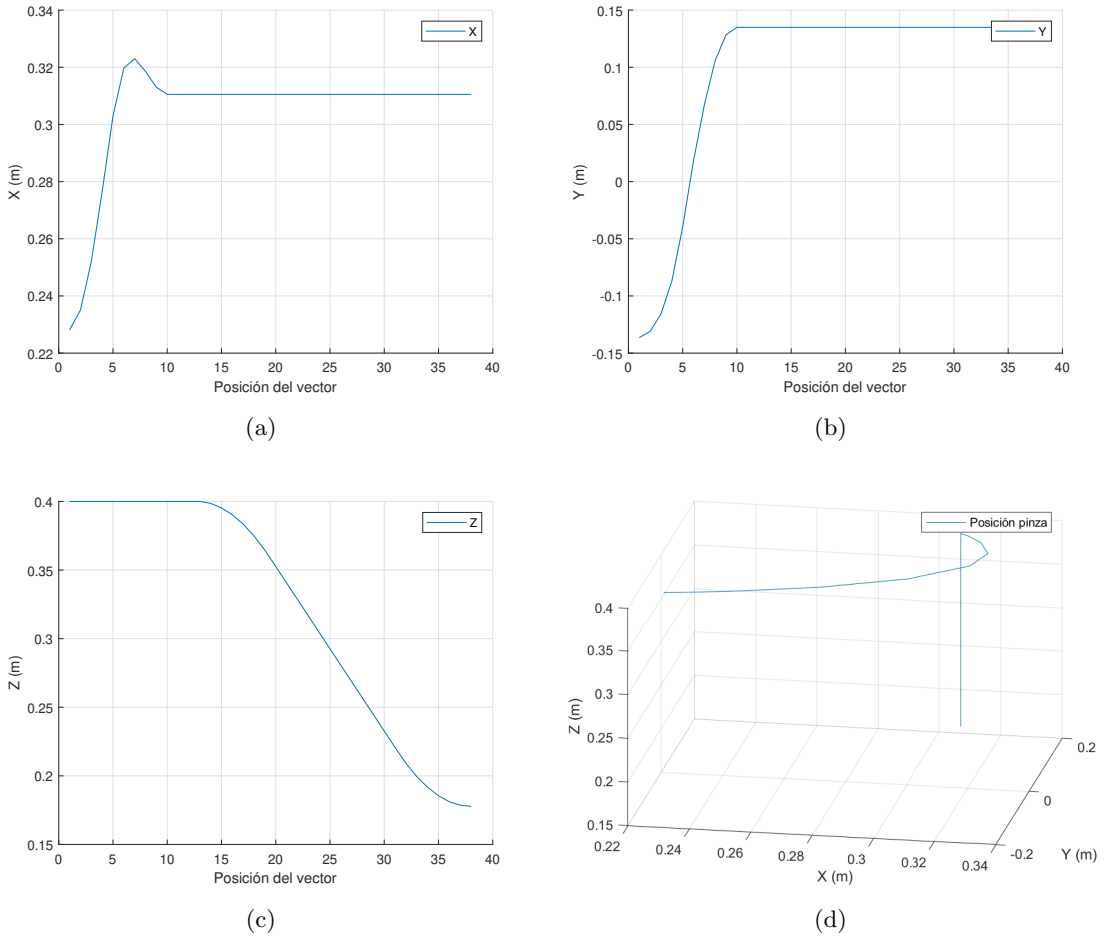
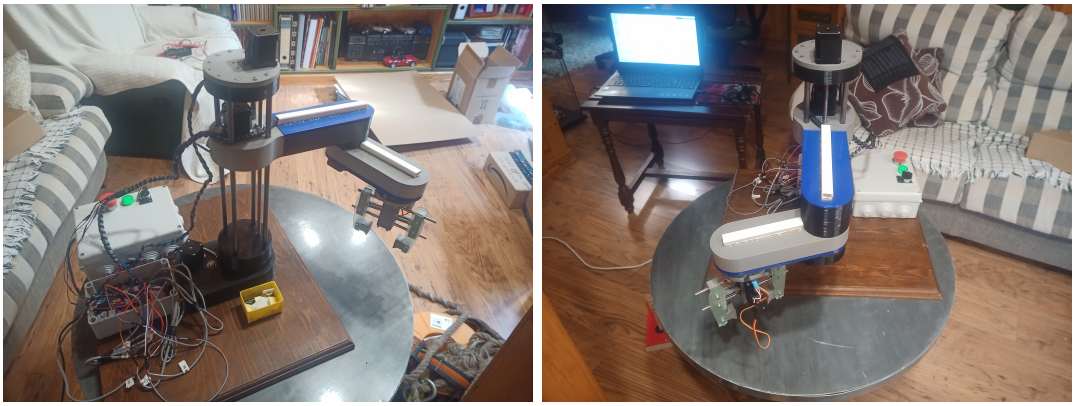


Figura 5.27: Simulación Experimental 1 - Posiciones

5.5. Segundo movimiento

En esta segunda prueba de movimiento se sigue realizando un único desplazamiento pero se modifican los valores de aceleración y velocidad de los motores. El movimiento parte de la posición de *home* (véase Fig. 5.24) y se desplaza hasta alcanzar la posición vista en la Fig. 5.28.



(a) (b)

Figura 5.28: Simulación Experimental 2 - Robot SCARA

En la Fig. 5.29 se puede observar la evolución de las coordenadas articulares a lo largo del movimiento. Cabe destacar que la variación de velocidad y aceleración no se aprecia significativamente y esto es debido a que no se han modificado los valores de las velocidades y aceleraciones en un rango de unidades alto.

En la Fig. 5.30 se puede observar la evolución de las coordenadas xyz. Se puede apreciar una evolución suave hasta conseguir los valores especificados por el usuario.

Con estas dos pruebas realizadas correctamente y habiendo obtenido los resultados esperados, se procede a realizar una simulaciones más complejas.

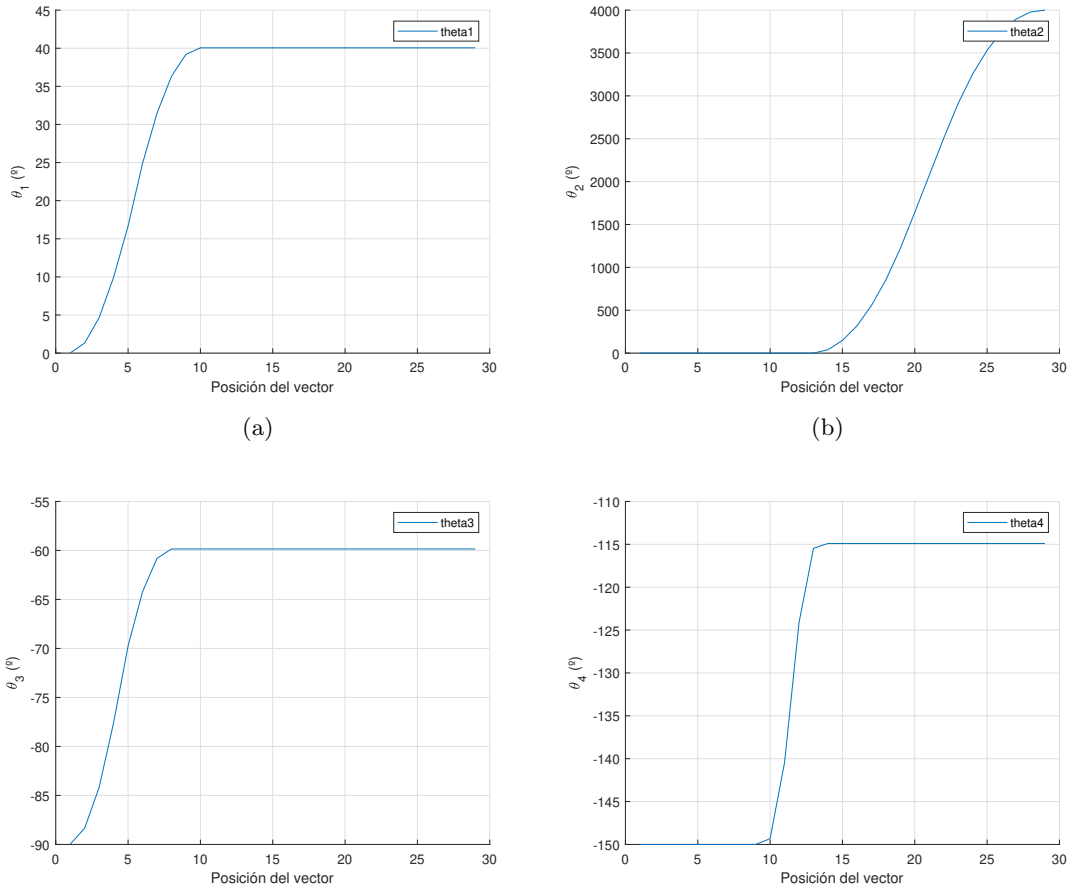


Figura 5.29: Simulación Experimental 1 - Coordenadas articulares

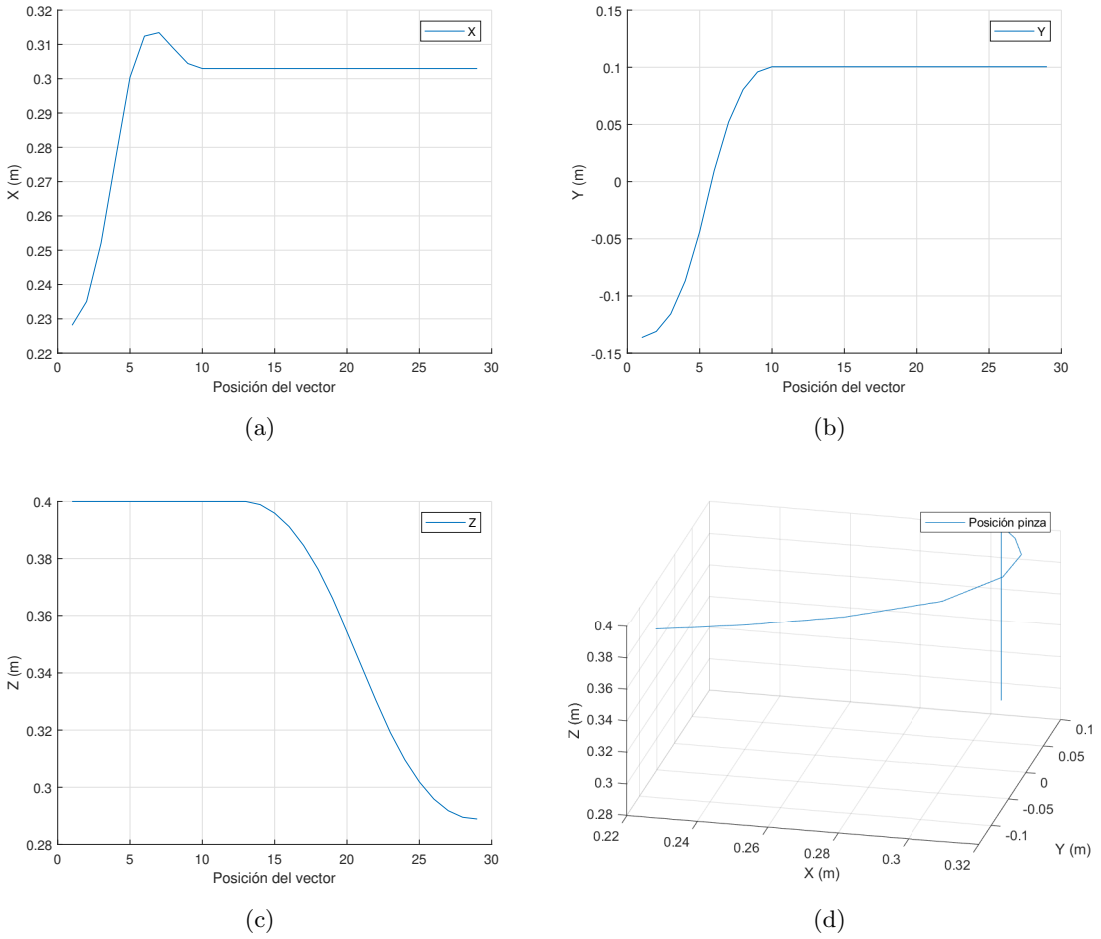


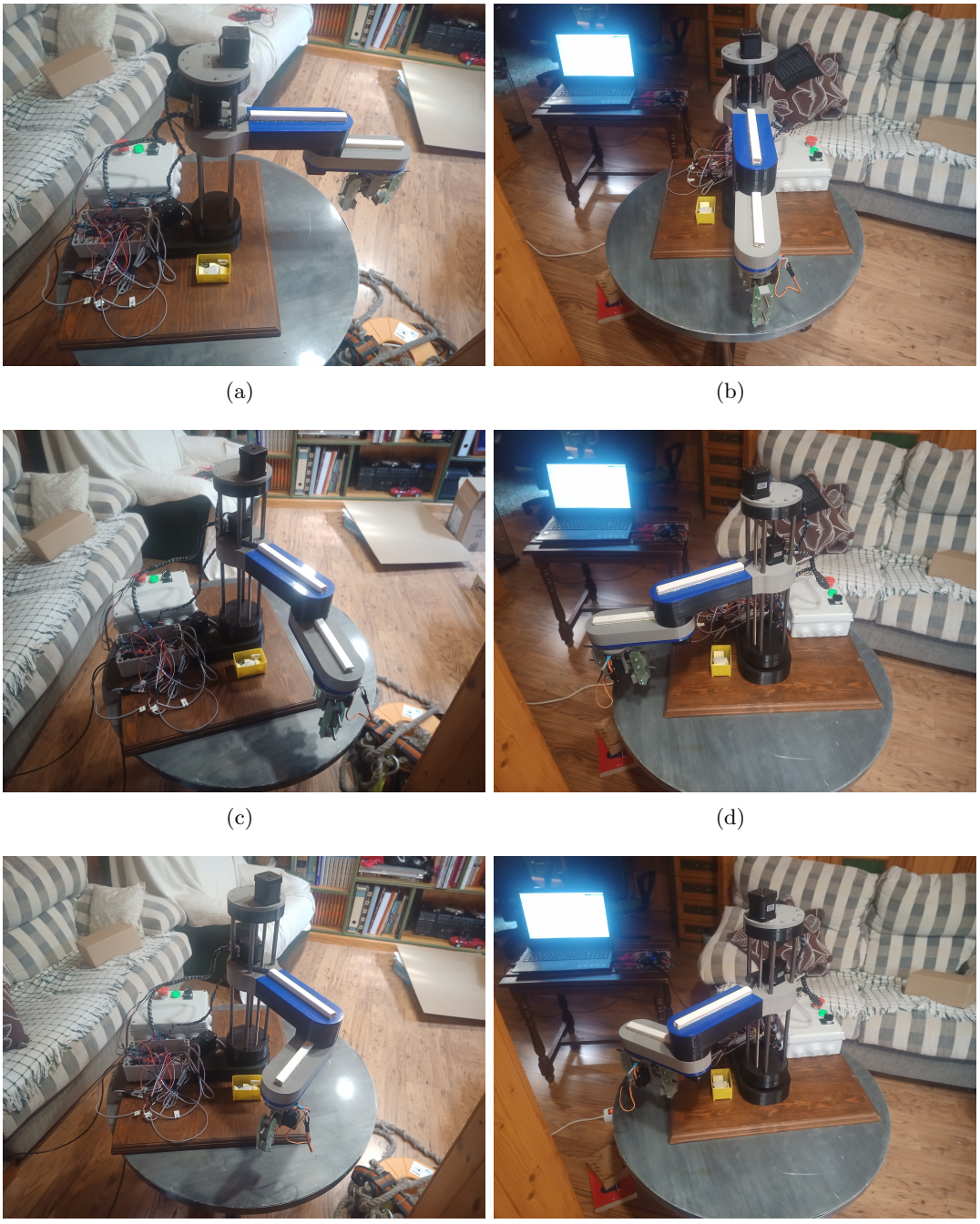
Figura 5.30: Simulación Experimental 2 - Posiciones

5.6. Tercer movimiento

En la siguiente sección se procede a realizar una simulación de varios movimientos encadenados. Este tipo de simulación nos va a servir para ver el funcionamiento del brazo robótico ante la solicitación de movimientos consecutivos por parte del usuario. En la Fig. 5.31 se pueden observar las tres posiciones distintas que alcanza el brazo durante la prueba: la primera posición (Figs. 5.31 (a) y (b)), la segunda posición (Figs. 5.31 (c) y (d)) y la tercera posición (Figs. 5.31 (e) y (f)).

En la Fig. 5.32 se puede observar la evolución de las coordenadas articulares a lo largo del movimiento. Tras analizar estas gráficas se puede deducir que todos los movimientos se realizan de forma suave, aparentemente siguiendo trayectorias trapezoidales.

En la Fig. 5.33 se puede observar las posiciones de las coordenadas xyz de la consecución de movimientos. Más concretamente, en la Fig. 5.33 (d) se puede observar el movimiento de la herramienta final del robot y se puede observar un movimiento que evoluciona de forma moderada sin cambios bruscos tanto de velocidades como de aceleraciones.



(a) (b)

(c) (d)

(e) (f)

Figura 5.31: Simulación Experimental 3 - Robot SCARA

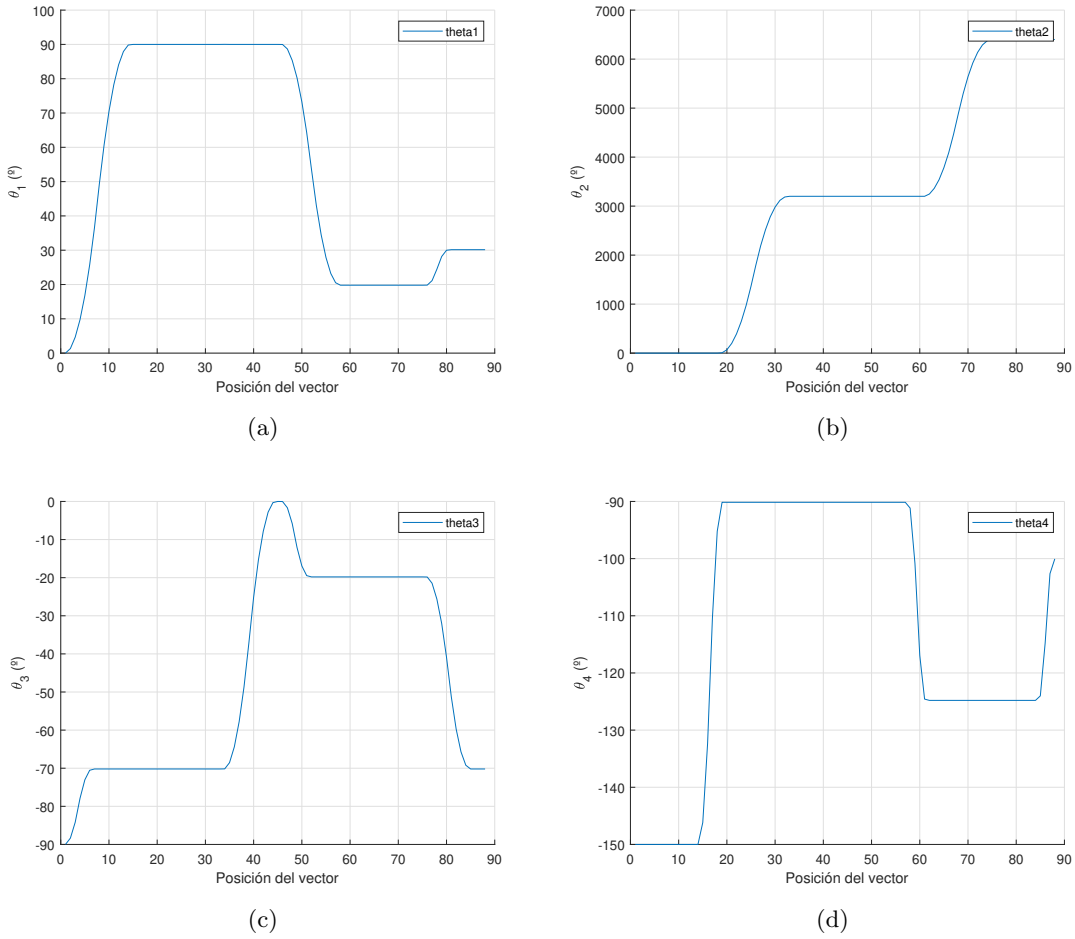
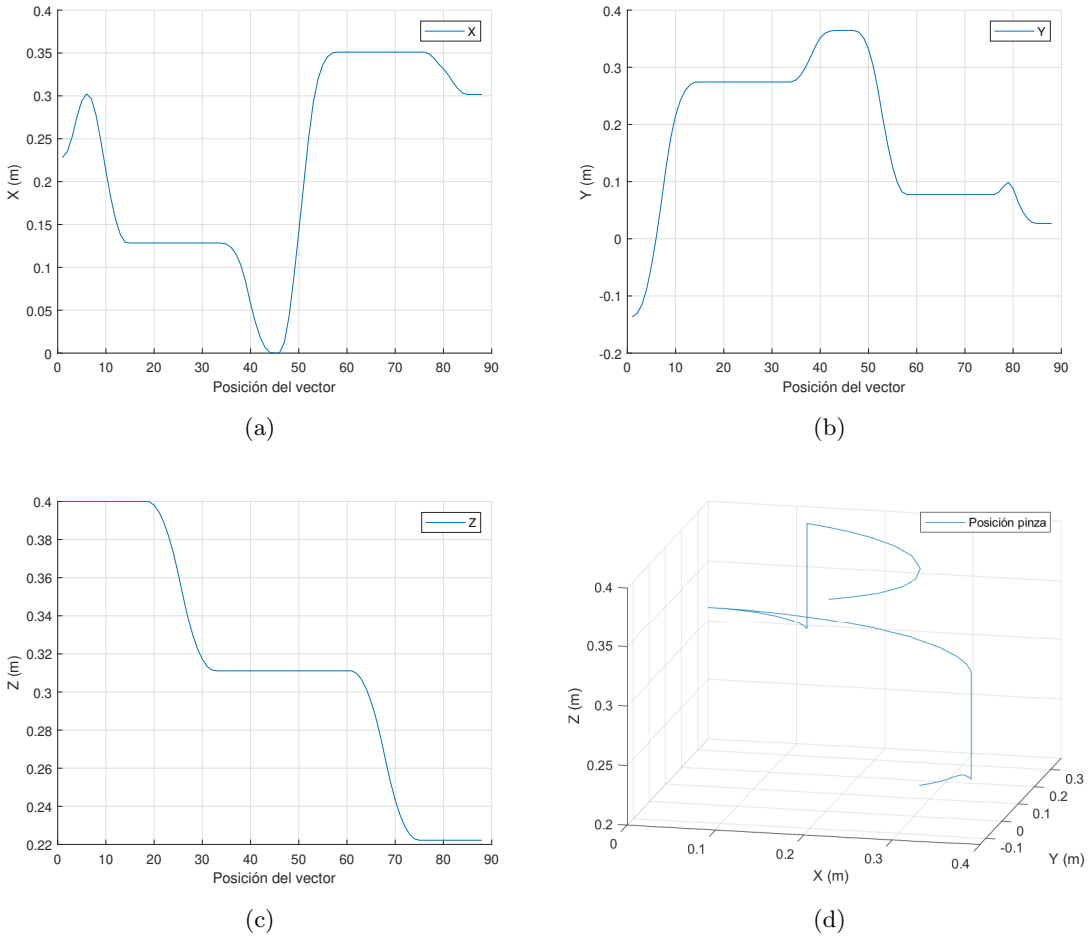


Figura 5.32: Simulación Experimental 3 - Coordenadas articulares



(a) (b)

(c) (d)

Figura 5.33: Simulación Experimental 3 - Posiciones

Capítulo 6

Conclusiones y Propuestas de mejora

En este capítulo se mostrarán las conclusiones a las que se ha llegado tras el desarrollo del proyecto y se plantearán las posibles propuestas de mejora a las que se ha llegado tras la realización del proyecto.

6.1. Conclusiones

A continuación, se exponen las conclusiones a las que se ha llegado tras la realización del presente TFM:

- Se ha podido realizar un estudio del diseño del prototipo de robot SCARA usado para este proyecto gracias a los conocimientos mecánicos obtenidos durante la realización del máster.
- Se ha conseguido resolver mediante métodos geométricos básicos los problemas cinemáticos tanto directo como inverso. En este punto, ya que se trata de un tipo de robot muy conocido, no ha sido ningún problema poder comprender la resolución de estos problemas.
- Se ha conseguido crear un modelo de simulación que sirve como paso previo para validar los cálculos realizados y para comprobar que el diseño mecánico funciona correctamente. Como ya se explico en la memoria, el modelo simula fielmente el comportamiento que tendría el robot para su futura implementación real.
- Se ha conseguido implementar satisfactoriamente el robot físicamente. Además, se ha creado una interfaz de usuario para establecer y facilitar la comunicación del usuario con el microcontrolador. Además, se ha podido comprobar experimentalmente que tanto el robot como la interfaz funcionan correctamente y corroboran las simulaciones realizadas con el modelo creado.

Como conclusión se tiene que se han conseguido alcanzar todos los objetivos planteados al inicio del trabajo y, que para realizar un proyecto como éste es necesario poseer conocimientos de electricidad, de mecánica y de electrónica. El aunar estos conocimientos en un proyecto es lo que considero que tiene como objetivo el Máster de Ingeniería Mecatrónica.

6.2. Propuestas de mejora

En la siguiente sección se detallan las posibles mejoras que se pueden añadir al proyecto para el perfeccionamiento de este. Las propuestas que se exponen a continuación ha surgido tras la realización del presente proyecto.

- La herramienta final, *end-effector*, que se ha utilizado en el proyecto ha sido una pinza. Este tipo de herramientas requieren de sensores de presión o una precisión en el posicionamiento muy elevada para utilizarse en robots de pick and place ya que se podría dañar los objetos a coger durante su trabajo. Como propuesta a este problema se ha planteado el cambio a una herramienta que evite este tipo de problemas sin necesidad de requerir de una alta precisión y sin elevar el coste del proyecto como es una ventosa de succión. Este cambio de herramienta no supondría un gran problema ya que solo es necesario principalmente llevar un tubo neumático hasta la posición del extremo del robot.

- Un factor importante en los robots picks and place es el conocimiento de la posición de los objetos a mover mediante sensores. La técnica más utiliza para este tipo de aplicaciones es la visión por computador. Por tanto, como segunda propuesta de mejora se propone la incorporación de una cámara para detectar la posición de los objetos cuando éstos entren en el espacio de trabajo predefinido del robot. Esta mejora es esencial para que el brazo robótico pueda trabajar de forma autónoma y, además, se adapte a la variación de orientación de los objetos por causas ajenas al brazo como puede ser el choque con otro objeto antes de llegar a una posición específica del espacio de trabajo.

- La tercera propuesta que se plantea es la incorporación de encorders en los motores del robot. Esta propuesta supondría tener un conocimiento exacto de la posición del robot durante su movimiento, pudiendo realizar un control en lazo cerrado. Al existir sensores de contacto (finales de carrera) para alcanzar una posición conocida del robot, no es necesario que los encoders sean absolutos. Por tanto, se pueden utilizar encoders incrementales que generalmente son más baratos que los absolutos.

- Aunque el *Ácido poliláctico* (PLA) es un material utilizado en la impresión en 3D para la elaboración de numerosas piezas y proyectos. Se ha podido observar que no deja de ser un polímero y ante la presencia de ciertos valores de inercias o fuerzas externas este material presenta cierta elasticidad que puede provocar fallos. Más concretamente, durante la realización del proyecto se ha detectado que el acoplamiento J_1 adquiere cierta inclinación respecto a la horizontal debido a la inercia ejercida por el conjunto del brazo robótico.

Capítulo 7

Presupuesto

En este capítulo se detalla el presupuesto del proyecto.

7.1. Coste material

En el siguiente capítulo se detalla el presupuesto de los materiales para llevar a cabo el proyecto. Para una mayor facilidad de identificación de los componentes se ha dividido el presupuesto en presupuestos de los conjuntos de piezas que forman el robot del proyecto. Se puede observar el presupuesto del conjunto base en la Tabla 7.1, del conjunto elevación en la Tabla 7.2, del conjunto brazo 1 en la Tabla 7.3, del conjunto brazo 2 en la Tabla 7.4, del conjunto pinza en la Tabla 7.5 y del conjunto electrónica en la Tabla 7.6.

Tabla 7.1: Presupuesto - Base

Conjunto	Material	Unidades	Precio (€)
Base	Correa GT2 de 200 mm	1	1.65
	Correa GT2 de 300 mm	1	1.65
	Rodamiento axial - 40x60x13 mm	2	7.8
	Rodamiento radial - 35x47x7 mm	2	10
	Rodamiento radial 608 - 8x22x7 mm	2	1.125
	Tornillo M8 45 mm	1	0.10
	Arandela M8	1	0.02
	Tuerca M8 autoblocante	1	0.04
	Tornillo M4 55 mm	4	0.20
	Tuerca M4 autoblocante	4	0.06
	Motor Nema 17HD48002H-22B	1	5
	Polea de 5 mm de 20 dientes	1	1.21
	Final de carrera	1	0.7
Total conjunto base			29.528 €

Tabla 7.2: Presupuesto - Elevación

Conjunto	Material	Unidades	Precio (€)
Elevación	Rodamiento radial 608 - 8x22x7 mm	1	0.5625
	Barra calibrada D10 mm L400 mm	4	12
	Barra roscada D8 mm L380 mm	1	10
	Acoplamiento de 5mm a 8mm	1	5
	Rodamiento radial 10x19x29 mm	4	8
	Final de carrera	1	0.7
	Motor Nema 17HD48002H-22B	1	5
	Polea de 5 mm de 20 dientes	1	1.21
Total conjunto elevación			42.4725 €

Cabe señalar que para la realización del presupuesto no se ha tenido en cuenta el precio de fabricar todas las piezas impresas y cortadas por láser ya que se ha utilizado el servicio de impresión de la UPV. Tampoco se ha tenido en cuenta el coste de las licencias por la utilización del software matemático Matlab y el software de diseño SolidWorks sebido a que

Tabla 7.3: Presupuesto - Brazo 1

Conjunto	Material	Unidades	Precio (€)
Brazo 1	Correa GT2 de 300 mm	1	1.65
	Correa GT2 de 400 mm	1	1.65
	Rodamiento axial - 35x52x12 mm	2	8
	Rodamiento radial - 30x42x7 mm	1	3.5
	Rodamiento radial 608 - 8x22x7 mm	2	1.125
	Tornillo M8 45 mm	1	0.10
	Arandela M8	1	0.02
	Tuerca M8 autoblocante	1	0.04
	Tornillo M4 50 mm	4	0.20
	Tuerca M4 autoblocante	4	0.06
	Polea de 5 mm	4	4.88
	Polea de 5 mm de 20 dientes	1	1.21
	Motor Nema 17HD48002H-22B	1	5
	Final de carrera	1	0.7
Total conjunto brazo 1			28.135 €

se han utilizado con la licencia de estudiante de la UPV. El presupuesto total de los materiales del proyecto se puede observar en la Tabla 7.7.

Tabla 7.4: Presupuesto - Brazo 2

Conjunto	Material	Unidades	Precio (€)
Brazo 2	Correa GT2 de 400 mm	1	1.65
	Rodamiento axial - 35x52x12 mm	2	8
	Rodamiento radial - 30x42x7 mm	1	3.5
	Tornillo M4 55 mm	4	0.20
	Tuerca M4 autoblocante	4	0.06
	Polea de 5 mm	2	2.44
	Polea de 5 mm de 20 dientes	1	1.21
	Motor Nema 17Hs2408S 17	1	12.49
	Final de carrera	1	0.7
Total conjunto brazo 2			30.25 €

Tabla 7.5: Presupuesto - Pinza

Conjunto	Material	Unidades	Precio (€)
Pinza	Servomotor MG996R	1	5
	Tornillería	4	3.26
Total pinza			8.26 €

Tabla 7.6: Presupuesto - Electrónica

Conjunto	Material	Unidades	Precio (€)
Electrónica	Fuente DC/DC de 230V a 12V 6A	1	28
	Arduino Uno	1	3.37
	CNC escudo V3	4	3.21
	Driver Motor Nema 17 DRV8825	4	3.26
	Cableado		5
	Interruptor 230V 2p	1	10.72
	Seta de emergencia NC	1	17.36
	Led 230V verde	1	6.32
	Cuadro 150x250x50mm	1	4.65
	Cuadro 50x100x50mm	1	1.21
Total electrónica			83.1 €

Tabla 7.7: Presupuesto - Materiales

Conjunto	Precio (€)
Base	29.528 €
Elevación	42.4725 €
Brazo 1	28.135 €
Brazo 2	30.25 €
Pinza	8.26 €
Electrónica	83.1 €
Total	221.7455 €

7.2. Coste personal

El la siguiente sección se especifica el sueldo percibido por el autor del proyecto en función de las labores realizadas y horas dedicadas a cada labor.

A continuación, se puede observar en la Tabla 7.8 los precios a cobrar por parte del autor del presente proyecto en relación a las horas dedicadas y en función al tipo de tarea que se han dedicado. Cabe destacar que las horas señaladas en dicha tabla no son la cantidad de horas reales que el autor ha dedicado a la realización del proyecto pero si se pusieran la cantidad de horas reales para mantener un precio asequible sería necesario establecer un precio por hora realmente bajo.

Tabla 7.8: Coste personal

Dedicación	Precio por hora (€/h)	Horas de trabajo (h)	
Creación de modelo	20	40	800 €
Implementación real	15	70	1050 €
Documentación	10	40	400 €

7.3. Coste total

Por último, en la Tabla 7.9 se puede observar el presupuesto final del presento proyecto con el precio total de los materiales utilizados y con la retribución que el autor del proyecto debe recibir con razón de sus labores realizadas.

Tabla 7.9: Presupuesto - Final

Concepto	Precio (€)
Materiales	221.75
Coste personal	2250
Total	2471.75

Capítulo 8

Pliego de condiciones

En este capítulo se exponen las condiciones mínimas necesarias para llevar a cabo este proyecto.

8.1. Introducción

El objeto de este documento es establecer las condiciones técnicas mínimas que debe cumplir el control e implementación de un robot SCARA para de tareas de pick and place. Al tratarse de un proyecto académico, este apartado no será tan extenso como sería en un proyecto de cierta índole ni incluirá especificaciones respecto a las diferentes condiciones necesarias para su puesta en funcionamiento, por ejemplo, una línea de producción con almacenamiento automatizado. Por ello se centra principalmente en las fases de control, programación, montaje y pruebas de funcionamiento real.

8.2. Condiciones generales

Es obligatorio el cumplimiento de todas las normas y requerimientos que con carácter general se observan durante proyectos de la misma índole. En concreto, con el fin de velar por la seguridad del proyectista, durante el proceso de control e implementación; es necesario especificar el obligado cumplimiento de lo establecido en el Reglamento Electrotécnico de Baja Tensión por parte de todos los componentes de carácter electrónico o eléctrico. Es este el campo de referencia sobre el cual se aplica la normativa por el hecho de trabajar siempre con un suministro eléctrico inferior a 1000 V en CA e inferior a 1500 V en CC.

8.3. Condiciones facultativas

Por la presente, se recogen una serie de derechos y obligaciones por parte del autor de este trabajo que encarna la figura del contratista dentro del marco de este proyecto.

- Conocer la normativa a aplicar.
- Conocer el proyecto en todos sus conjuntos.
- Obligación de disponer un documento donde se reflejen las indicaciones, aclaraciones o modificaciones del proyecto.
- Disposición de cuantos medios auxiliares fuese necesario para garantizar el correcto desarrollo del proyecto (en caso de que no fuera posible acceder al material proporcionado por la universidad, el contratista adquirirá lo que precise siendo incluido en el coste de material).
- Derecho a percibir compensación económica por los trabajos realizados no especificados en los documentos del proyecto y necesarios para la correcta ejecución.

En lo que se refiere a la dirección por parte del tutor de la universidad sobre el autor del proyecto, se pueden destacar las siguientes obligaciones y facultades:

- Supervisión de aquellos aspectos del proyecto que puedan afectar a la fiabilidad, calidad y seguridad durante la ejecución del proyecto.
- Estar presente en los momentos del desarrollo del proyecto que se necesite.
- Estar involucrado en la aportación de soluciones técnicas a problemas no previstos durante la realización del proyecto..

8.4. Condiciones técnicas

Con respecto al uso de todos los aparatos electrónicos y eléctricos subscritos al REBT mencionado anteriormente, se debe complementar con la información técnica especificada por cada fabricante en cuestión. Se debe seguir en todo momento las recomendaciones de uso especificadas en las hojas técnicas o datasheets de los componentes. Todas las piezas cuya fabricación se basa en la impresión 3D, se deben seguir las normas establecidas para la aplicación de dichas tecnologías y su impresión se debe realizar utilizando material apropiado y siempre bajo la supervisión de personal cualificado. Los requisitos de hardware son:

- Sistema operativo Windows 10 o superior
- Matlab 2020b o superior
- SOLIDWORKS 2020 o superior
- Arduino IDE

Para poder realizar todas las tareas referente al control del hardware en las diferentes fases del proyecto de manera adecuada es recomendable tener el equipo con las especificaciones descritas anteriormente. A pesar de esto, se pueden utilizar ordenadores con características inferiores aunque los procesos no sean del todo óptimos.

- Procesador Intel Core i7 o superior.
- Memoria RAM 8GB o superior.
- Almacenamiento libre mínimo 40 GB.
- Tarjeta gráfica GeForce GTX 1050 o superior.

8.5. Condiciones legales y económicas

Dada la naturaleza docente del proyecto, que como se ha especificado en varias ocasiones a lo largo de la memoria tiene como único objeto el desarrollo de las capacidades ingenieriles del contratista dentro del marco de la realización del Trabajo de Fin de Máster, así como el

de aquellos que en el futuro decidan utilizar algún procedimiento explicado en este estudio o una modificación de este, se considera irrelevante el establecimiento de unas condiciones económicas o legales.

Referencias

[1] Carlos Balaguer y Rafael Aracil-Santoja Antonio Barrientos, Luis Felipe-Peñín. *Fundamentos de Robótica*, volume 2. McGraw-Hill Interamericana de España, 2007.

[2] Análisis de simulación y modelado cinemático de robot scara de cuatro grados de libertad. *International Journal of Computer, Consumer and Control.*

[3] Las soluciones de cinemática inversa de los manipuladores de robots industriales. In *Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM'04.*

[4] Á Noriega González, A García Martínez, and M Muñiz Calvente. Resolución del problema cinemático inverso en un robot scara mediante grupos de assur. *Anales de Ingeniería Mecánica. Área de Ingeniería Mecánica Departamento de Ingeniería Mecánica Universitat Jaume I*, 2012.

[5] Robot scara. *How to Mechatronics*, 2020.

Anexo A

Código Arduino

A.1. Código

En la siguiente sección se va a detallar y se puede observar el código desarrollado en el IDE de Arduino para el correcto funcionamiento del brazo robótico. Para ello se va a mostrar una explicación de cada parte del código desarrollado:

- **Lineas [1 5].** Se incluyen las librerías necesarias para el funcionamiento del brazo robótico. Cabe destacar la librería <AccelStepper.h> para el control de los motores paso a paso y la librería <Servo.h> para el control del servomotor que mueve la pinza.
- **Lineas [7 11].** Se dan nombres a unos valores constantes mediante el comando #define, en este caso, se definen los pines de los finales de carrera.
- **Lineas [13 17].** Se crean los objetos que se relacionan con los motores paso a paso para su posterior parametrización y control.
- **Lineas [18 19].** Se crea la variable que habilita a la CNC Shield el envió de pulsos a los motores paso a paso.
- **Lineas [20 21].** Se crea el objeto que se relaciona con el servomotor para su parametrización y control.
- **Lineas [22 35].** En las siguientes líneas de código se definen los valores críticos de los motores paso a paso que son la velocidad y la aceleración máximas.
- **Lineas [36 56].** Se definen las variables que se van a utilizar. La variable *estado* que almacena el estado actual del robot, *str* variable donde se almacena la trama de datos recibida por el puerto serie, *datos* el vector de datos donde se almacena la información recibida después de ser tratada, *periodo* la variable que almacena la periodicidad del envio de tramas y *TiempoAhora* variable que almacena el tiempo transcurrido en milisegundos.

A.1.1. setup()

La función *setup()* es una función que se ejecuta nada más alimentar el microcontrolador. Por tanto, en esta sección de código se establecen los parámetros de los motores, la velocidad de transmisión de datos y el modo de los pines.

- **Lineas [59 61].** Se define la velocidad de transmisión y el máximo tiempo de espera.
- **Lineas [62 66].** Se definen los finales de carrera como entradas al microcontrolador.
- **Lineas [67 69].** Se define el pin de habilitación de la CNC Shield como salida y se establece a un valor lógico bajo, es decir, se habilitan los motores.

- **Lineas [70 82].** Se establecen los valores críticos de velocidad y aceleración definidos en objetos creados para controlar los motores paso a paso.
- **Lineas [83 84].** Se establece el valor mínimo y máximo del ancho de pulso que se le puede enviar al servomotor en función de las especificaciones de este.
- **Lineas [85 86].** Se llama a la función *HOME* para que el brazo se mueva a una posición de referencia conocida.

A.1.2. loop()

La función *loop()* es una función que se ejecutará continuamente y es el núcleo de la programación de arduino. Esta función se ejecuta tras ejecutarse la función *setup()*. Dentro de la función loop se ha implementado una máquina de estados mediante la función switch. Se ha implementado de esta forma para añadir la posibilidad de incorporar nuevos estados del brazo robótico sin necesidad de modificar el código existente para adaptarlo a los nuevos requerimientos.

- **Case 0. Lineas [93 95].** El brazo robótico se encuentra en este estado cuando se encuentra parado. En este estado el microcontrolador esta constantemente esperando a recibir datos por parte del PC.
- **Case 1. Lineas [97 142].** El brazo robótico se encuentra en este estado cuando se encuentra en movimiento. En las Lineas [99 114] se establecen los parámetros de los motores que se han seleccionado desde el PC que son: la velocidad, la aceleración y la posición a la que se quiere mover cada motor. En las Lineas [115 129] se dan las instrucciones de mover los motores hasta que se alcanzan las posiciones especificadas. Además, se envía información sobre la posición actual del brazo robótico al PC con una frecuencia determinada. Durante esta etapa de movimiento el microcontrolador también puede recibir datos que principalmente se utiliza para detener el movimiento del brazo robótico si el usuario lo requiere desde el PC. En las Líneas [131 138] se determina si se abre la pinza o se cierra. Por último, en las Lineas [139 142] se envía información al PC y se cambia el estado del brazo a parado.

A.1.3. HOME()

La función *HOME()* (Líneas [148 182]) es una función creada para enviar a el brazo robótico a una posición conocida cuando se alimenta por primera vez al microcontrolador. Para enviar al robot a una posición conocida se utilizan sensores mecánicos, en este caso, finales de carrera.

- **Lineas [149 155].** En la siguientes líneas se realiza el *home* del motor 1 mediante el final de carrera 1. Para ello, se mueve el motor en un sentido especifico y a una velocidad

relativamente baja para evitar generar inercias altas durante el frenado de dicho motor debido a que no se existirá una deceleración controlada. Además, se establece el cero como posición absoluta cuando se detecte el final de carrera. Por último, se añade un pequeño retardo de tiempo antes de comenzar con el *home* del siguiente motor.

- **Lineas [156 166].** En la siguientes líneas se realiza el *home* del motor 2 mediante el final de carrera 2. Se realiza de manera análoga al *home* del motor 1.

- **Lineas [167 173].** En la siguientes líneas se realiza el *home* del motor 3 mediante el final de carrera 3. Se realiza de manera análoga al *home* del motor 1 con la excepción que se establece como posición absoluta un valor de -3200. Este valor de posición es el resultado de transformar -90º a pasos del motor que es el ángulo que forma el brazo 2 respecto al brazo 1 cuando se detecta el final de carrera.

- **Lineas [174 180].** En la siguientes líneas se realiza el *home* del motor 4 mediante el final de carrera 4. Se realiza de manera análoga al *home* del motor 1.

- **Lineas [181 181].** Se envían datos al PC mediante la función *EnviarDatos(2)* para comunicarle el finalizado del home.

A.1.4. RecibirDatos()

La función *RecibirDatos()* (Líneas [184 191]) es una función creada para saber si se ha transmitido información del PC. Sólo se entra dentro de esta función si hay datos disponibles por el canal serie.

- **Lineas [187 189].** En las siguientes líneas se lee por el puerto serie hasta detectar el carácter especial $. Tras esto, se llama a la función *ConvertirDatos()* y, por último, se limpia el canal serie.

A.1.5. ConvertirDatos()

La función *ConvertirDatos()* (Líneas [193 205]) es una función creada para tratar la información que se ha recibido del PC. Esta función sólo es llamada cuando se han recibido datos.

- **Lineas [195 204].** En las siguientes líneas se va descomponiendo y tratando el string, que forma la trama de datos recibida por puerto serie, en la información necesaria para establecer las nuevas especificaciones de los motores. Cabe destacar que si el primer dato que se lee es un cero se para el movimiento del brazo robótico.

A.1.6. EnviarDatos(int tipo_envio)

La función *EnviarDatos()* (Líneas [207 223]) es una función creada para enviar datos desde el microcontrolador al PC. Esta función requiere de una variable de entrada que se utiliza para seleccionar el tipo de información que se envía al PC.

- **Lineas [214 216].** En las siguientes líneas se encuentra el envío de la posición de los motores cuando aún se está realizando el movimiento.
- **Lineas [217 219].** En las siguientes líneas se encuentra el envío de la posición de los motores cuando el movimiento ha finalizado.
- **Lineas [220 223].** En las siguientes líneas se envía la información de finalización de home al PC.

```
1  // Librerías
2  #include <AccelStepper.h>
3  #include <math.h>
4  #include <String.h>
5  #include <Servo.h>
6
7  // Finales de carrera
8  #define limitSwitch2 11 // Z
9  #define limitSwitch3 10 // Y
10 #define limitSwitch1 9 // X
11 #define limitSwitch4 A3 // Rotacion pinza
12
13 // Motores - nombre(Tipo:controlador/driver, pin de Paso, pin de dirección)
14 AccelStepper stepper1(1, 2, 5); // X
15 AccelStepper stepper3(1, 3, 6); // Y
16 AccelStepper stepper2(1, 4, 7); // Z
17 AccelStepper stepper4(1, 12, 13); // Rotacion pinza
18 // CNC Shield V3.0
19 const byte enablePin = 8; // Pin que habilita los motores
20 // Servomotor - Pinza
21 Servo PinzaServo;  // create servo object to control a servo
22 // Motores - Variables
23 // Frecuencia de reloj de arduino uno 16MHz
24 // Motor 1
25 int vel_max_m1 = 2000; // Velocidad motor 1 (pasos/s)
26 int ac_max_m1 = 1000; // Aceleracion motor 1 (pasos/s^2)
27 // Motor 2
28 int vel_max_m2 = 400; // Velocidad motor 2 (pasos/s)
29 int ac_max_m2 = 500; // Aceleracion motor 2 (pasos/s^2)
30 // Motor 3
31 int vel_max_m3 = 2000; // Velocidad motor 3 (pasos/s)
32 int ac_max_m3 = 1000; // Aceleracion motor 3 (pasos/s^2)
33 // Motor 4
34 int vel_max_m4 = 2000; // Velocidad motor 4 (pasos/s)
35 int ac_max_m4 = 1000; // Aceleracion motor 4 (pasos/s^2)
36 // Variables
```

```
int estado = 0; // Variable que indica el estado del robot
// estado = 0 - Parado
// estado = 1 - Marcha
String str; // Variable donde se almacenan los datos recibidos
int datos[14]; // Vector de datos
// datos[0] - 1=Marcha / 0=Paro
// datos[1] - Pasos motor 1
// datos[2] - Velocidad motor 1
// datos[3] - Aceleracion motor 1
// datos[4] - Pasos motor 2
// datos[5] - Velocidad motor 2
// datos[6] - Aceleracion motor 2
// datos[7] - Pasos motor 3
// datos[8] - Velocidad motor 3
// datos[9] - Aceleracion motor3
// datos[10] - Pasos motor 4
// datos[11] - Velocidad motor 4
// datos[12] - Aceleracion motor4
// datos[13] - Estado pinza - 1=Cerrar / 0=Abrir
int periodo = 300; // Periodo de envío de tramas de datos
unsigned long TiempoAhora = 0; // Variable que almacena el tiempo
    transcurrido

void setup() {
  //Configuramos el puerto serial
  Serial.begin(115200); // Velocidad de transmision
  Serial.setTimeout(50); // Tiempo de espera ****
  // Entradas
  pinMode(limitSwitch1, INPUT_PULLUP); // Motor 1 - X
  pinMode(limitSwitch2, INPUT_PULLUP); // Motor 2 - Z
  pinMode(limitSwitch3, INPUT_PULLUP); // Motor 3 - Y
  pinMode(limitSwitch4, INPUT_PULLUP); // Motor 4 - ROT
  // CNC SHIELD V3.0
  pinMode(enablePin, OUTPUT);
  digitalWrite(enablePin, LOW); // Se habilitan los motores
  // Parametros motores
  // Motor 1
  stepper1.setMaxSpeed(vel_max_m1); // Establecer velocidad máxima motor 1
  stepper1.setAcceleration(ac_max_m1); // Establecer aceleracion máxima
    motor 1
  // Motor 2
  stepper2.setMaxSpeed(vel_max_m2); // Establecer velocidad máxima motor 2
  stepper2.setAcceleration(ac_max_m2); // Establecer aceleracion máxima
    motor 2
  // Motor 3
  stepper3.setMaxSpeed(vel_max_m3);  // Establecer velocidad máxima motor 3
  stepper3.setAcceleration(ac_max_m3); // Establecer aceleracion máxima
    motor 3
  // Motor 4
  stepper4.setMaxSpeed(vel_max_m4);  // Establecer velocidad máxima motor 4
  stepper4.setAcceleration(ac_max_m4); // Establecer aceleracion máxima
    motor 4
  // Servomotor
  PinzaServo.attach(A0, 550, 2500); // Establecer maximo y minimo del ancho
```

```
    de pulso  650-1400
  // Funcion Home
  HOME();
}

void loop() {
  // Maquina de estados
  switch(estado) {

    case 0: // Robot parado
      RecibirDatos();
      break;

    case 1: // Robot en movimiento
      delay(500);
      // Motor 1
      stepper1.setSpeed(datos[2]);
      stepper1.setAcceleration(datos[3]);
      stepper1.moveTo(datos[1]);
      // Motor 2
      stepper2.setSpeed(datos[5]);
      stepper2.setAcceleration(datos[6]);
      stepper2.moveTo(datos[4]);
      // Motor 3
      stepper3.setSpeed(datos[8]);
      stepper3.setAcceleration(datos[9]);
      stepper3.moveTo(datos[7]);
      // Motor 4
      stepper4.setSpeed(datos[11]);
      stepper4.setAcceleration(datos[12]);
      stepper4.moveTo(datos[10]);
      // Movimiento
      while ((stepper1.currentPosition() != datos[1]) || (stepper2.
    currentPosition() != datos[4]) || (stepper3.currentPosition() != datos
    [7]) || (stepper4.currentPosition() != datos[10])) {
        // Movimiento de motores
        stepper1.run();
        stepper2.run();
        stepper3.run();
        stepper4.run();
        // Envio de datos cada periodo establecido
        if(millis() > TiempoAhora + periodo){
          TiempoAhora = millis();
          EnviarDatos(1);
        }
        // Recibir datos
        RecibirDatos();
      }
      // Pinza
      if (datos[13] == 0) {
        PinzaServo.write(650);
        delay(20);
      } else {
        PinzaServo.write(1400);
```

```
137           delay(20);
138         }
139         // Envio de datos
140         EnviarDatos(0); // Trama de movimiento finalizado
141         estado = 0; // Estado parado
142         break;
143 
144     }
145 
146 }
147 
148 void HOME() {
149   // Home del motor 1 - X
150   while (digitalRead(limitSwitch1) != 1) {
151     stepper1.setSpeed(-200); // Velocidad lenta
152     stepper1.runSpeed();
153     stepper1.setCurrentPosition(0); // Establecer posicion cero cuando se
        pulse el final de carrera
154   }
155   delay(20);
156 //  // Home del motor 2 - Z
157 //  while (digitalRead(limitSwitch2) != 1) {
158 //    stepper2.setSpeed(-200); // Velocidad lenta
159 //    stepper2.runSpeed();
160 //    stepper2.setCurrentPosition(0); // Establecer posicion cero cuando se
        pulse el final de carrera
161 //  }
162 //  delay(20);
163 
164   stepper2.setCurrentPosition(0);
165   delay(20);
166 
167   // Home del motor3 - Y
168   while (digitalRead(limitSwitch3) != 1) {
169     stepper3.setSpeed(-200); // Velocidad lenta
170     stepper3.runSpeed();
171     stepper3.setCurrentPosition(-3200); // Establecer posicion -90 cuando se
         pulse el final de carrera
172   }
173   delay(20);
174   // Home del motor 4 - ROT
175   while (digitalRead(limitSwitch4) != 1) {
176     stepper4.setSpeed(-100); // Velocidad muy lenta
177     stepper4.runSpeed();
178     stepper4.setCurrentPosition(0); // Establecer posicion cero cuando se
        pulse el final de carrera
179   }
180   delay(20);
181   EnviarDatos(2); // Trama de home finalizado
182 }
183 
184 // Funcion de recibir datos por el puerto serie
185 void RecibirDatos() {
186   if(Serial.available()){ //Nos dice si hay datos dentro del buffer
```

```
187     str = Serial.readStringUntil ('$');
188     ConvertirDatos(); // Se llama a la función
189     Serial.flush(); // Se limpia el serial tras recibir y convertir los
        datos
190   }
191 }
192
193 // Funcion de convertir los datos recibidos por el puerto serie en un vector
        de enteros
194 void ConvertirDatos() {
195   for (int i = 0; i < 13; i++) {
196       int index = str.indexOf(","); // ****
197       datos[i] = atol(str.substring(0, index).c_str()); // ****
198       str = str.substring(index + 1); // ****
199       if (datos[0] == 0) { // Si el primer dato es un cero
200         estado = 0; // Estado parado
201         break;
202       }
203     }
204    estado = 1;
205 }
206
207 // Función de enviar datos por el puerto serie
208 // EnviarDatos(0) = Se ha terminado el movimiento
209 // EnviarDatos(1) = Se envia posición de los motores
210 // EnviarDatos(2) = Se envia home terminado
211 void EnviarDatos(int tipo_envio) {
212   String datos_envio;
213
214   if (tipo_envio == 1) {
215     datos_envio =  String(1) + ',' +  String(stepper1.currentPosition()) + '
        ,' +  String(stepper2.currentPosition()) + ',' +  String(stepper3.
        currentPosition()) + ',' +  String(stepper4.currentPosition()) +  '$';
216     Serial.print(datos_envio);
217   } else if (tipo_envio == 0) {
218     datos_envio =  String(0) + ',' +  String(stepper1.currentPosition()) + '
        ,' +  String(stepper2.currentPosition()) + ',' +  String(stepper3.
        currentPosition()) + ',' +  String(stepper4.currentPosition()) +  '$';
219     Serial.print(datos_envio);
220   } else if (tipo_envio == 2) {
221     datos_envio =  String(3) + '$';
222     Serial.print(datos_envio);
223   }
224
225 }
```

Listing A.1: Código arduino

Anexo B

Código Matlab

B.1. Introducción

En el siguiente capítulo se van a explicar las funciones y scripts creados en el software matemático Matlab.

B.2. Funciones

En la siguiente sección se detallan las funciones que se han creado para el desarrollo del presente proyecto.

B.2.1. Cinemática Inversa

A continuación, se muestra la función que se ha creado que, mediante la introducción de ciertos parámetros de entrada, resuelve el problema cinemático inverso del brazo robótico.

```
1  %CINEMÁTICA INVERSA
2  %function [theta] = CinematicaInversa2(x, y, z, L1, L2, Paso_r, codo, H,
       thetha4)
3  function [theta] = CinematicaInversa2(datos)
4      %Variables
5      x = datos(1);
6      y = datos(2);
7      z = datos(3);
8      L1 = datos(4);
9      L2 = datos(5);
10     Paso_r = datos(6);
11     codo = datos(7); %-1 CODO ABAJO // +1 CODO ARRIBA
12     H = datos(8);
13     theta4 = datos(9);
14
15     %Segundo cuadrante o Tercer cuadrante
16     if ( ((x < 0) && (y <= 0)) || ((x <= 0) && (y < 0)) || ((x < 0) && (y >
       0)) || (((abs(x) + abs(y)) >= (L1+L2))) )
17         codo = -1 * codo;
18     end
19
20     %Cálculo del ángulo del brazo 2
21     M = ( x * x + y * y - L1 * L1 - L2 * L2 )...
22           / ( 2 * L1 * L2 );
23     if (((x == 0) && ((y == L1+L2))  || (y == -(L1+L2)))...
24             || ((y == 0) && ((x == L1+L2) || (x == -(L1+L2)))))
25         alpha = atan(((codo) * (0 / (M))));
26     else
27         alpha = atan(((codo) * sqrt(1 - (M * M))) / (M));
28     end
29
30     if (((abs(x) + abs(y)) >= (L1+L2)))
31         theta3 =  alpha;
32     else
```

```
33          theta3 =  pi-alpha;
34      end
35
36      %Cálculo del ángulo del brazo 1
37      theta1 = atan ( y / x ) - atan (( L2 * sin ( theta3 )) ...
38                  / ( L1 + L2 * cos ( theta3 )));
39      % Radianes a grados
40      theta1 = theta1 * 180 / pi;
41      theta3 =  theta3 * 180 / pi;
42      % Modificación del ángulo theta1
43      % Primer cuadrante
44      if ( (x >= 0) && (y >= 0) )
45      end
46      % Segundo cuadrante
47      if ( (x < 0) && (y > 0) )
48          theta1 = 180 + theta1;
49      end
50      % Tercer cuadrante
51      if ( ((x < 0) && (y <= 0)) || ((x < 0) && (y < 0)) )
52          theta1 = 180 + theta1;
53      end
54      % Cuarto cuadrante
55      if ( (x >= 0) && (y < 0) )
56      end
57
58      %Cálculo del ángulo de elevación
59      theta2 = ((H-z) / Paso_r) * 360;
60      % theta2 = ((H-z)*1000) / Paso_r * 360;
61
62      if (theta3>180)
63          theta3 = theta3 - 360;
64      end
65
66      theta4 = theta4 + theta1 + theta3;
67
68      theta = [theta1;theta2;theta3;theta4];
69  end
```

Listing B.1: Cinemática inversa

B.2.2. Cinemática Directa

En la siguiente sección se muestra la función que se ha creado que, mediante la introducción de ciertos parámetros de entrada, resuelve el problema cinemático directo del brazo robótico.

```
1   %CINEMÁTICA DIRECTA
2   function [Posicion] = CinematicaDirecta(theta1, theta2, theta3, L1, L2,
        Paso_r)
3       % Grados a radianes
4       theta1 = theta1 * pi / 180;
```

```
5       %theta2 = theta2 * pi / 180;
6       theta3 = theta3 * pi / 180;
7
8       %Cálculo de la componente x
9       x = L1 * cos(theta1) + L2 * cos(theta1 + theta3);
10
11      %Cálculo de la componente y
12      y = L1 * sin(theta1) + L2 * sin(theta1 + theta3);
13
14      %Cálculo de la componente Z
15      z = (theta2 / 360 * Paso_r);
16
17      %Return
18      Posicion(1) = x;
19      Posicion(2) = y;
20      Posicion(3) = z;
21  end
```

Listing B.2: Cinemática directa

B.2.3. Abrir Puerto serie

La siguiente función ha sido creada para abrir el puerto serie *puerto* que es el parámetro de entrada de dicha función. Lo primero que se hace es ver si existe un objeto serial creado en el puerto especificado y se elimina (líneas [3 4]). Tras esto, se crea el objeto serial, definiendo la velocidad de transmisión de información y el terminador a utilizar en las tramas de datos (líneas [5 7]). A continuación, se establece el tamaño del buffer de entrada y de salida del puerto creado (líneas [9 10]). Por último, se abre el puerto creado y se introduce un pequeño retardo de tiempo para evitar problemas al abrir el puerto.

```
1   %Creamos y abrimos el puerto serie
2   function puerto_serial = AbrirPuerto(puerto)
3       %Se inicializa el puerto serie
4       delete(instrfind({'Port'},{puerto}));
5       %Se crea el puerto serie
6       puerto_serial = serial(puerto,'BaudRate',115200,'Terminator','$');
7       warning('off','MATLAB:serial:fscanf:unsuccessfulRead');
8
9       puerto_serial.OutputBufferSize = 3500;
10      puerto_serial.InputBufferSize = 3500;
11
12      %Se abre el puerto serie
13      fopen(puerto_serial);
14      %Retardo para que se abra el puerto serie
15      pause(0.25);
16  end
```

Listing B.3: Abrir puerto serie

B.2.4. Cerrar Puerto serie

La siguiente función se utiliza para eliminar un puerto serie. Esta función tiene como entrada el puerto que se quiere eliminar. Lo primero que se realiza es cerrar el puerto mencionado (líneas [2 3]), luego se elimina dicho puerto (líneas [4 5]) y, por último, se limpia el puerto (líneas [6 7]).

```
1 function CerrarPuerto(puerto_serial)
2     %Se cierra el puerto serie
3     fclose(puerto_serial);
4     %Se elimina el puerto serie creado
5     delete(puerto_serial)
6     %Se limpia el puerto serie
7     clear puerto_serial
8 end
```

Listing B.4: Cerrar Puerto serie

B.3. Scripts

En la siguiente sección se detallan los scripts que se han creado para el desarrollo del presente proyecto.

B.3.1. Espacio de trabajo

El siguiente código ha sido creado para mostrar gráficamente el espacio de trabajo que tiene el brazo robótico. En este caso, el brazo se ve limitado por el ángulo de giro de los motores que se ha limitado para evitar el enrollamiento de los cables. No limitar el ángulo de giro de los motores puede provocar la ruptura de los cables o el mal funcionamiento de piezas mecánicas con las que los cables entren en contacto.

```
1 %Declaración de variables
2 L1 = 0.228; %Longitud brazo 1
3 L2 = 0.1365; %Longitud brazo 2
4 Paso_r = 10; %Paso de rosca
5
6 %Scara sin limitaciones de ángulo
7 %theta1 = 0:1:360;
8 %theta2 = 0;
9 %theta3 = 0:1:360;
10
11 %%Scara limitado
12 theta1 = 0:1:180;
13 theta2 = 0;
14 theta3 = -90:1:90;
15
16 x = [];
17 y = [];
```

```
18
19 venProceso = waitbar(0, 'Procesando...');
20
21 for i=1:length(theta1)
22
23     waitbar(i/length(theta1),venProceso)
24
25     for j=1:length(theta3)
26 %         [Posicion] = CinematicaDirecta(theta1(i), theta2(j), theta3, L1,
       L2, Paso_r);
27         [Posicion] = CinematicaDirecta(theta1(i), theta2, theta3(j), L1, L2,
        Paso_r);
28         x = [x Posicion(1)];
29         y = [y Posicion(2)];
30     end
31 end
32
33 %Graficar espacio de trabajo
34 figure;
35 hold on;
36 plot(x,y,'.');
37 plot([-0.45 0.45],[0 0],'k');
38 plot([0 0],[-0.45 0.45],'k');
39 axis([-0.45 0.45 -0.45 0.45]);
40 grid on;
```

Listing B.5: Espacio de trabajo

Anexo C

Simscape

C.1. Entorno simulación

En la siguiente sección se observan los subgrupos creados en Simscape que muestran la composición y estructura dentro de los grupos.

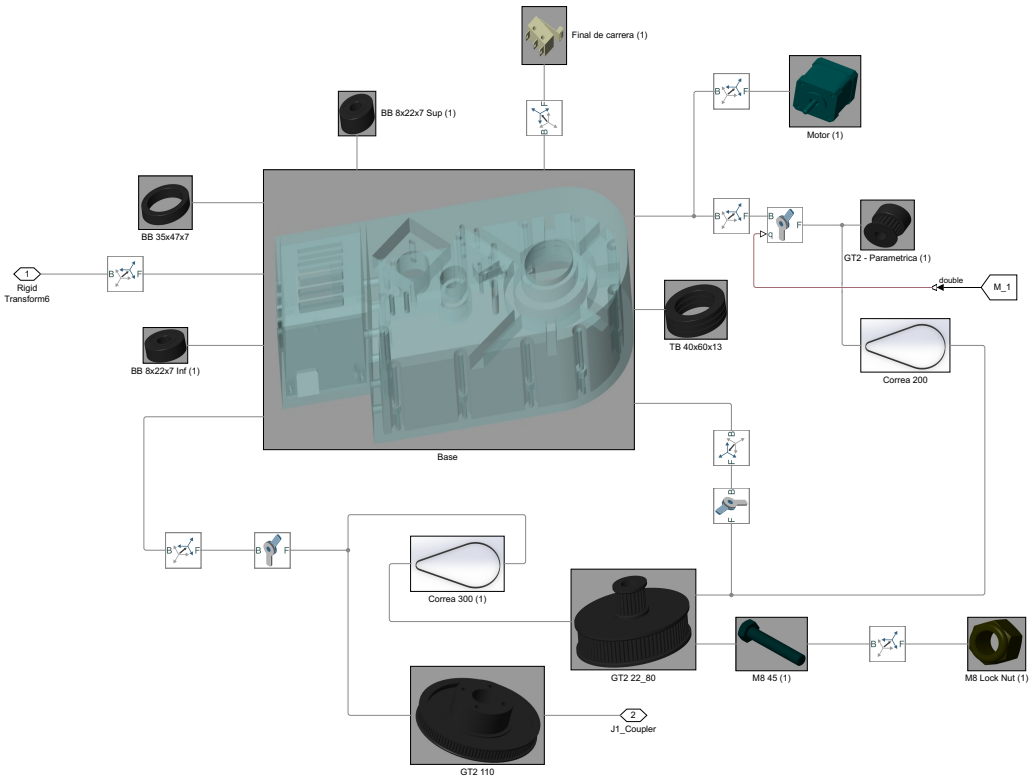


Figura C.1: Simulación - Base

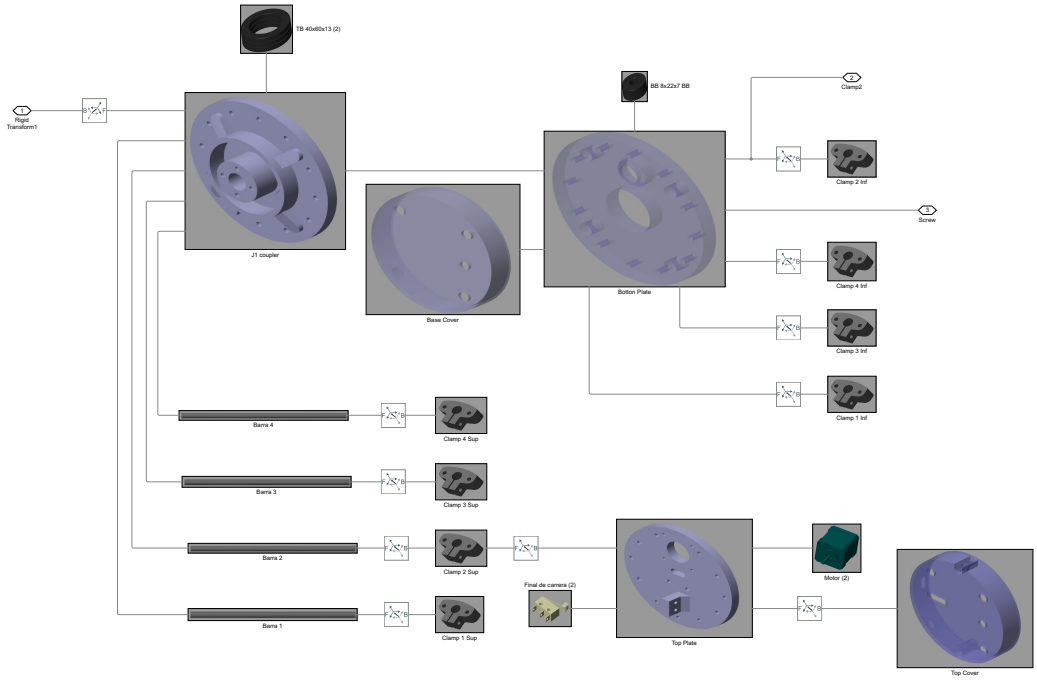


Figura C.2: Simulación - Barras

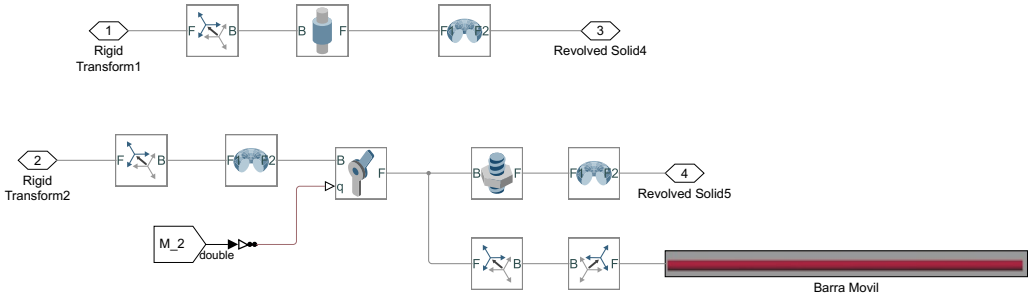


Figura C.3: Simulación - Barra móvil

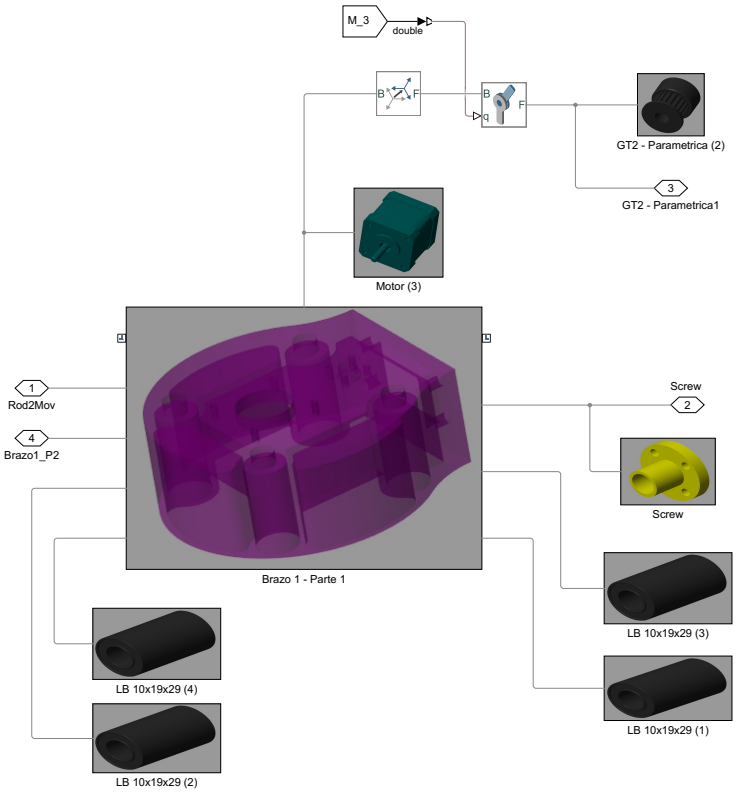


Figura C.4: Simulación - Brazo 1 - Parte 1

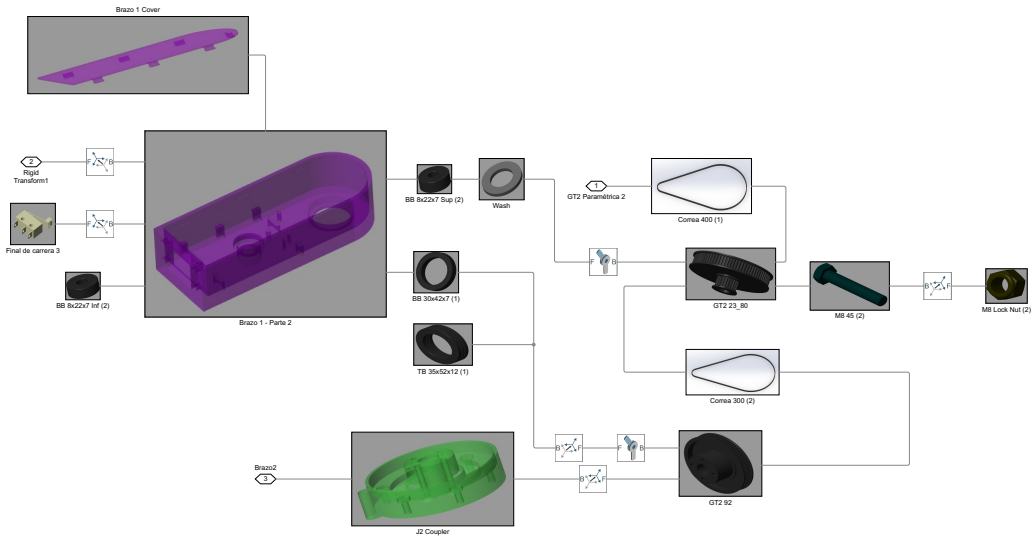


Figura C.5: Simulación - Brazo 1 - Parte 2

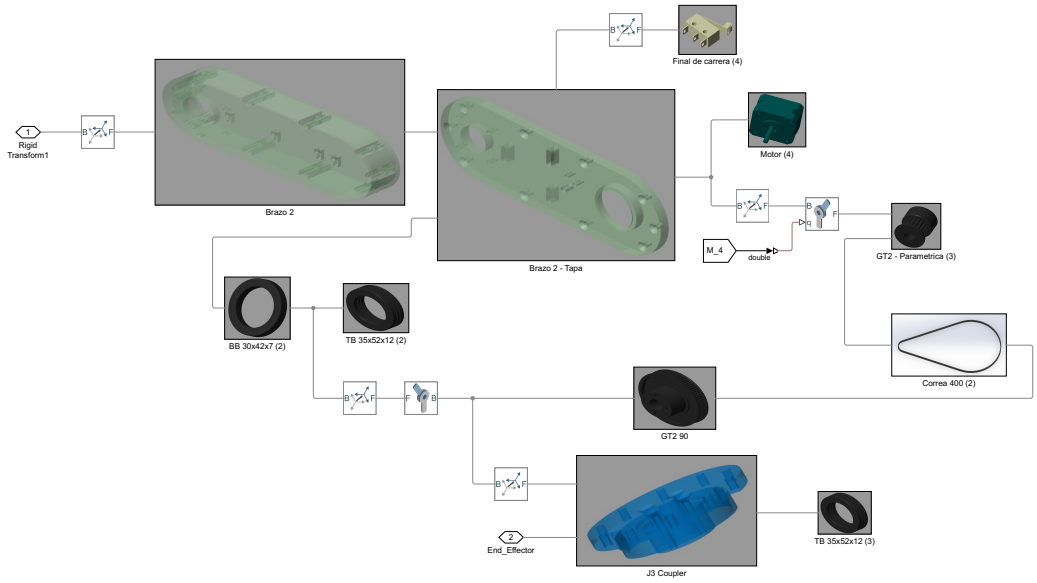


Figura C.6: Simulación - Brazo 2

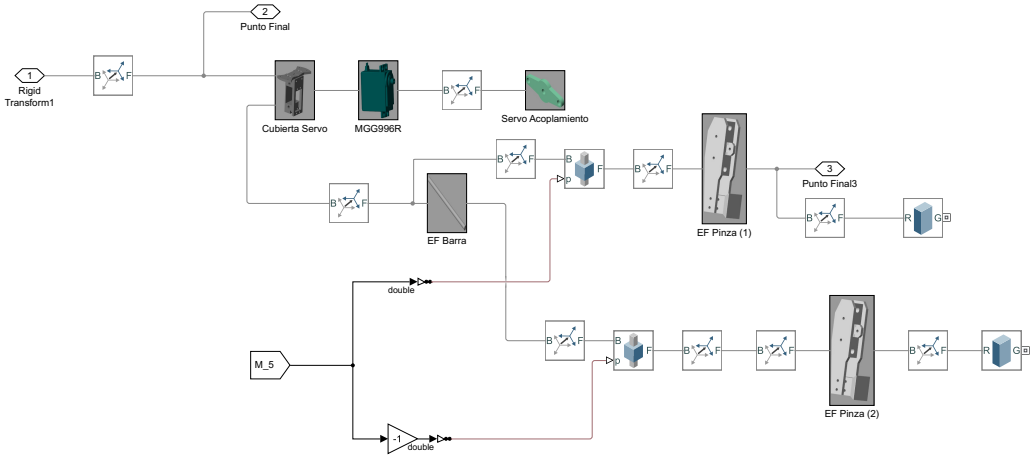


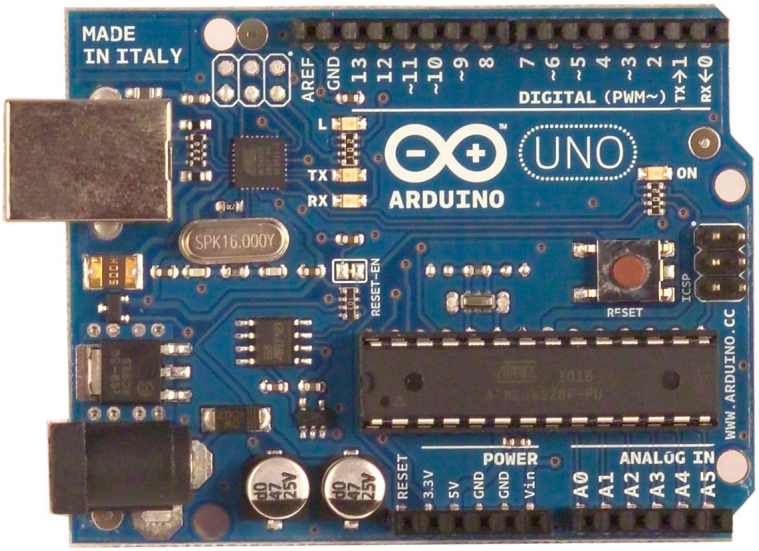
Figura C.7: Simulación - Pinza

Anexo D

Datasheet

D.1. Arduino Uno

Arduino UNO



CE

Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Index

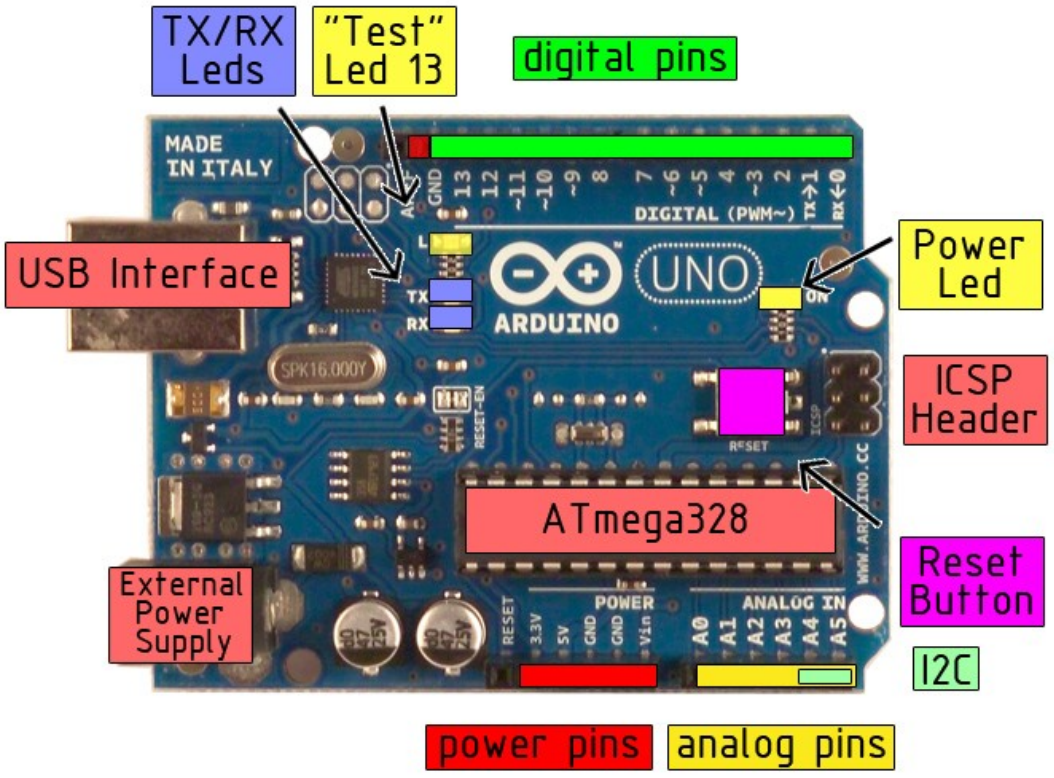
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

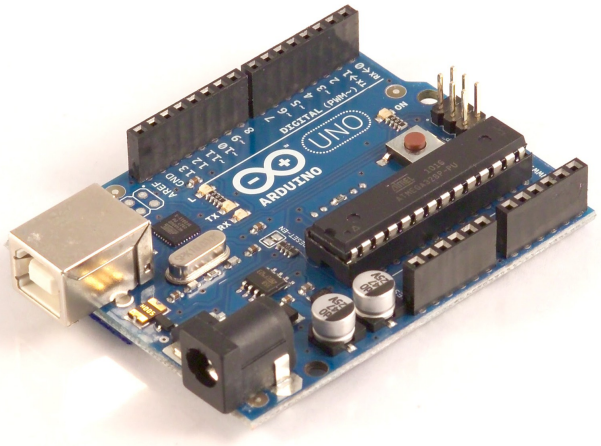
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.



How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

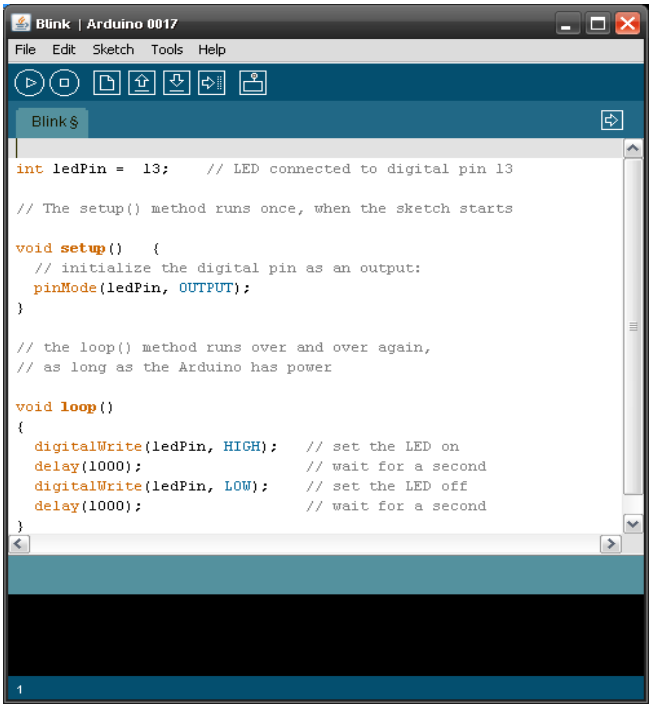
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

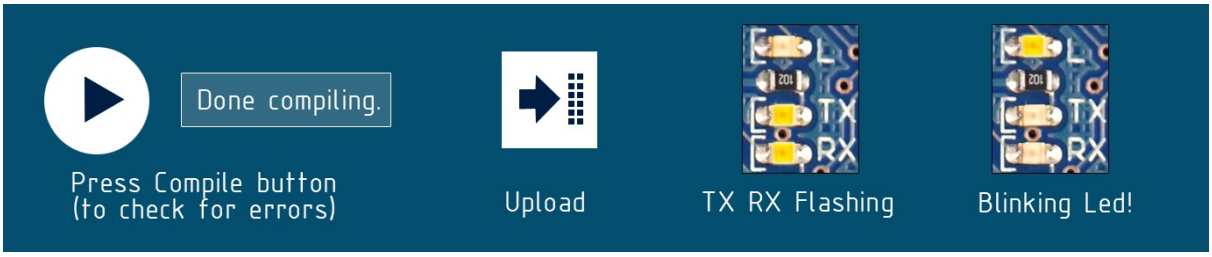
File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

Once you have your skecth you'll see something very close to the screenshot on the right.

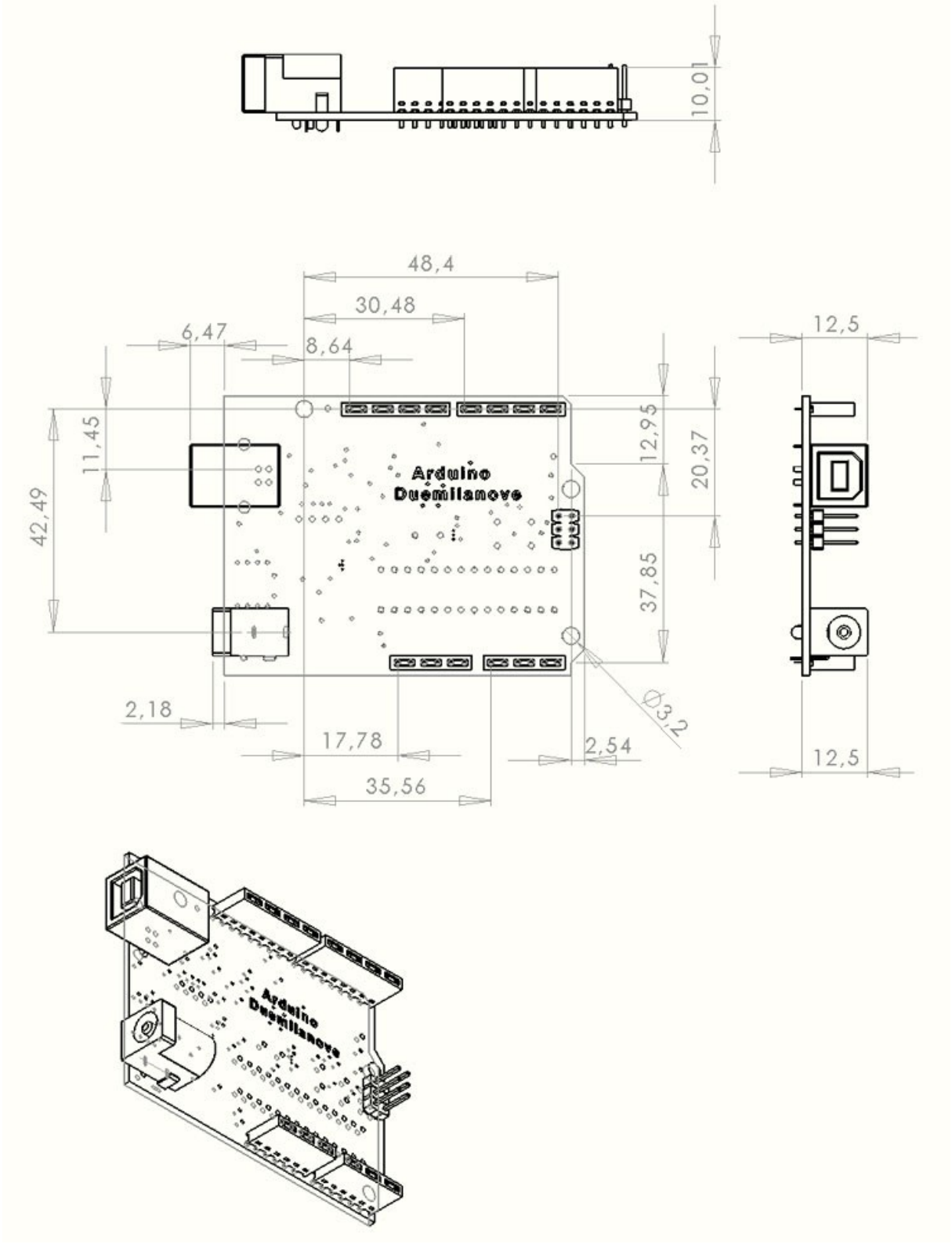
In **Tools>Board** select

Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.





Dimensioned Drawing



Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

Enviromental Policies

The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

D.2. CNC shield V3

Protoneer.co.nz
Electronic Prototyping Specialists

Arduino CNC Shield - 100% GRBL Compatable

2015/08/30 8:01 pm / 258 Comments / Bertus Kruger / Featured

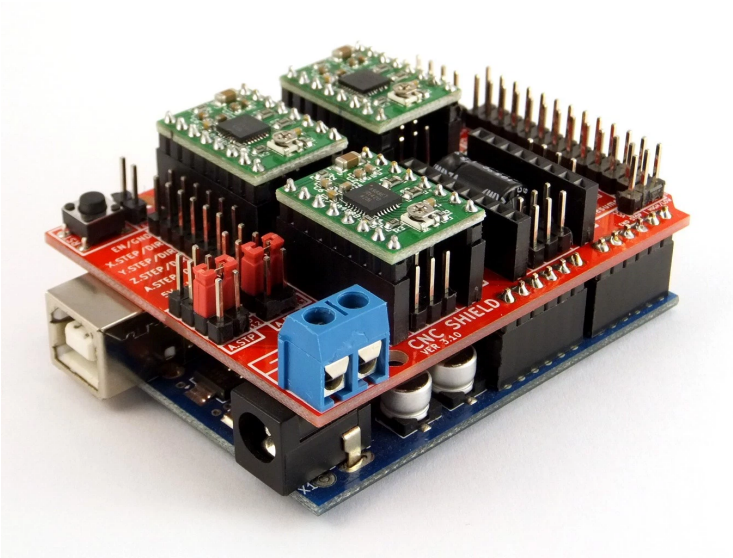
Do it yourself CNC projects are popping up everywhere and we decided that we wanted to contribute to the growth.

Here are a few of our design goals:

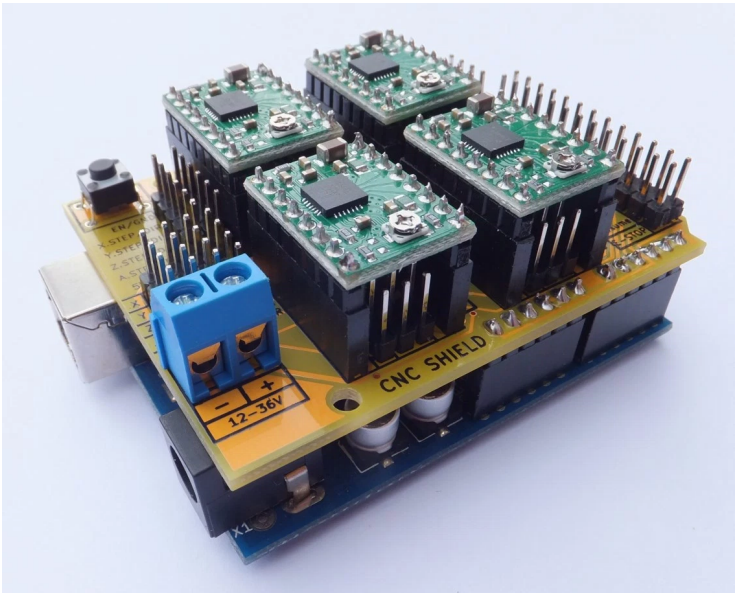
- **Modular Design** – We wanted to do more than just keep cost down. We wanted things to be reusable and up-gradable. (Arduino , Individual Stepper Motor Drivers and more...)
- **Compact Design** – Squeezing a 4 axis design into a board the same size and Arduino Uno.
- **Opensource Software** – 100% GRBL compatible (G-Code Interpreter)
- **Opensource Hardware** – Arduino has opened up the power of micro-controllers to everyone. (Easy but powerful computing)
- **Evolving Development** – We are keen to improve on the design and welcome all feedback.

NOW AVAILABLE at our Ebay store… http://stores.ebay.com/Protoneer … or in assembled from from Elecrow.com

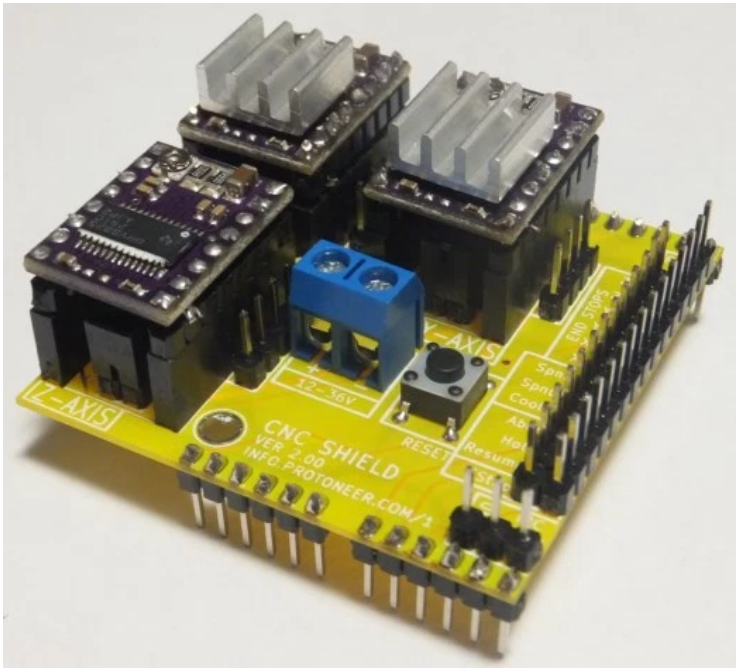
Arduino CNC Shield V3.10 – GRBL v0.9 compatible (PWM Spindle + Soft limits)



Arduino CNC Shield V3.00

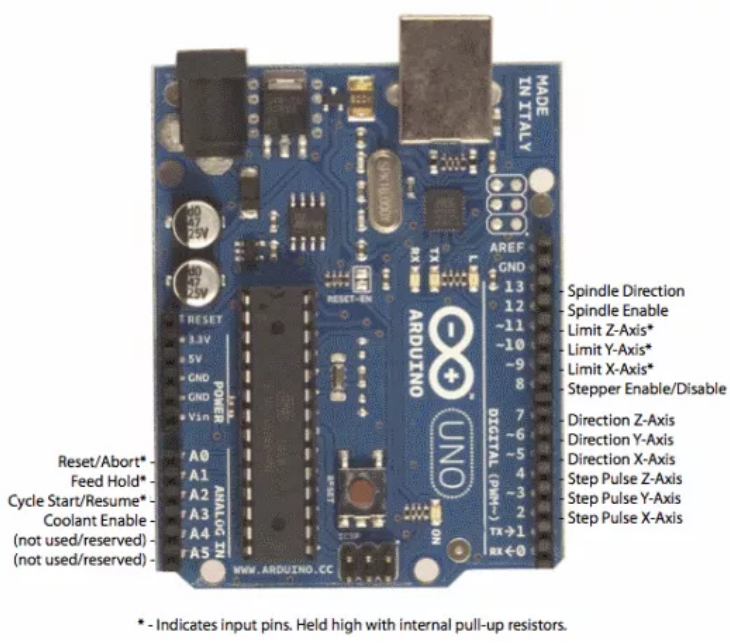


Arduino CNC Shield V2.00



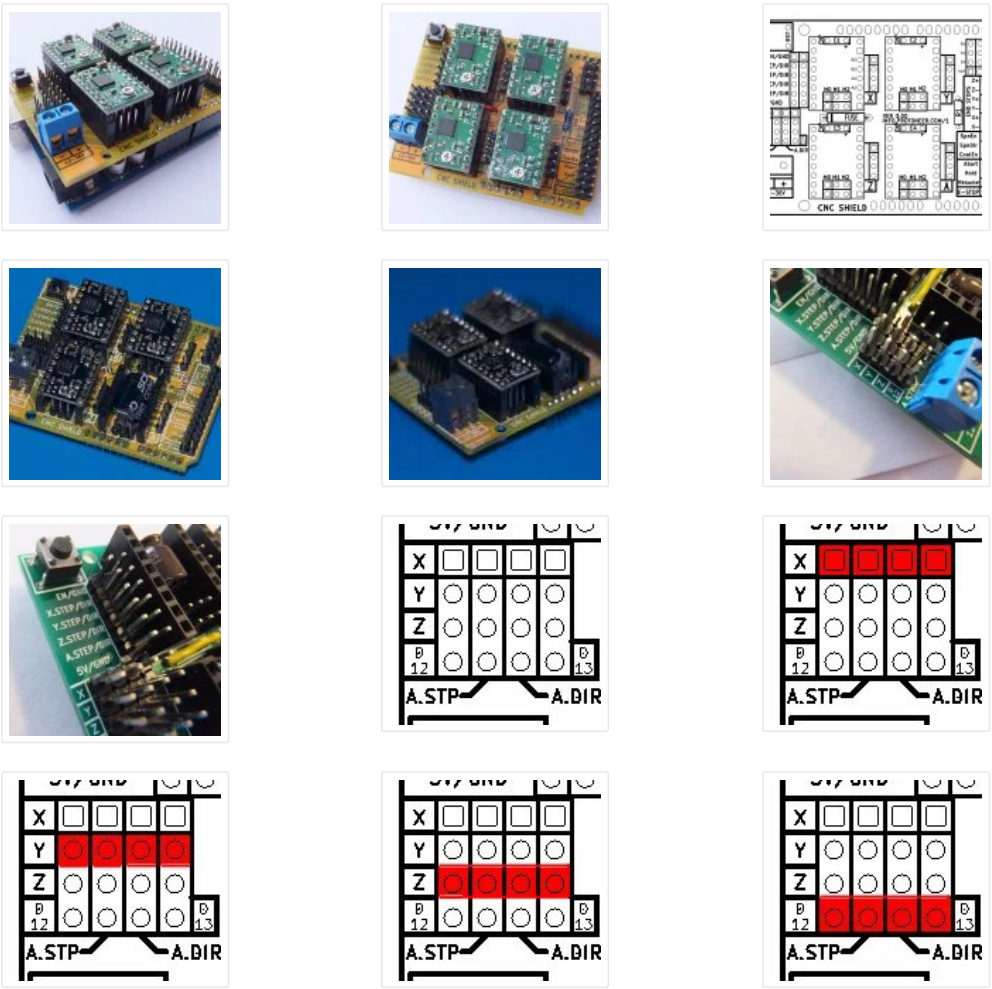
Board Layout

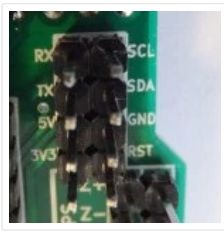
The following image displays the functionality of the Arduino pins as used by GRBL.

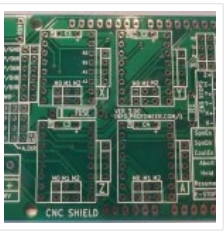


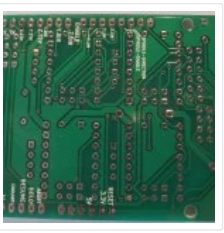
GRBL Pin Layout

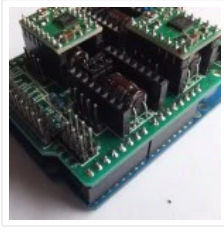
We have designed the Arduino CNC Shield to use all the pins that GRBL implemented. We have also added a few extra pins to make things a little easier.

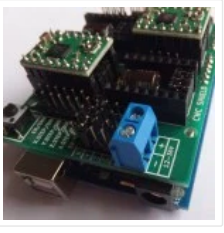


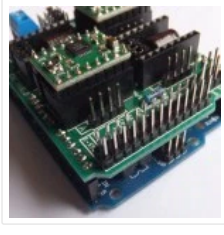

RX
TX
5V
3V3
SCL
SDA
GND
RST
Z-

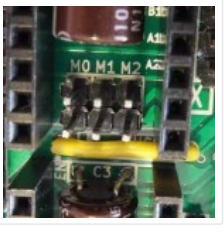

CNC SHIELD

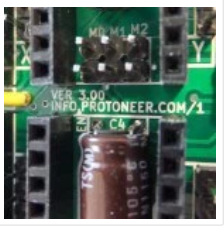


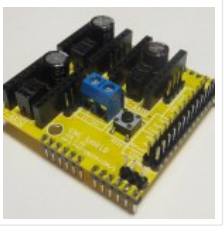


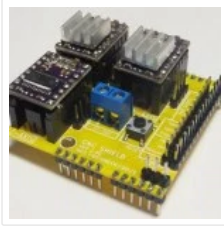


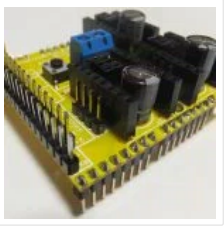


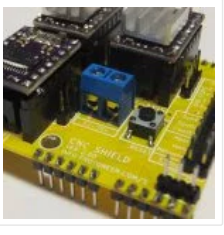

M0 M1 M2

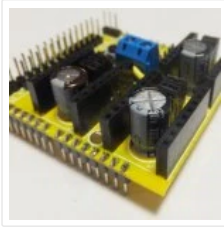

M0 M1 M2
VER 3.00
INFO.PROTONEER.COM/1

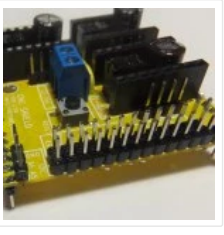


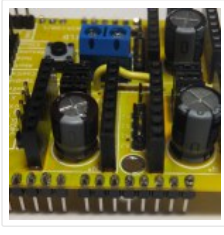




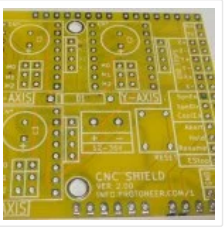


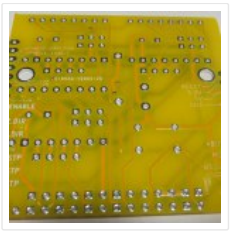


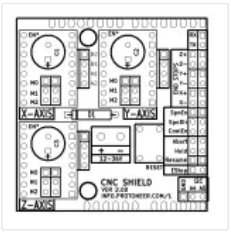


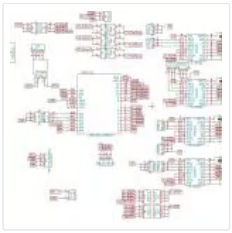



CNC SHIELD
Z-AXIS


CNC SHIELD







Extra pins:

- Limit switch pins have been doubled up so that each axis has a “Top/+” and “Bottom/-“. This makes it easier to install two limit switches for each axis. (For use with a normally open switch)
- EStop – These pins can be connected to an emergency stop switch. This does the same as the RESET button on the Arduino board. (We do advice that an extra emergency button also be installed that cuts power to all machinery. **A REAL EMERGENCY BUTTON**)
- Spindle and coolant control has their own pins.
- External GRBL Command Pins have been broken out allowing you to add buttons for Pause/Hold , Resume and Abort.
- Serial Pins (D0-1) and I2C Pins (A4-5) have their own break out pins for future extensions. I2C can later be implemented by software to control things like spindle speed or heat control.
- Version 3.00 of the board added a jumpers to configure the 4th axis(Clone the other axis’s or run from Pin D12-13), Comms Header(RX+TX , I2C) and a Stepper Control Header(All Pins needed to run 4 steppers)

Bill Of Materials

- Arduino CNC Shield PCB
- 100uF 50v 8mm High Capacitors (493-3289-ND)
- Jumpers (A26228-ND)
- 8-Pin Female Headers(S7006-ND)
- Male Headers(A26514-40-ND)
- Tactile Switch (450-1650-ND)
- Screw Terminal 5mm (A97996-ND)
- Pololu Stepper Drivers . (Please note that the shield was designed to work with A4988 compatable polulu drivers)
- 10K pull-up resistors

Assembly Instructions

Click here for the Assembly instructions for the Arduino CNC Shield V3.XX
Click here for the Assembly instructions for the Arduino CNC Shield V2.XX

GRBL Software/Firmware

GRBL is opensource software that runs on an Arduino Uno that takes G-Code commands via Serial and turns the commands into motor signals.

The GRBL source Code is located here.

I have also written a few turorials on uploading the firmware onto an Arduino Board:

- Quick GRBL setup guide for Windows
- GRBL Arduino Library – Use the Arduino IDE to flash GRBL directly to your Arduino

Versions

- **Version 3.10**
  - Added Support for GRBL 0.9v with PWM Spindle.
- **Version 3.00 (4 Axis)**
  - Enlarged board to add a 4th Axis that can clone the X,Y or Z axis. With a 4th option to use pin D12-13 to control it.(Setting up the 4th Axis)
  - Added a breakout header for all the Axis’s.
  - Added a communication header for UART(Serail) and I2C.

- Added the a connector for an optional fuse.(Fuse not supplies as it needs to be selected for the current that will be used.)
- Capacitors are mounted horizontally giving more clearance between them and the stepper drivers. Good for ventilation.
- Added a pull-up resistor on the axis enable pins. This prevents the pin from being in a floating state.
- Added 2 mounting holes

- **Version 2.02 (3 Axis)**
  - Fixed High Voltage label
  - Removed Diode D1.
  - Reduced the number of Via's.
- **Version 2.01**
  - Added a 5V Breakout
  - Filled in both sides with Ground Copper
  - Moved RX/TX pins to the side so that 26-Pin header can be used. Same as the headers on a Raspberry Pi.
  - Small Text Changes
- **Version 2.00**
  - First official version of the CNC Shield.
  - All pins used by GRBL has been broken out.

Gerber Files

Arduino CNC Shield Ver3.00 – Gerber Files

Arduino CNC Shield Ver2.xx – Gerber Files

License and Warnings

This is a work in progress design. All liabilities are on the users at their own risk and they take full responsibility for any harm that might happen to them or their property.

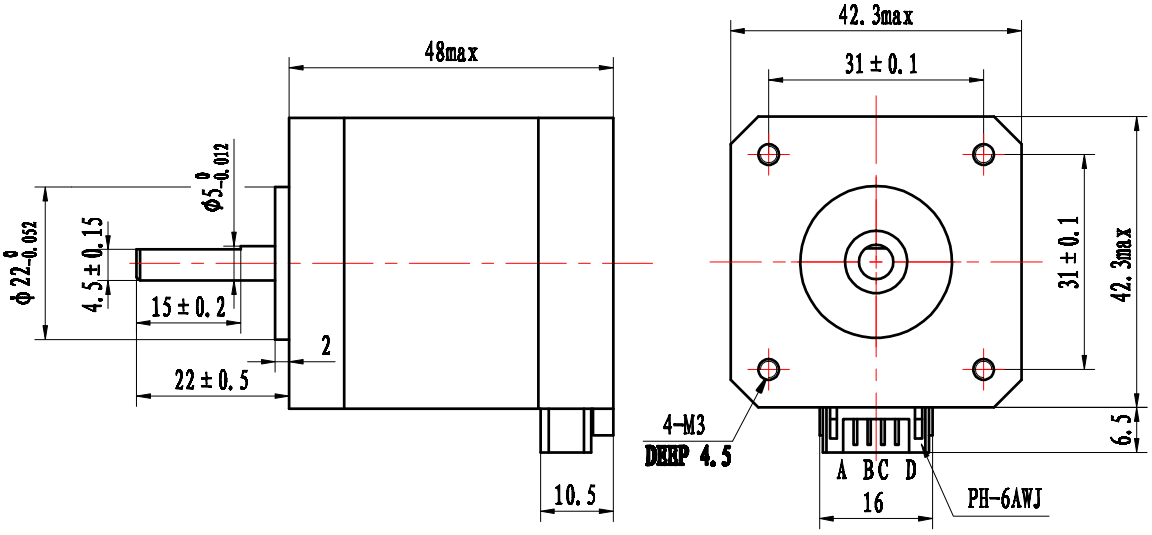
Extra Reading

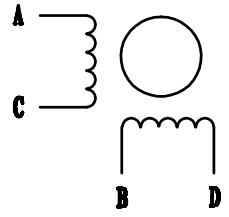
- **Build Your Own CNC Machine** – By James Floyd Kelly and P. Hood-Daniel
- **CNC Machining Handbook: Building, Programming, and Implementation**– By Alan Overby
- **Pololu Stepper Motor Drivers** – http://www.pololu.com/catalog/category/120
- **Pololu FQA covering how to connect stepper motors** – http://www.pololu.com/catalog/product/1182/faqs
- **GRBL Wiki including upload, setup and running** – https://github.com/grbl/grbl/wiki/_pages
- **GRBL Supported G-Codes** – www.shapeoko.com/wiki/index.php/Grbl

D.3. Nema 17 - 17HD48002-22B

Busheng

17HD48002-22B



General specification		Electrical specification		Wiring diagram
Step angle	1.8° ±5%	Rated voltage	3.06V	
Number of phase	2	Rated current	1.7A	
Insulation resistance	100MΩmin. (500V DC)	Resistance per phase	1.8Ω±10%	
Insulation class	Class B	Inductance per phase	3.8mH±20%	
Rotor inertia	82g.cm²	Holding torque	560mN.m	
Mass	0.36kg	Detent torque	24mN.m	

标记	处数	更改文件号	签字	日期

设计		标准化	
		批准	
审核			
工艺		日期	

图样标记	重量	比例
共 1 张	第 1 张	

东莞步升电机有限公司
DONGGUAN BUSHENG MOTOR CO.,LTD.

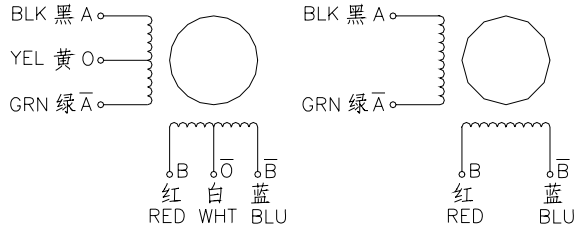
17HD48002-22B型步进电机
规格书

BS3.00127

D.4. Nema 17 - 17HS2408

2 Phase Hybrid Stepper Motor
17HS series-Size 42mm(1.8 degree)

Wiring Diagram:



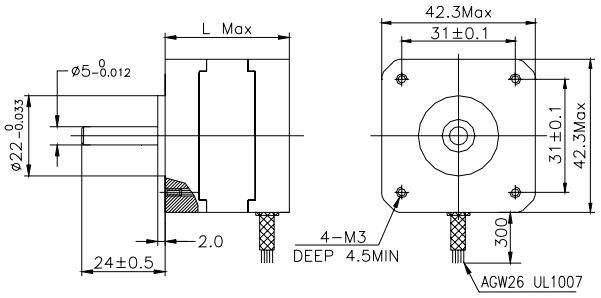
UNI-POLAR(6 LEADS) BI-POLAR(4LEADS)

Electrical Specifications:

Series Model	Step Angle (deg)	Motor Length (mm)	Rated Current (A)	Phase Resistance (ohm)	Phase Inductance (mH)	Holding Torque (N.cm Min)	Detent Torque (N.cm Max)	Rotor Inertia (g.cm²)	Lead Wire (No.)	Motor Weight (g)
17HS2408	1.8	28	0.6	8	10	12	1.6	34	4	150
17HS3401	1.8	34	1.3	2.4	2.8	28	1.6	34	4	220
17HS3410	1.8	34	1.7	1.2	1.8	28	1.6	34	4	220
17HS3430	1.8	34	0.4	30	35	28	1.6	34	4	220
17HS3630	1.8	34	0.4	30	18	21	1.6	34	6	220
17HS3616	1.8	34	0.16	75	40	14	1.6	34	6	220
17HS4401	1.8	40	1.7	1.5	2.8	40	2.2	54	4	280
17HS4402	1.8	40	1.3	2.5	5.0	40	2.2	54	4	280
17HS4602	1.8	40	1.2	3.2	2.8	28	2.2	54	6	280
17HS4630	1.8	40	0.4	30	28	28	2.2	54	6	280
17HS8401	1.8	48	1.7	1.8	3.2	52	2.6	68	4	350
17HS8402	1.8	48	1.3	3.2	5.5	52	2.6	68	4	350
17HS8403	1.8	48	2.3	1.2	1.6	46	2.6	68	4	350
17HS8630	1.8	48	0.4	30	38	34	2.6	68	6	350

***Note:** We can manufacture products according to customer's requirements.*

Dimensions: unit=mm



Motor Length:

Model	Length
17HS2XXX	28 mm
17HS3XXX	34 mm
16HS4XXX	40 mm
16HS8XXX	48 mm

Anexo A

Interfaz de usuario

A.1. Manual

En la siguiente sección se busca dar una pequeña explicación de la utilización de la interfaz de usuario. La interfaz creada es sencilla en cuanto a la utilización por parte del usuario.

En la Fig. A.1 se puede observar el aspecto de la interfaz de usuario creada. A continuación, se procede a explicar detalladamente cuál es la función de cada parte:

- *Parámetros.* En esta sección se permite al usuario modificar tanto las velocidades como las aceleraciones de cada uno de los motores paso a paso. Estas variables tienen limitaciones fijadas para evitar que los motores actúen fuera de su punto óptimo de funcionamiento.

- *Pasos a enviar.* Se trata de una sección meramente informativa que muestra la información en pasos a enviar al microcontrolador Arduino UNO.

- *Panel2.* En el siguiente recuadro se encuentra el botón para enviar información al microcontrolador. Este aparece deshabilitado al arrancar la interfaz y esto es debido a que para que se deshabilite tiene que recibir la información de que el HOME del robot ha sido realizado por parte del microcontrolador. Además, este botón se encuentra deshabilitado cuando el robot esta en movimiento, hasta que recibe información por parte del microcontrolador de que el movimiento se ha realizado.

- *Pinza.* En este recuadro se permite seleccionar si entre abrir o cerrar la pinza al final del movimiento descrito.

- *Estado.* Esta sección de la interfaz de usuario está enfocada en informar al usuario del estado del robot y en el futuro de otros posibles datos de interés.

- *Alarmas.* En el siguiente recuadro se van a mostrar una serie de avisos en forma escrita para informar al usuario. Por ejemplo se muestra cuando se ha terminado el HOME, si se está realizando un movimiento o si el movimiento ha sido realizado.

- *Gráfica.* En la siguiente sección de la interfaz se muestran gráficas de la posición actual del robot, además, del valor de las coordenadas articulares en ese momento. Esta parte de la interfaz es meramente informativa para el usuario y no sería necesaria a no ser que el *Personal Computer*s (PCs) de control se encuentre a cierta distancia del robot real.

- *Modo de funcionamiento.* En el siguiente recuadro se permite seleccionar el modo de funcionamiento que se va a tener a la hora de manejar el robot, es decir, si se quiere mover el robot de manera directa o inversa.

- *Directa.* Esta sección se puede observar en la Fig. A.1. Como se esta resolviendo el problema cinemático directo, el usuario puede introducir directamente las coordenadas articulares de los motores.

- *Inversa.* Esta sección se puede observar en la Fig. A.2. Como se esta resolviendo el problema cinemático inverso, el usuario puede introducir las coordenadas espaciales de la pinza para que el robot sitúe la herramienta en dichas posiciones especificadas.

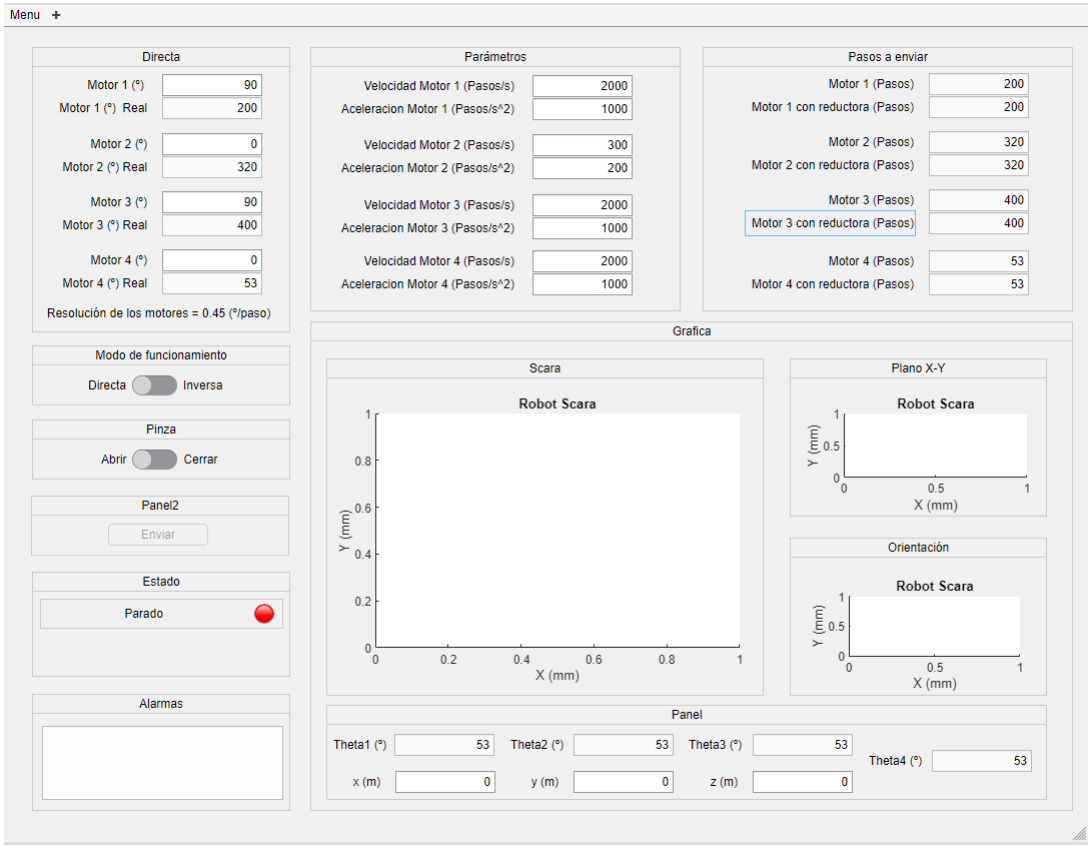


Figura A.1: Interfaz de usuario 1

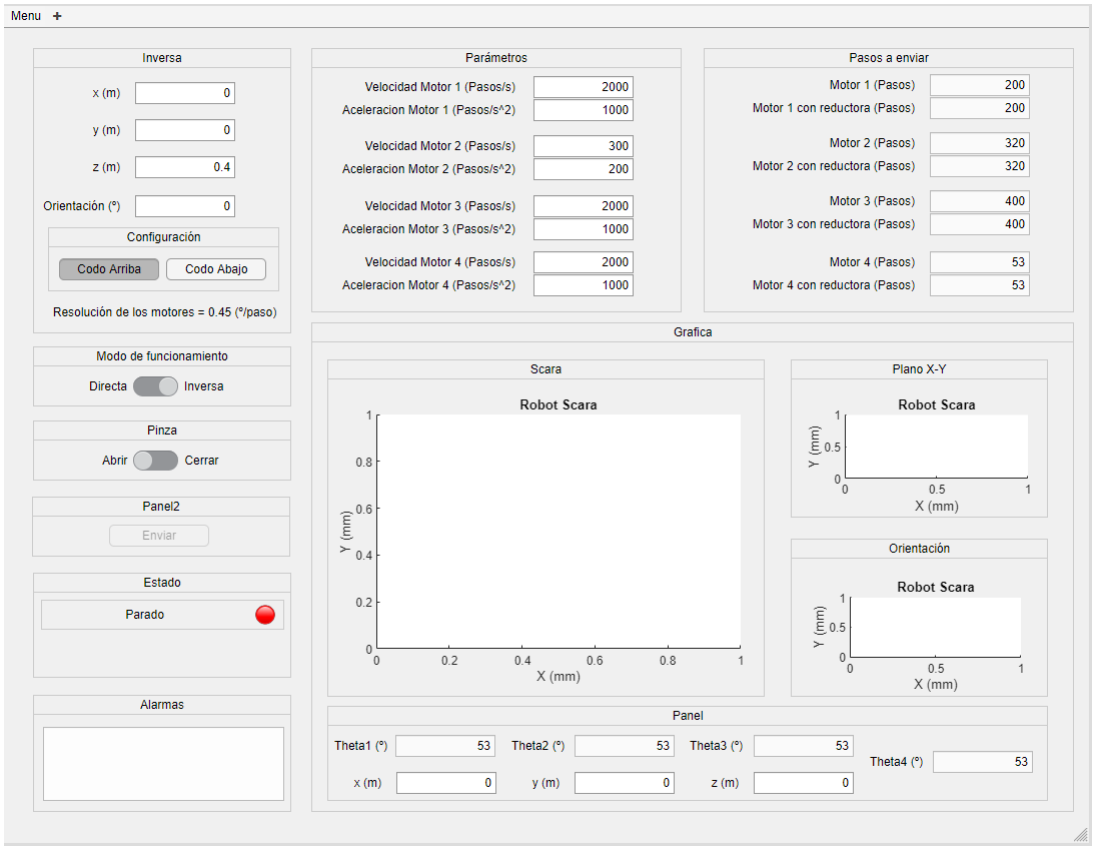


Figura A.2: Interfaz de usuario 2