Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Construcción y programación de un brazo robótico de madera

Autor: Aythami Sosa Alemán
Tutor: Ignacio Alvarado Aldea

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Construcción y programación de un brazo robótico de madera

Autor:
Aythami Sosa Alemán

Tutor:
Ignacio Alvarado Aldea
Profesor titular

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Carrera: Construcción y programación de un brazo robótico de madera

Autor: Aythami Sosa Alemán

Tutor: Ignacio Alvarado Aldea

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis maestros

A mis amigos

Agradecimientos

Me gustaría expresar mi agradecimiento a mi familia especialmente, por el sacrificio y el apoyo que me han dado a lo largo de mi formación académica. A mis compañeros, por ayudarme a comprender conceptos difíciles y hacer que este proyecto pueda realizarse. A mis amigos, por el apoyo y los momentos que han hecho de la formación un buen recuerdo. A mi pareja, por darme el cariño y las fuerzas que necesitaba. A mi tutor, por poder darme las herramientas que necesitaba y guiarme durante el proyecto.

Resumen

El proyecto que se presenta, consiste en la construcción y programación de un brazo robótico de madera para su uso didáctico en las asignaturas que se requiera usar para la fácil comprensión del alumnado ante estos nuevos conocimientos.

Lo primero que haremos será explicar el modelo usado para el robot y su construcción, las piezas usadas, la electrónica y los materiales, así como las medidas aproximadas de cada una de ellas ya que algunas son importantes a la hora de programar.

Por último, se hablará del lenguaje que usaremos para programarlo, así como el algoritmo que se ha usado para su funcionamiento.

Abstract

The project presented consists in the construction and programming of a robotic wooden arm for its didactic use in the subjects that are required to be used for the easy understanding of the students before this new knowledge.

The first thing we will do is explain the model used for the robot and its construction, the parts used, the electronics and the materials, as well as the approximate measurements of each one of them since some are important when programming.

Finally, we will talk about the language we will use to program it, as well as the algorithm that has been used for its operation.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

i	i-ésima articulación del robot
t	Tiempo
j(t)	Posición respecto del tiempo
j_f	Posición final
j_{in}	Posición inicial
t_{in}	Tiempo inicial
t_f	Tiempo final
$\phi(t)$	Ángulo roll
ϕ_{in}	Ángulo roll inicial
ϕ_f	Ángulo roll final
$\theta(t)$	Ángulo pitch
θ_{in}	Ángulo pitch inicial
θ_f	Ángulo pitch final
$\psi(t)$	Ángulo yaw
ψ_{in}	Ángulo yaw inicial
ψ_f	Ángulo yaw final
p.e.	Por ejemplo
$q\,(t)$	Angulo de la articulación
p	muestreo p-ésimo
n	Nº de puntos muestreados
dmv	Duración de cada tramo del muestreo

1 INTRODUCCIÓN

1.1 Justificación

El trabajo desarrollado a lo largo de este proyecto se encuentra englobado dentro del campo de la robótica manipuladora. Los robots manipuladores son los más usados en las industrias, convirtiéndose en uno de los campos más importantes de entender. Por ello, es importante que los alumnos aprendan a programar y controlar este tipo de robots. Este motivo tan importante para el desarrollo de futuros ingenieros robóticos, es el que ha incitado a crear este proyecto que ayudará al aprendizaje de los mismos a la hora de controlar robots manipuladores.

1.2 Objetivos

Con este proyecto se busca utilizar los métodos aprendidos durante la carrera sobre el control de robots manipuladores e implementarlo en un robot real que además sea económico. Con este método, nos aseguramos de que cada alumno pueda tener un robot con el que aprender.

Lo primero que haremos será buscar materiales económicos que nos puedan servir para construir el robot. Seguidamente, calcularemos su modelo cinemático para posteriormente controlarlo y crear un generador de trayectoria. Toda la parte de control se hará primero mediante una simulación y luego se implementará en el robot. Con esto podremos comprobar también si el robot se mueve como debería y todos los cálculos son correctos.

2 CONCEPTOS BÁSICOS Y MATERIALES

En este capítulo haremos una introducción a los pasos que vamos a seguir a la hora de controlar y programar el robot. También, se explicará los distintos programas que se usaron, así como las características de los distintos dispositivos electrónicos como la placa.

2.1 Conceptos básicos

Empezaremos explicando las características básicas a tener en cuenta de los robots, así como, como de los distintos tipos de robots que hay actualmente.

2.1.1 Robótica

La robótica es la rama de la ingeniería mecánica, de la ingeniería eléctrica, de la ingeniería electrónica, de la ingeniería biomédica, y de las ciencias de la computación, que se ocupa del diseño, construcción, operación, estructura, manufactura, y aplicación de los robots.

Los robots se pueden clasificar de dos formas. La primera manera de clasificarlos es según su cronología:

- 1º generación: Robots manipuladores. Son sistemas mecánicos multifuncionales con un sencillo sistema de control, bien manual, de secuencia fija o de secuencia variable.
- 2º generación: Robots de aprendizaje. Repiten una secuencia de movimientos que ha sido ejecutada previamente por un operador humano. El modo de hacerlo es a través de un dispositivo mecánico. El operador realiza los movimientos requeridos mientras el robot le sigue y los memoriza.
- 3º generación: Robots con control sensorizado. El controlador es un ordenador que ejecuta las órdenes de un programa y las envía al manipulador o robot para que realice los movimientos necesarios.

La segunda manera es según su estructura:

- Poliarticulados: En este grupo se encuentran los robots de muy diversa forma y configuración, cuya característica común es la de ser básicamente sedentarios (aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados) y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas, y con un número limitado de grados de libertad. En este grupo se encuentran los robots manipuladores, los robots industriales y los robots cartesianos, que se emplean cuando es preciso abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o reducir el espacio ocupado en el suelo.

Figura 2-2-1. Robot poliarticulado

- Móviles: Son Robots con gran capacidad de desplazamiento, basados en carros o plataformas y dotados de un sistema locomotor de tipo rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores. Estos robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación. Guiados mediante pistas materializadas a través de la radiación electromagnética de circuitos empotrados en el suelo, o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos y están dotados de un nivel relativamente elevado de inteligencia.

Figura 2-2-2. Robot móvil

- Androides: Son los tipos de robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano. Actualmente, los androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica, y destinados, fundamentalmente, al estudio y experimentación. Uno de los aspectos más complejos de estos robots, y sobre el que se centra la mayoría de los trabajos, es el de la locomoción bípeda. En este caso, el principal problema es controlar dinámica y coordinadamente en el tiempo real el proceso y mantener simultáneamente el equilibrio del Robot. Vulgarmente se los suele llamar "marionetas" cuando se les ven los cables que permiten ver cómo realiza sus procesos.

Figura 2-3. Androide

- Zoomórficos: Los robots zoomórficos, que considerados en sentido no restrictivo podrían incluir también a los androides, constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos. A pesar de la disparidad morfológica de sus posibles sistemas de locomoción es conveniente agrupar a los Robots zoomórficos en dos categorías principales: caminadores y no caminadores. El grupo de los robots zoomórficos no caminadores está muy poco evolucionado. Los experimentos efectuados en Japón basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación. Los Robots zoomórficos caminadores multípedos son muy numerosos y están siendo objeto de experimentos en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrenos, pilotados o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos robots serán interesantes en el campo de la exploración espacial y en el estudio de los volcanes.

Figura 2-4. Robot zoomórfico

- Híbridos: Estos robots corresponden a aquellos de difícil clasificación, cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas, bien sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo segmentado articulado y con ruedas es, al mismo tiempo, uno de los atributos de los robots móviles y de los robots zoomórficos.

2.1.1.1 Grado de libertad y espacio de trabajo

Mecánicamente, un robot está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos. El movimiento de cada articulación puede ser de desplazamiento, de giro, o una combinación de ambos.

Cada uno de los movimientos independientes (giros y desplazamientos) que puede realizar cada articulación con respecto a la anterior, se denomina grado de libertad. El número de grados de libertad del robot viene dado

por la suma de los grados de libertad de las articulaciones que lo componen.

Los movimientos posibles para las articulaciones son: un desplazamiento (articulación de tipo prismático), un giro (articulación de rotación o revolución), o una combinación de ambos, siendo éstos últimos menos habituales. Las dos primeras son las más usadas prácticamente.

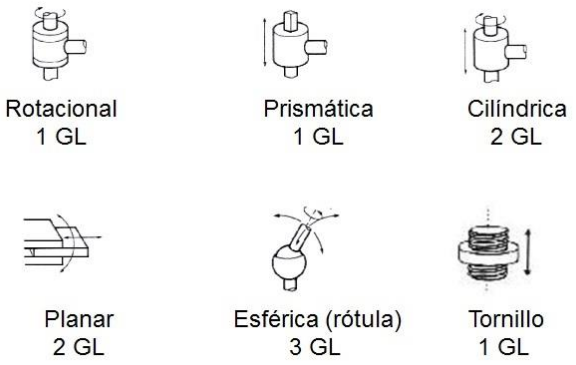


Figura 2-2-3. Tipos de articulaciones y sus GDL

El espacio de trabajo de un robot está definido como el grupo de puntos que pueden ser alcanzados por su efector-final. El conocimiento exacto sobre la forma, dimensiones y estructura de su espacio de trabajo es esencial puesto que la forma es importante para la definición del entorno donde el robot trabajará; las dimensiones son importantes para la determinación del alcance del efector-final y, la estructura del espacio de trabajo es importante para asegurar las características cinemáticas del robot las cuales están relacionadas con la interacción entre el robot y el entorno.

Además, la forma, dimensiones y estructura del espacio de trabajo dependen de las propiedades del robot en cuestión:

- Las dimensiones de los eslabones del robot y las limitaciones mecánicas de las articulaciones.
- La forma depende de la estructura geométrica del robot (interferencia entre eslabones) y también de las propiedades de los grados de libertad.
- La estructura del espacio de trabajo viene definida por la estructura del robot y las dimensiones de sus eslabones.

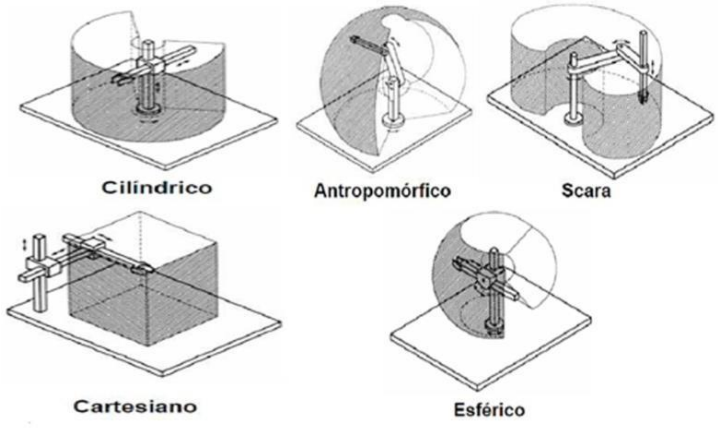


Figura 2-2-4. Diferentes espacios de trabajo

2.2 Cinemática

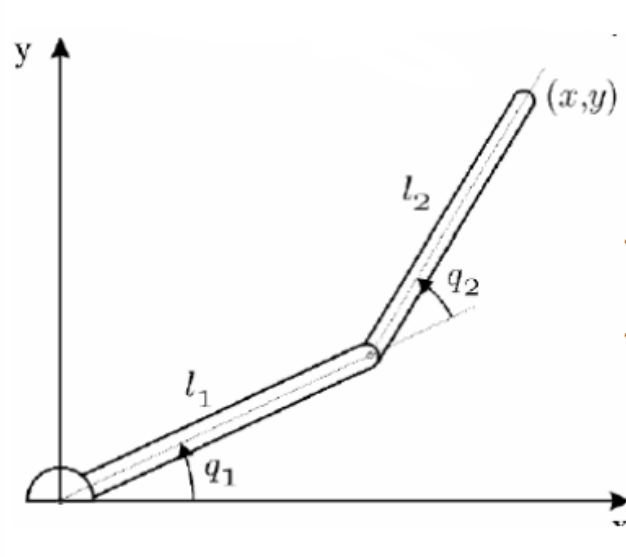
2.2.1 Modelo Cinemático

La cinemática de un robot es el estudio de los movimientos de un robot. En un análisis cinemático, la posición, velocidad y aceleración de cada uno de los elementos del robot son calculadas sin considerar las fuerzas que causan el movimiento. Hay dos formas de conseguir la cinemática de un robot: la directa y la inversa.

La cinemática directa es el problema geométrico estático de calcular la posición y orientación del efector final del manipulador. Se usa para calcular la posición de partes de una estructura articulada a partir de sus componentes fijas y las transformaciones inducidas por las articulaciones de la estructura. El proceso inverso que calcula el conjunto de parámetros a partir de una posición específica del actuador final es la cinemática inversa. A continuación, explicaremos las distintas formas para obtener el modelo cinemático directo.

2.2.1.1 Método geométrico

El método geométrico se basa en la obtención de la posición y orientación basándonos en las relaciones geométricas. No es un método sistemático y es usado cuando tenemos pocos grados de libertad.



$$x = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)$$
$$y = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)$$

Figura 2-2-5. Calculo del modelo cinemático directo mediante el método geométrico

2.2.1.2 Matrices de transformación homogénea

Otro método son las matrices de transformación homogéneas. Se calculan las translaciones y rotaciones relativas entre cada uno de los eslabones para obtener la relación entre el eslabón base y la posición final mediante el producto del resto de transformadas. Para ello lo que haremos será asociarle un sistema de referencia solidario a cada eslabón. Existen métodos sistemáticos para situar los sistemas de coordenadas asociadas a cada eslabón y obtener la cadena cinemática del robot. El método sistemático elegido en este proyecto es el explicado en el siguiente apartado.

2.2.1.3 Método Denavit-Hartenberg

Para calcular la cinemática directa utilizaremos un método conocido como parametrización Denavit-Hartenberg. Mediante el siguiente algoritmo, podemos sacar la cinemática directa del robot a partir de una parametrización. Para el caso de nuestro robot, se verá como se resuelve en el apartado 3.2.1.

1. Numerar los eslabones: se llamará «0» a la «tierra», o base fija donde se ancla el robot. «1» el primer eslabón móvil, etc.
2. Numerar las articulaciones: La «1» será el primer grado de libertad, y «n» el último.
3. Localizar el eje de cada articulación: Para pares de revolución, será el eje de giro. Para prismáticos

será el eje a lo largo del cual se mueve el eslabón.

4. Ejes Z: Empezamos a colocar los sistemas XYZ. Situamos los Z_{i-1} en los ejes de las articulaciones i, con $i = 1, \dots, n$. Es decir, Z_0 va sobre el eje de la 1ª articulación, Z_1 va sobre el eje del 2º grado de libertad, etc.
5. Sistema de coordenadas 0: Se sitúa el punto origen en cualquier punto a lo largo de Z_0. La orientación de X_0 e Y_0 puede ser arbitraria, siempre que se respete, evidentemente, que XYZ sea un sistema dextrógiro.
6. Resto de sistemas: Para el resto de sistemas $i = 1, \dots, N-1$, colocar el punto origen en la intersección de Z_i con la normal común a Z_i y Z_{i+1}. En caso de cortarse los dos ejes Z, colocarlo en ese punto de corte. En caso de ser paralelos, colocarlo en algún punto de la articulación $i+1$.
7. Ejes X: Cada X_i va en la dirección de la normal común a Z_{i-1} y Z_i, en la dirección de Z_{i-1} hacia Z_i.
8. Ejes Y: Una vez situados los ejes Z y X, los Y tienen sus direcciones determinadas por la restricción de formar un XYZ dextrógiro.
9. Sistema del extremo del robot: El n-ésimo sistema XYZ se coloca en el extremo del robot (herramienta), con su eje Z paralelo a Z_{n-1} y X e Y en cualquier dirección válida.
10. Ángulos teta: Cada θ_i es el ángulo desde X_{i-1} hasta X_i girando alrededor de Z_i.
11. Distancias d: Cada d_i es la distancia desde el sistema XYZ$_{i-1}$ hasta la intersección de las normales común de Z_{i-1} hacia Z_i, a lo largo de Z_{i-1}.
12. Distancias a: Cada a_i es la longitud de dicha normal común.
13. Ángulos alfa: Ángulo que hay que rotar Z_{i-1} para llegar a Z_i, rotando alrededor de X_i.
14. Matrices individuales: Cada eslabón define una matriz de transformación:

$$ {}^{i-1}A_i = \begin{pmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2\text{-}1) $$

15. Transformación total: La matriz de transformación total que relaciona la base del robot con su herramienta es la encadenación (multiplicación) de todas esas matrices:

$$ T = {}^0A_1\,{}^1A_2 \cdots {}^{n-1}A_n \quad (2\text{-}2) $$

2.2.1.4 Cinemática inversa

Una vez obtenido el modelo directo, podremos calcular el inverso que nos dará el movimiento de cada articulación para que el robot se coloque en la posición en cartesianas que le hemos pedido. Sin embargo, no existe siempre una solución cerrada. Puede ocurrir que el movimiento hacia un punto tengas más de una solución posible, debido al ángulo que puede tomar alguna articulación como se puede observar en la imagen siguiente.

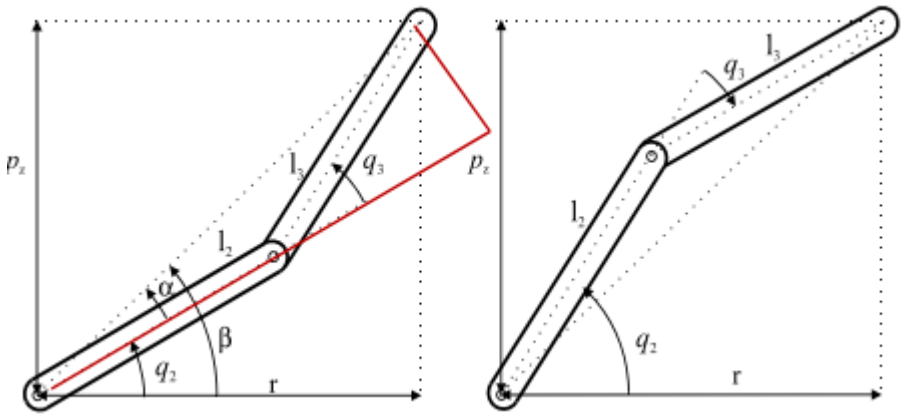


Figura 2-2-6. Soluciones posibles al posicionarse el robot

Se puede apreciar que la articulación 2 tiene dos posibles soluciones. Es importante tenerlo en cuenta a la hora de calcular la cinemática inversa para decidir qué solución queremos que coja el robot.

Para calcular la cinemática inversa hay varios métodos. Al igual que en la cinemática directa, podemos resolverlo mediante el método geométrico, que es el que usaremos en el apartado 3.2.2. Se suelen utilizar para las primeras variables articulares. Otro método, es la resolución a partir de las matrices de transformación homogéneas. También podemos calcularlo mediante el desacoplamiento cinemático que se suele usar en robots de 6 GDL. Además de estos métodos, hay otros como el álgebra de tornillo, cuaterniones duales, métodos iterativos, etc.

2.2.2 Control cinemático

Un generador de trayectoria es un modo de controlar el robot. Se debe tener en cuenta el punto de destino, el tipo de trayectoria del extremo, la posición inicial y el tiempo de movimiento. Para ello es necesario conocer el modelo cinemático del robot. Es importante, también, atender a las restricciones físicas de los accionamientos y criterios de calidad (suavidad, precisión, …).

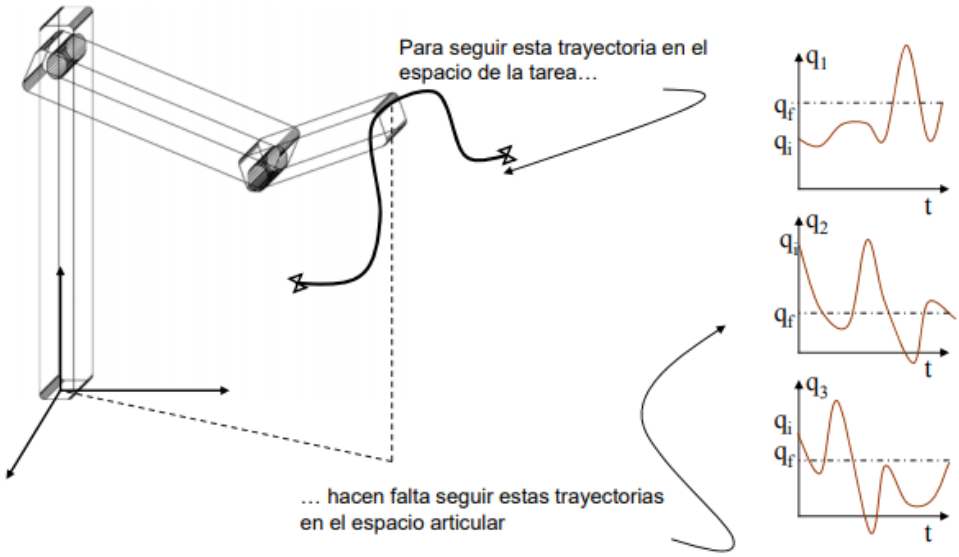


Figura 2-2-7. Valores de las articulaciones ante el seguimiento de una trayectoria

Debido a las limitaciones de los accionamientos, no es posible seguir cualquier trayectoria articular. A la hora de programar el control, en este punto, hay que tener en cuenta lo descrito en el apartado anterior sobre la cinemática inversa. Al haber limitaciones, debemos elegir el movimiento que se ajuste a ellas.

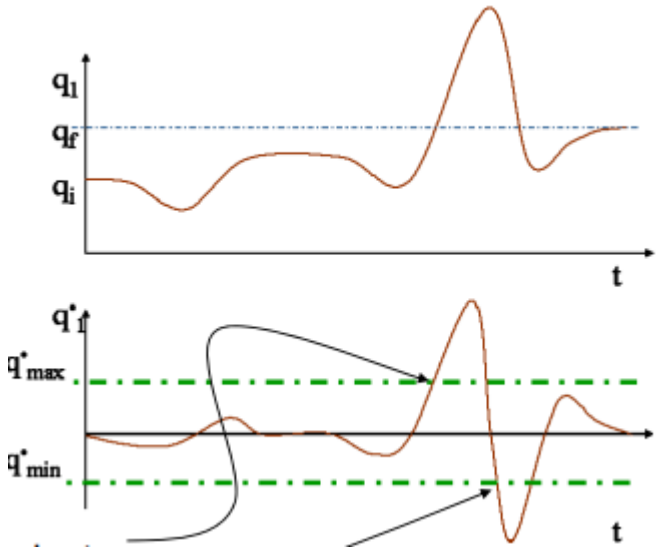


Figura 2-2-8. Movimiento de una articulación con restricciones.

Para generar trayectorias, primero calculamos las ecuaciones de trayectoria en cartesianos a partir del orden de movimiento. Seguidamente, obtenemos los puntos concretos muestreando en el tiempo la trayectoria en cartesianas. Convertimos los puntos en cartesianos a espacio articular con modelo cinemático inverso. Por último, generamos la trayectoria (posición, velocidad y aceleración) en espacio articular mediante interpolación de coordenadas articulares en el tiempo.

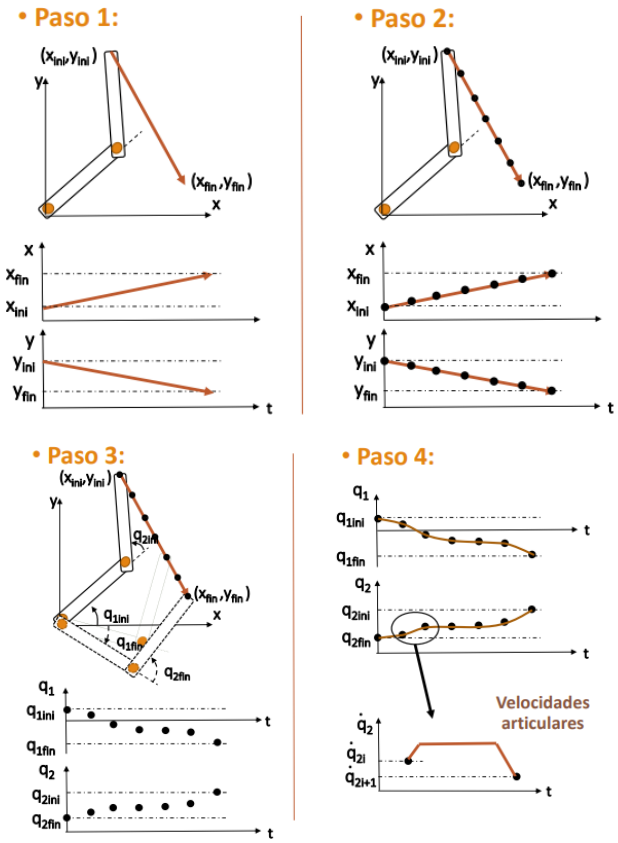


Figura 2-9. Pasos para el seguimiento de trayectoria

2.2.2.1 Tipos de trayectorias

Hay distintos tipos de trayectorias, así como distintos tipos de control. Esta las trayectorias punto a punto, en donde se puede programar de distintas maneras, mediante el movimiento eje a eje, movimiento simultaneo de ejes o las trayectorias coordinadas o isócronas. El otro tipo de trayectoria es la continua. En cuanto al control, podemos controlarlo mediante un control de posición a posición o un control de trayectorias continuas.

En la trayectoria punto a punto no importa el camino del extremo del robot. Solo importa que alcance el punto final indicado. En el movimiento eje a eje, solo se mueve un eje cada vez, con lo que aumenta el tiempo de ciclo (Solo en robots muy simples o con unidad de control limitada). En el movimiento simultaneo de ejes, los ejes comienzan a la vez y cada uno acaba cuando puede. Por último, tenemos el movimiento coordinado donde los ejes comienzan y acaban a la vez, no importa el camino del extremo del robot, pero los ejes se mueven simultáneamente, ralentizando las articulaciones más rápidas, de forma que todos los ejes acaben a la vez. El tiempo total debe ser el menor posible y se evitan exigencias inútiles de velocidad y aceleración.

En las trayectorias continuas se pretende que el extremo del robot describa una trayectoria concreta y conocida. Importe el camino (soldadura con arco, aplicación de sellante, lijado de piezas, …). Las trayectorias típicas suelen ser rectas o arcos de circunferencia.

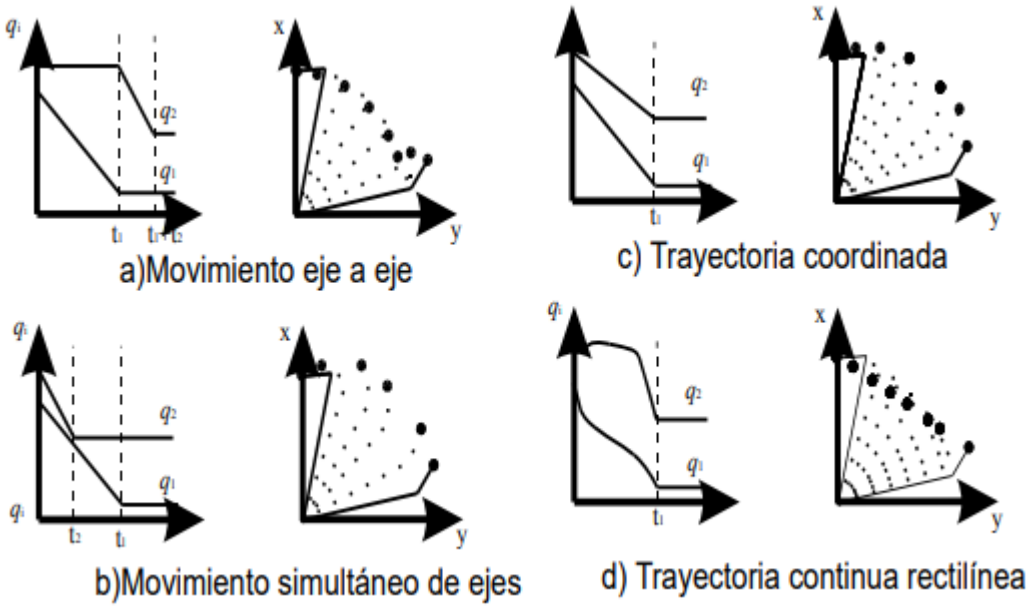


Figura 2-10. Tipos de trayectorias

2.2.3 Generación de trayectorias

Para determinar la posición que debe tomar, la podemos hacer de punto a punto (p.e., línea recta) o por varios puntos, unidos por rectas u otros interpoladores.

$$j(t) = \left(j_f - j_{in}\right)\frac{t - t_{in}}{t_f - t_{in}} + j_{in} \qquad (2\text{-}3)$$

En cuanto a la orientación, la calcularemos mediante interpolaciones. L as interpolaciones no se pueden hacer con elementos de MTH. Sí se puede interpolar en ángulos de Euler y también se puede interpolar el par de rotación y el de los cuaternios.

$$\phi(t) = (\phi_f - \phi_{in})\frac{t - t_{in}}{t_f - t_{in}} + \phi_{in} \tag{2-4}$$

$$\theta(t) = (\theta_f - \theta_{in})\frac{t - t_i}{t_f - t_{in}} + \theta_{in} \tag{1-5}$$

$$\psi(t) = (\psi_f - \psi_{in})\frac{t - t_i}{t_f - t_{in}} + \psi_{in} \tag{2-6}$$

Una vez obtenidas las trayectorias las muestreamos. No es posible una transformación analítica desde la trayectoria cartesiana a la articular ($j(t) \rightarrow q(t)$). Una alternativa es la conversión (mediante MCI) de algunos puntos de $j(t)$ →Muestreo. Con esto, cabe preguntarse ahora cuantos puntos tomar para muestrear la trayectoria cartesiana. Si tomamos muchos puntos la precisión será muy alta, pero precisará de transformada inversa para cada uno de ellos. Si cogemos pocos puntos simplemente tendremos poca precisión.

2.2.4 Interpolación de trayectorias

Es la unión de una sucesión de puntos en el espacio articular, por los que han de pasar las articulaciones del robot en un instante determinado. Es necesario que respete algunas restricciones como la velocidad y el par máximo de los actuadores. Hay varios tipos de interpoladores que se utilizan.

2.2.4.1 Interpoladores lineales

Unión de sucesión de puntos articulares q_i por los que se debe pasar en instantes t_i mediante líneas rectas. Es una interpolación simple, pero se requerirían aceleraciones infinitas en los puntos de paso.

$$q(t) = (q_p - q_{p-1})\frac{t - t_{p-1}}{T} + q_{(p-1)} \qquad t_{p-1} < t < t_p \tag{2-7}$$

$$T = t_p - t_{p-1}$$

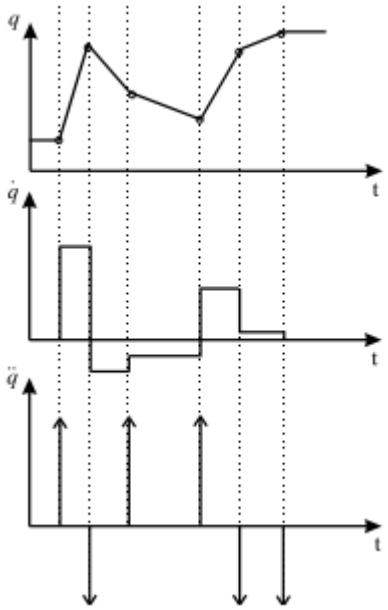


Figura 2-11. Valores de posición, velocidad y aceleración de una articulación.

2.2.4.2 Interpoladores polinómicos

Para unir n puntos (t_p, q_p) se puede utilizar un polinomio de grado n-1. En la práctica esto conduce a polinomios en t de grado (n-1), originándose problemas computacionales. Como alternativa se recurre a polinomios de grado bajo (3 a 5) que unen unos pocos puntos consecutivos y a los que se impone adicionalmente la continuidad en las primeras derivadas (posición, velocidad, etc.). Los interpoladores lineales, antes planteados, son el caso particular n=2.

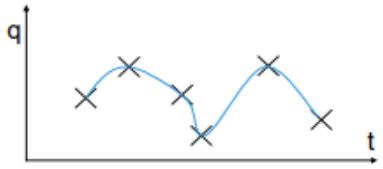


Figura 2-12. Ejemplo interpolador lineal.

2.2.4.3 Interpoladores cúbicos

Se une cada pareja de puntos con un polinomio de grado 3 (4 parámetros) para cada tramo:

$$q(t) = a + bt + ct^2 + dt^3 \tag{2-8}$$

Se precisa de 4 condiciones de contorno: posiciones y velocidades iniciales y finales de cada tramo. La trayectoria está formada por una serie de polinomios cúbicos concatenados escogidos de forma que exista continuidad en posición y velocidad, denominados splines.

$$q(t) = a + b(t - t_p) + c(t - t_p)^2 + d(t - t_p)^3 \qquad t_p < t < t_{p+1} \tag{2-9}$$

$$a = q_p \tag{2-10}$$

$$b = \dot{q}_p \tag{2-11}$$

$$c = \frac{3}{T^2}(q_{p+1} - q_p) - \frac{1}{T}(\dot{q}_{p+1} + 2\dot{q}_p) \tag{2-12}$$

$$d = -\frac{2}{T^3}(q_{p+1} - q_p) + \frac{1}{T^2}(\dot{q}_{p+1} + \dot{q}_p) \tag{2-13}$$

$$T = t_{p+1} - t_p \tag{2-14}$$

A la hora de calcular las velocidades hay distintas alternativas. Podemos utilizar el método heurístico dándoles el valor 0 o valor medio de las velocidades lineales:

$$q'_p = \begin{cases} 0 & \text{si signo}\,(q_p - q_{p-1}) \neq \text{signo}(q_{p+1} - q_p) \\ \frac{1}{2}\left[\frac{q_{p+1} - q_p}{t_{p+1} + t_p} + \frac{q_p - q_{p-1}}{t_p - t_{p-1}}\right] & \text{si} \begin{cases} \text{signo}\,(q_p - q_{p-1}) = \text{signo}\,(q_{p+1} - q_p) \\ \text{o} \quad q_{p-1} = q_p \\ \text{o} \quad q_p = q_{p+1} \end{cases} \end{cases} \tag{2-15}$$

Otra alternativa es usar la jacobiana inversa a partir de las velocidades en el espacio de la tarea. La última, es obligando a tener continuidad en las aceleraciones, resolviendo un sistema de ecuaciones con todas las velocidades de paso de manera conjunta. Este generador es el que usaremos en nuestro robot.

2.2.4.4 Interpoladores quinticos

Se une cada pareja de puntos con un polinomio de grado 5 (6 parámetros). 6 condiciones de contorno: posiciones, velocidades y aceleraciones iniciales y finales de cada tramo. La trayectoria está formada por una serie de polinomios quinticos concatenados escogidos de forma que exista continuidad en posición, velocidad y aceleración, denominados splines.

$$q(t) = a + b(t - t_p) + c(t - t_p)^2 + d(t - t_p)^3 + e(t - t_{i-1})^4 + f(t - t_{i-1})^5 \qquad t_p < t < t_{p+1} \tag{2-16}$$

2.2.4.5 Interpoladores trapezoidales

Evitan que la velocidad varíe durante la mayor parte de la trayectoria (solo en los cambios de dirección). Utiliza un interpolador lineal (velocidad constante) durante todo el trayecto salvo en las cercanías de los cambios de dirección, donde usa un interpolador de segundo grado.

$$\text{Tramos rectos: } \begin{cases} q(t) = \dot{q}_p(t - t_{p-1}) + q_{p-1} & t_{p-1} + \tau_{p-1} < t < t_{p-1} - \tau_p \\ \text{con } \dot{q}_p = \frac{q_p - q_{p-1}}{t_p - t_{p-1}} & \tau_p = \frac{\dot{q}_{p+1} - \dot{q}_p}{2a} \end{cases} \tag{2-17}$$

$$\text{Tramos parabólicos: } \begin{cases} q(t) = \frac{1}{2}a(t - t_p)^2 + \frac{\dot{q}_{p+1} + \dot{q}_p}{2}t + C & t_{p-1} + \tau_{p-1} < t < t_{i-1} - \tau_p \\ \text{con } C = -\frac{1}{2}a\tau_p^2 - \frac{q_{p+1} + \dot{q}_p}{2}(t_p - \tau_p) + \dot{q}_p(t_p - \tau_p - t_{p-1}) + q_{p-1} \end{cases} \tag{2-18}$$

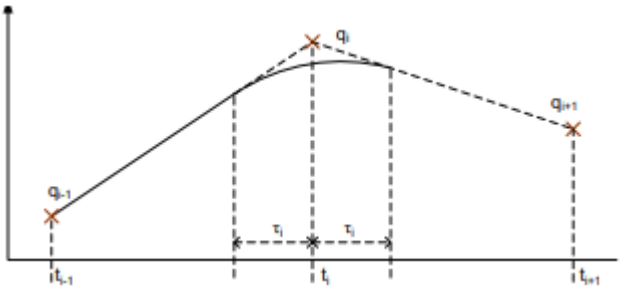


Figura 2-13. Ejemplo interpolador trapezoidal

2.3 Materiales

2.3.1 Mecánica

Se ha optado en usar la madera a la hora de construir el robot ya que se trata de un material asequible, de fácil uso y, además, económico. En cuanto a los motores se han elegido servos, debido a los mismos motivos que la madera, así como su fácil programación. La plantilla que se ha usado para el robot, la cual se utilizó para cortar la madera, es la siguiente:

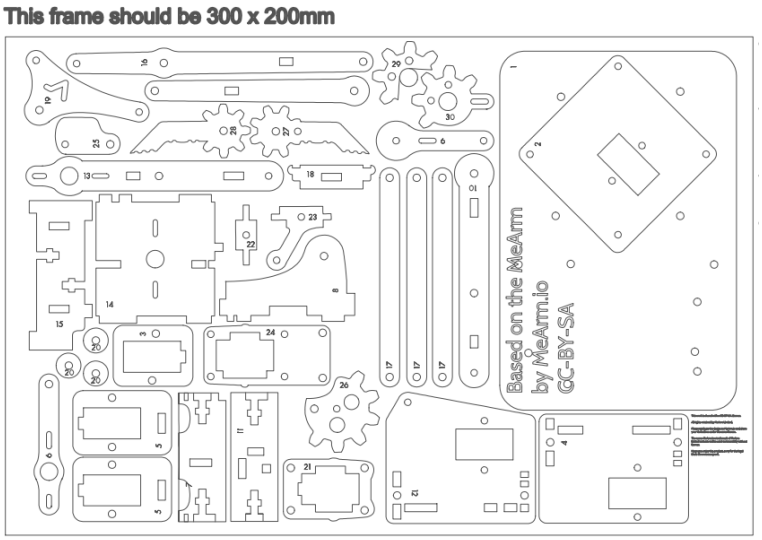


Figura 2-14. Plantilla de las piezas del robot.

2.3.2 Electrónica

2.3.2.1 Servomotores

Un servomotor es un dispositivo pequeño que tiene un eje de rendimiento controlado. Este puede ser llevado a posiciones angulares específicas al enviar una señal codificada. Con tal de que una señal codificada exista en la línea de entrada, el servo mantendrá la posición angular del engranaje. Cuando la señal codificada cambia, la posición angular de los piñones cambia.

Un servomotor está compuesto básicamente por un motor que tiene un potenciómetro conectado a su eje, todo ello gobernado por un circuito de control. El eje del motor se encuentra conectado a un tren de engranajes que reduce la velocidad del motor y la convierte en fuerza en el eje de salida. El servo puede tener un control de posición, velocidad o aceleración. El más común y el que se usará en este proyecto será el de control de posición que lo deja estable en un punto.

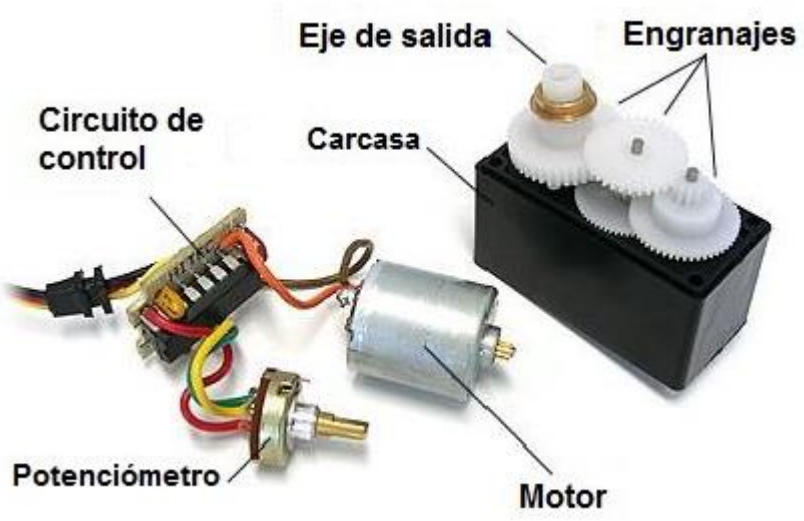


Figura 2-15. Elementos de un servomotor

La señal de entrada que recibe el circuito de control se trata de un PWM (Pulse Width Modulation). Dependiendo del ancho del pulso que se recibe en cada cierto periodo, el control sabrá si debe mantenerse en cierto ángulo u otro.

El funcionamiento de un servomotor es el siguiente. El potenciómetro contenido dentro de la carcasa del servomotor está sujeto al eje de salida, y mide en todo momento la posición en la que se encuentra, de modo que sirve de referencia de la posición actual al circuito de control, que será quien le indique cuánto girar al motor.

2.3.2.2 Joystick

Es un periférico de entrada que consiste en una palanca que gira sobre una base e informa su ángulo o dirección al dispositivo que está controlando. Los joysticks generalmente tienen uno o más botones cuyo estado también se puede leer.

En los joysticks más habituales como los joysticks de los mandos y lo que usaremos en el proyecto, se usan potenciómetros. Moviendo el contacto móvil por la pista, puedes aumentar o disminuir la resistencia de la electricidad fluyendo por el circuito. Al hacer movimientos, variamos en todos los sentidos los contactos alterando la electricidad que pasa por los circuitos, y portando comunicando al ordenador que debe hacer en cada momento. La señal eléctrica es totalmente analógica. Para hacer que la información sea útil, el ordenador debe cambiarla en una señal digital.

2.3.2.3 Microcontrolador

El microcontrolador elegido para este proyecto es el Arduino UNO que nos proporciona 14 pines de entrada y salida de señales digitales de 0-5 V, cabe destacar que cada pin del Arduino tiene una resistencia de 20KΩ o 50KΩ pull-up interna que podemos activar si nos hiciera falta, por ejemplo, para los botones. Cuenta también con 6 pines analógicos que trasladan las señales a un convertidor analógico/digital de 10 bits.

Otras características a tener en cuenta, es que los pines digitales 2 y 3 están configurados para que generen señales de interrupción. Dispone además de 6 salidas destinadas a la generación de señales PWM de hasta 8 bits. Los pines 10, 11, 12 y 13 se pueden utilizar para llevar a cabo comunicaciones SPI.

Es importante añadir que, a la hora de alimentar la placa, esta debe de estar alimentada por un voltaje de entre 6 y hasta 12 voltios. Si se supera esta tensión, la placa puede acabar dañada. Además, si se alimenta con poca tensión, la salida del regulador de tensión de 5V puede dar menos, algo importante si queremos alimentar varios dispositivos a la vez.

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines para entrada- salida digital.	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica.	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

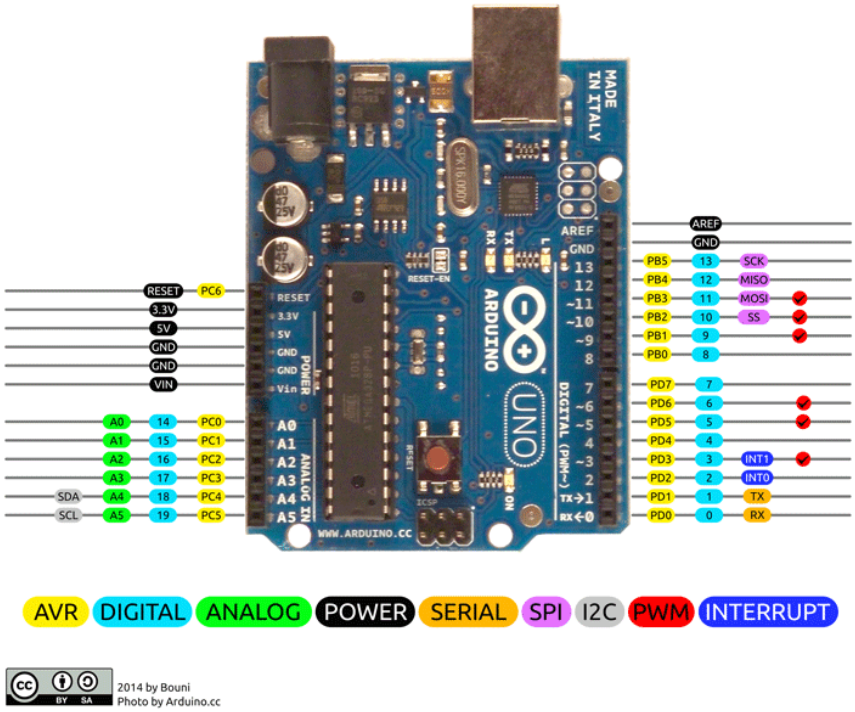


Figura 2-16. Arduino Genuino uno

3 MECÁNICA

3.1 Construcción

Mecánicamente, un robot está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo cada dos eslabones consecutivos. El robot elegido es un robot manipulador de 3 grados de libertad, dada su simplicidad, con articulaciones rotacionales, puesto que son las que más fácilmente se obtienen a partir de los actuadores seleccionados para este proyecto. Al ser similar a un brazo humano, podemos hacer referencias a los elementos empleando términos como cuerpo, brazo, codo y muñeca. Como se trata de un robot de 3 GDL, vamos a prescindir de la muñeca.

Al hablar de la estructura del manipulador, es significativamente importante comentar el número de módulos que afecta de manera directa al alcance del robot (zona de operación del manipulador), así como a las diferentes posiciones del espacio que puede alcanzar. El sistema propuesto por el autor consiste en el siguiente número de módulos:

- Módulo base: Rotación en torno al eje Z.
- Módulo antebrazo: Rotación en torno al eje Y.
- Módulo brazo: Rotación en torno al eje Y.
- Elemento aprehensor: Pinza.

Para cortar las piezas se ha utilizados una cortadora láser y se han fijado para quitar los desperfectos que podrían tener. Se utilizarán tuercas y tornillos de 3mm, los cuales serán 7 de 6mm de largo, 15 de 8 mm, 5 de 10 mm, 8 de 12 mm y 4 de 20 mm. El número de tuercas será 11.

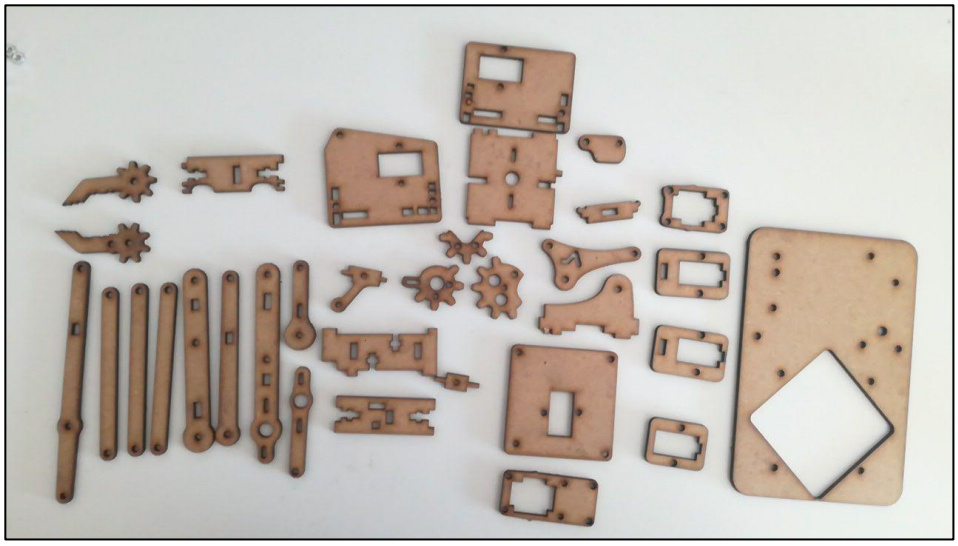


Figura 3-1.Piezas cortadas

Para poder ir explicando cómo se ha ido construyendo el robot vamos a enumerar las piezas para que sea más sencillo el identificarlas.

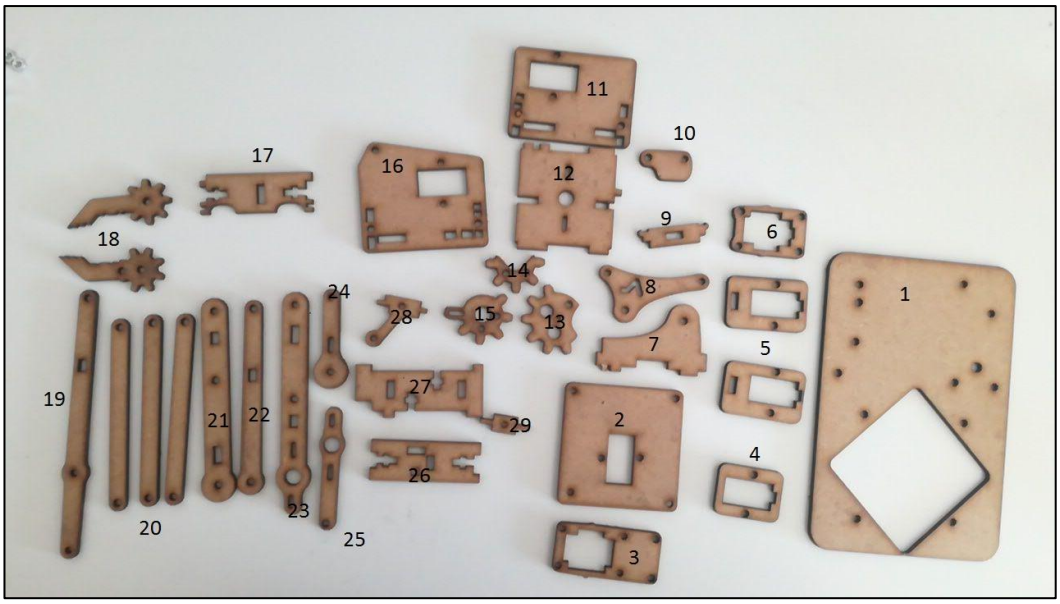


Figura 3-2. Piezas enumeradas

3.1.1 Base

Con los servos calibrados, comenzaremos por la base del robot. Vamos a usar las piezas 1, 2 y 4 junto a 4 tornillos de 20 mm, 2 de 8 mm y 4 tuercas.

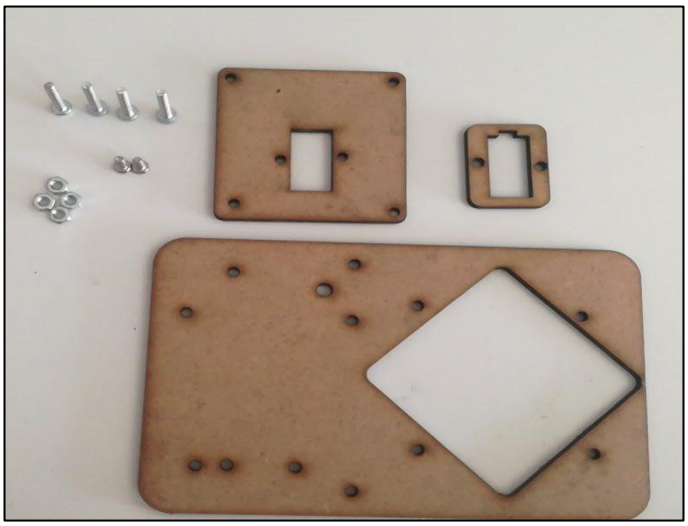


Figura 3-3. Elementos de la base

En la pieza 1 introducimos los 4 tornillos de 20 mm desde la parte baja y enroscamos las tuercas para que no se salgan los tornillos.

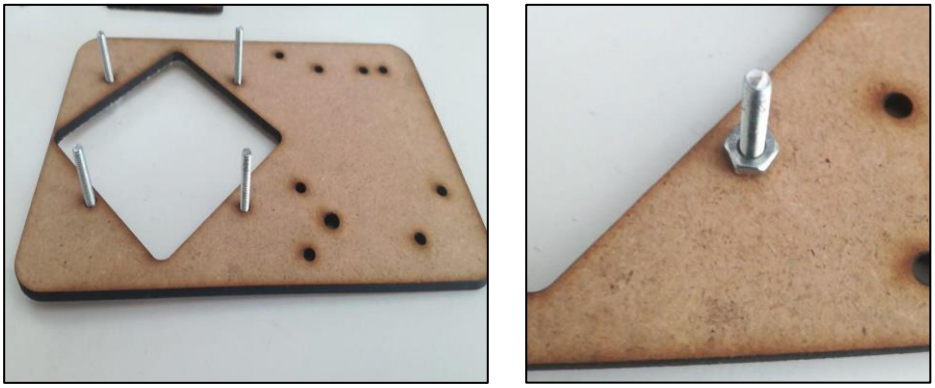


Figura 3-4. Soporte de la base

A continuación, cogemos el servo que usaremos para la base y las piezas 2 y 4. Introducimos el servo en la pieza 2 y pasamos la pieza 4 por el servo, una vez hecho atornillamos con los tornillos de 8 mm junto con la parte 2 dejando el servo fijo mediante la presión de las dos piezas.

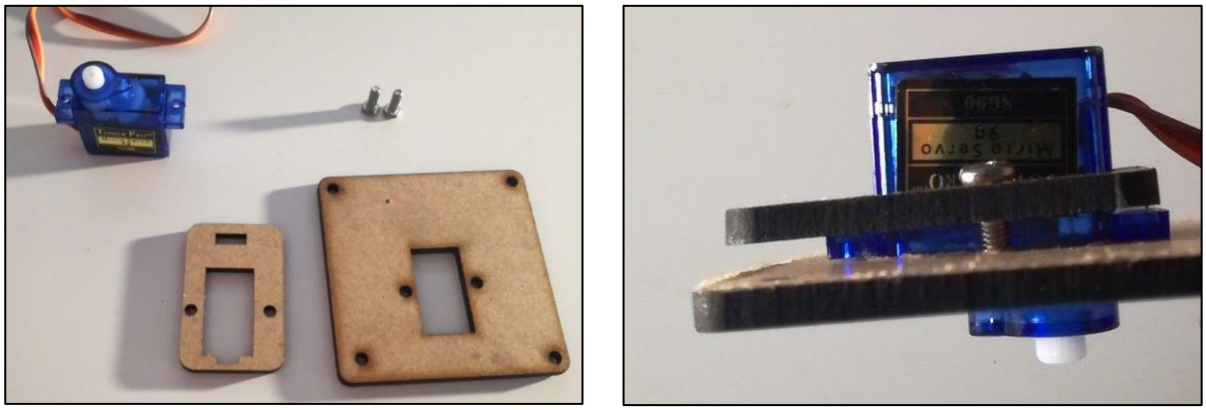


Figura 3-5. Articulación de la base.

Los tornillos se introducen con facilidad en la parte 4 ya que el diámetro del agujero es similar al del tornillo, en cambio en la pieza 2 son un poco más pequeños, pudiendo atornillarlos y dejarlos fijos. Por último, para acabar con la base, introducimos las esquinas de la parte 2 con el servo a la parte 1 a presión. Esta parte está diseñada para que no haya error al colocar las partes teniendo que ponerse si o si con la orientación correcta. Con esto la base estará terminada.

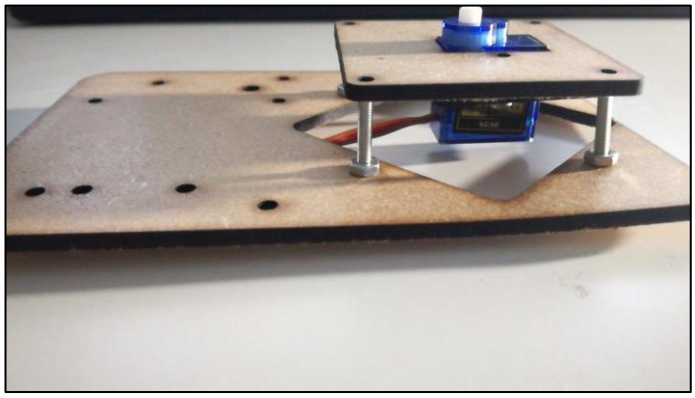


Figura 3-6. Base del robot.

3.1.2 Lado izquierdo

En esta parte utilizaremos las piezas 5,11 y 25; el servo que se usará para controlar el codo del robot junto con sus tornillos y piezas para fijarlo, y 2 tornillos de 8mm.

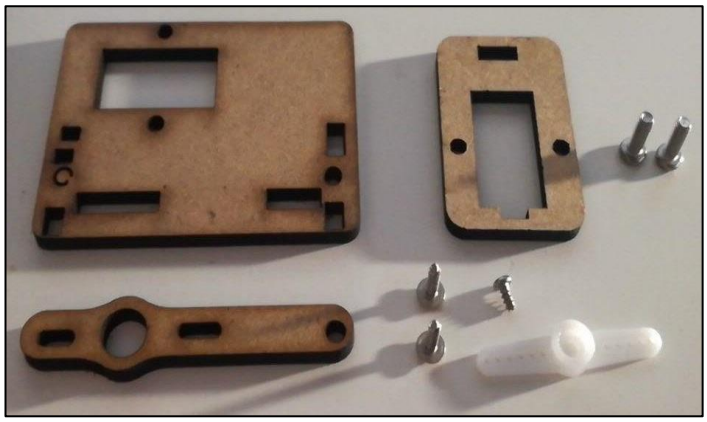


Figura 3-7. Elementos del lado izquierdo.

Introducimos al servo la parte 5 como hicimos en el apartado anterior y a su vez introducimos este por el hueco para el servo de la pieza 11 atornillándolo con los tornillos de 8 mm y dejando el servo fijo.

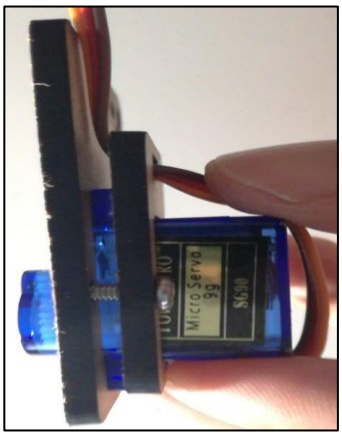


Figura 3-8. Servomotor para la articulación del codo

El siguiente paso será atornillar las piezas del servo a la parte 25 y seguidamente al servo, acabando con el lado izquierdo.

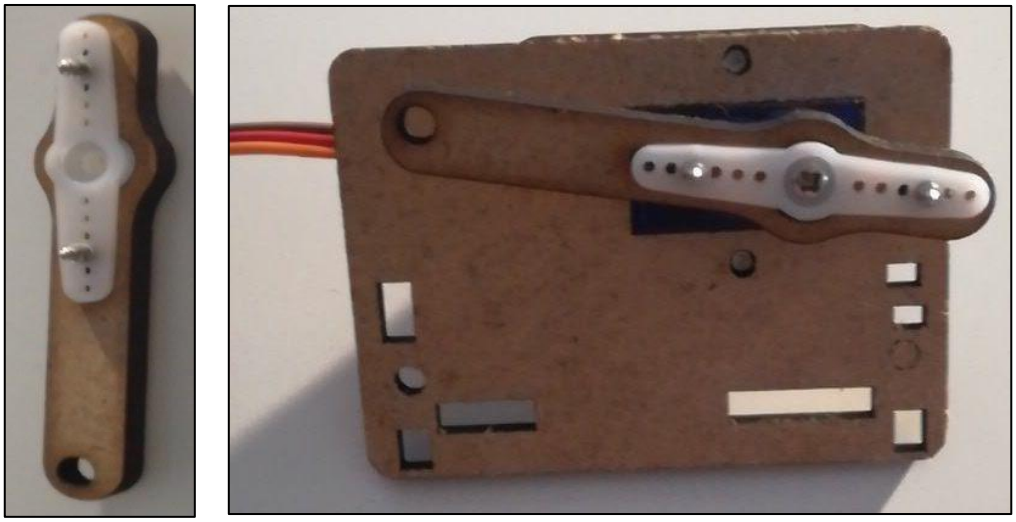


Figura 3-9. Primer eslabón del codo

3.1.3 Lado derecho

Para este apartado utilizaremos las piezas 16, 5, una del 20 y el 23, además del servo que usaremos para el hombro, y 2 tornillos de 8 mm y 1 de 6mm. Al igual que en los anteriores apartados colocamos el servo entre las piezas 5 y 16 y lo atornillamos con los tornillos de 8mm.

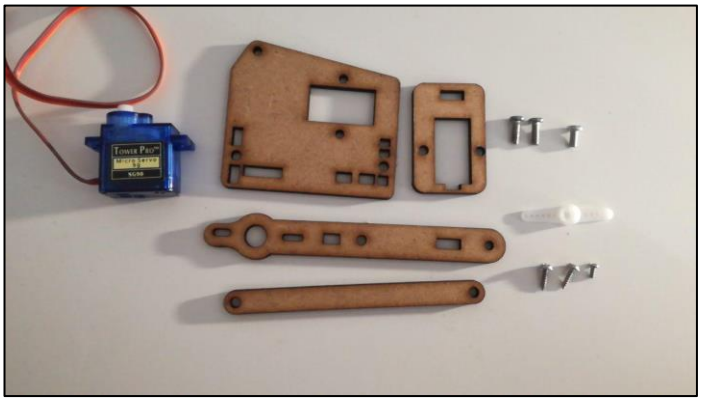


Figura 3-10. Elementos lado derecho

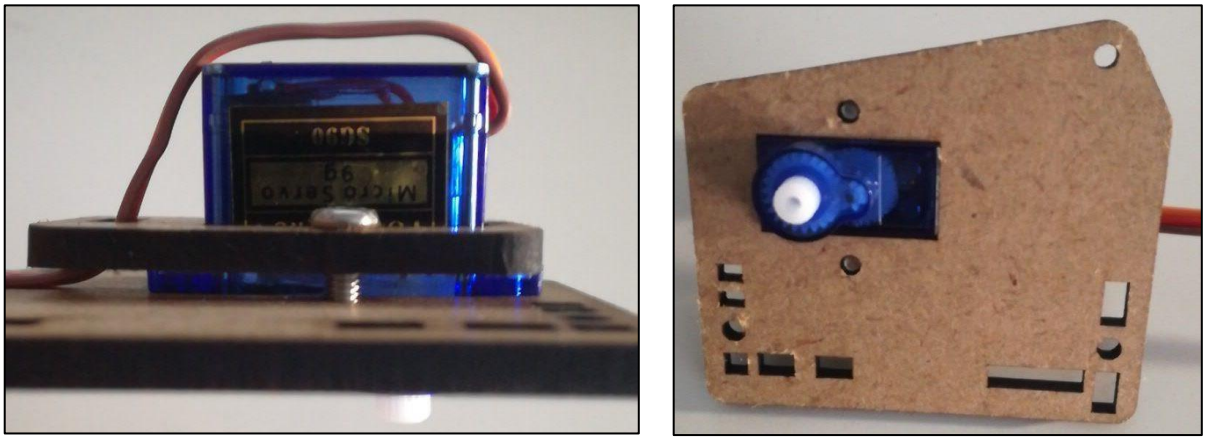


Figura 3-11. Servomotor para la articulación del hombro.

Atornillamos la pieza 23 al servo mediante el pack de fijado del servo como se hizo en el apartado anterior.

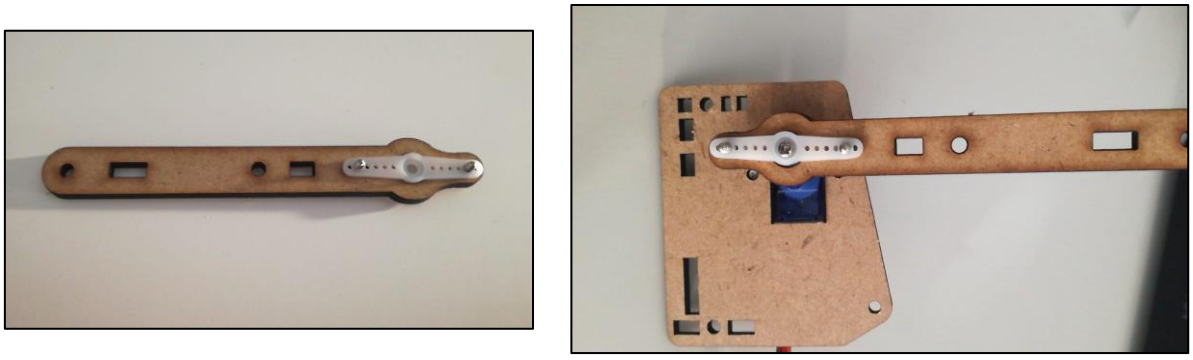


Figura 3-12. Primer eslabón del codo

Por último, atornillamos con el tornillo de 6 mm la pieza 20 en la esquina de la pieza 16.

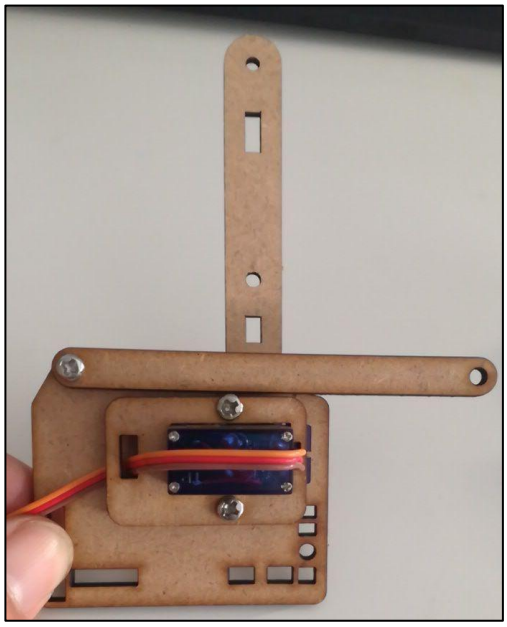


Figura 3-13. Segundo eslabón del hombro

3.1.4 Brazo

Las partes que usaremos a continuación serán la 12, 7, 21 y 24 junto con un tornillo de 10mm. Lo primero que haremos será montar la parte móvil de la base, para que el servo pueda mover el cuerpo del robot. Para ello, cogeremos la pieza 12 y le atornillamos las partes de fijado del servo para después atornillarlo al servo de la base.

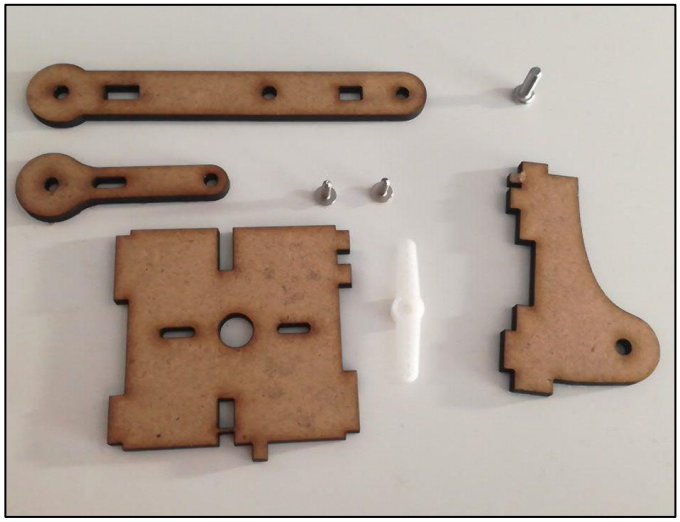


Figura 3-14. Elementos centrales del robot y soporte del brazo

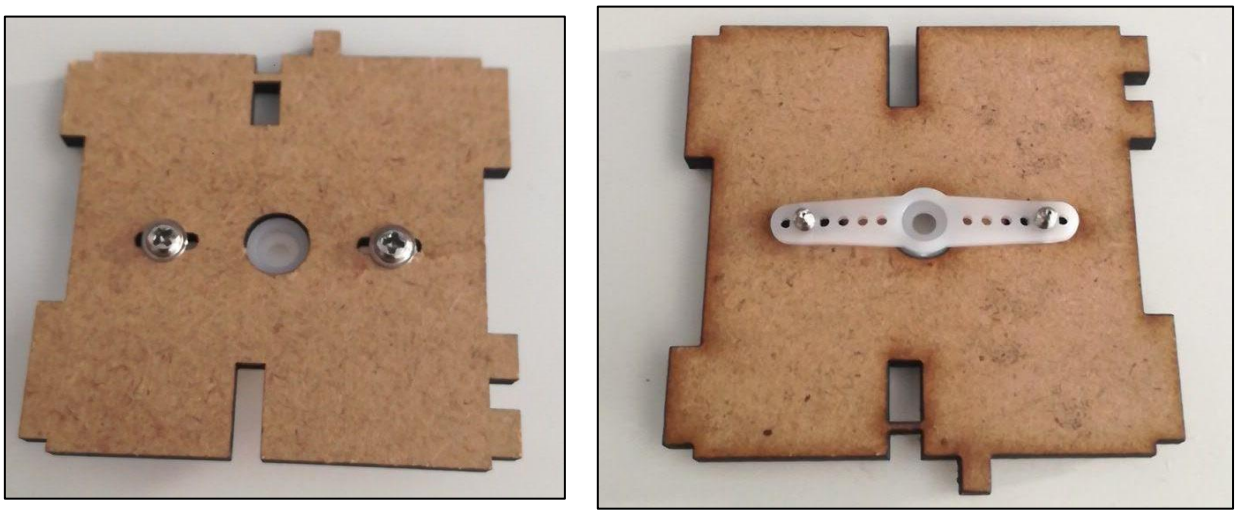


Figura 3-15. Eslabón de la articulación de la base

Ahora construiremos uno de los soportes para los brazos que nos ayudara más adelante a mover el brazo con los servos. Cogemos las piezas 7,21 y 24, y con el tornillo de 10 mm lo atornillamos poniendo el lado circular de las piezas 21 y 24.

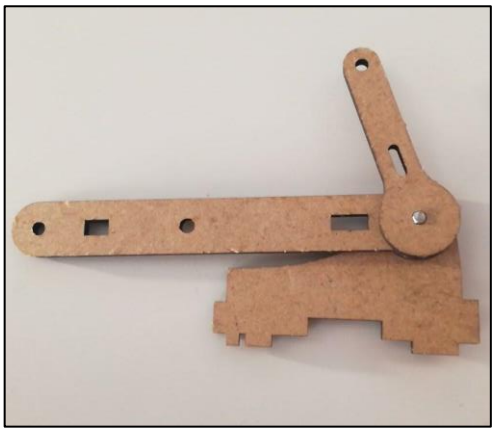


Figura 3-16. Soporte para el brazo

3.1.4.1 Codo

Con estas partes montadas y el lado izquierdo, junto con la pieza 17 y un tornillo de 12 mm con su tuerca, comenzaremos a montar el codo. Primero uniremos el conjunto montado de la parte izquierda al eslabón de la base y aprovecharemos para añadirle el soporte ya a la parte central.

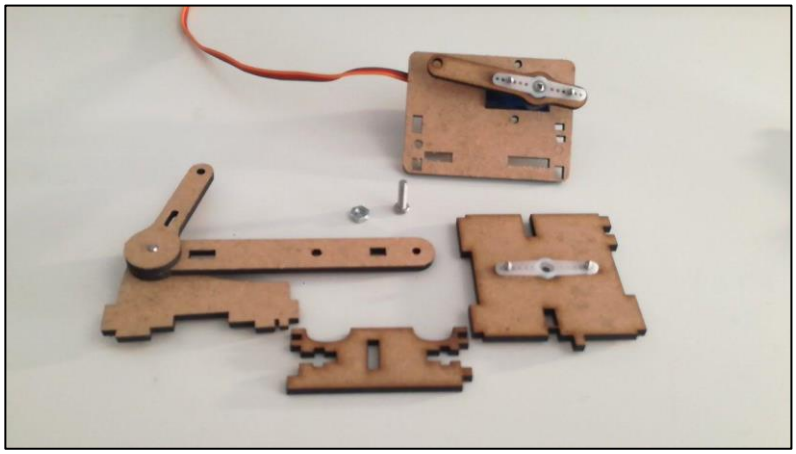


Figura 3-17. Elementos para unir la articulación del codo al eslabón de la base

Cogemos la pieza 17 y la encajamos en la parte izquierda montada, introduciéndolo por los agujeros que le corresponde. Una vez introducido insertamos la tuerza en la pieza 17 por la parte encajada y atornillamos para asegurarnos así de que el robot no se separa en ningún momento. Haciendo esto habremos unido la parte izquierda con la parte delantera de la base del robot.

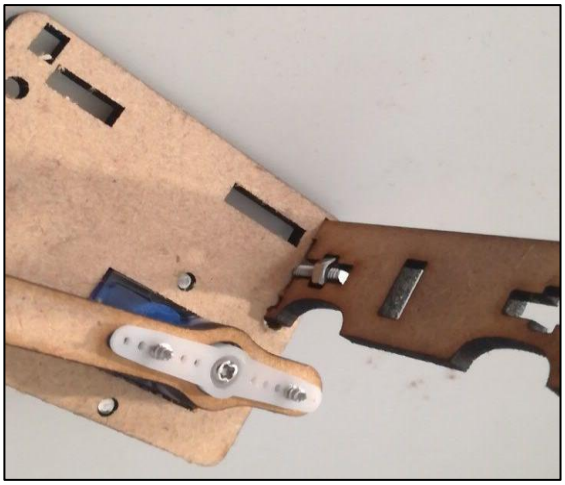


Figura 3-18. Unión de la parte delantera del robot al lado izquierdo

Con esto vamos a terminar de montar parte del codo juntándolo al eslabón de la articulación de la base acabando ya parte de la base del mismo y al soporte del brazo. Para ello, cogemos los dos conjuntos montados anteriormente y los encajamos de manera perpendicular por las rendijas que se encuentran en la pieza 12, ahora encajamos a la pieza 7 la pieza 17 juntando así el centro del brazo con el hombro.

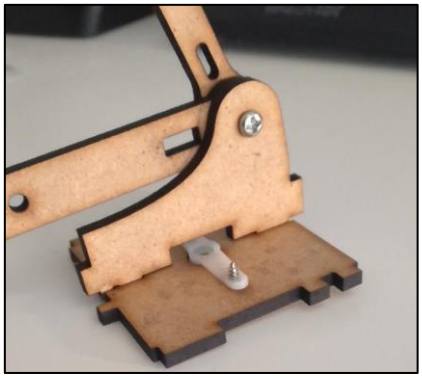


Figura 3-19.Unión del soporte con el eslabón de la base.

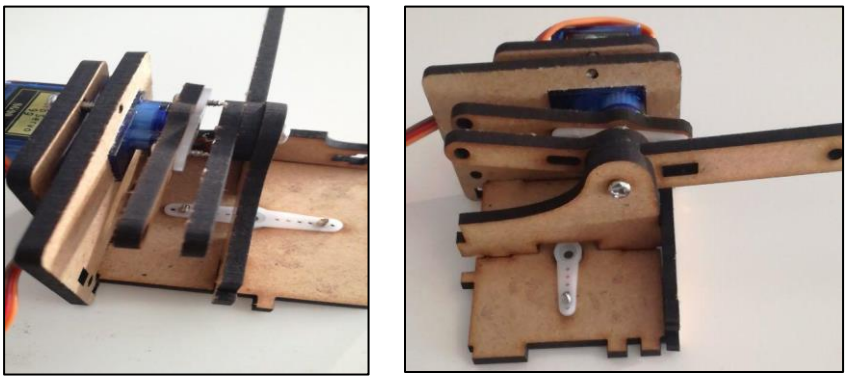


Figura 3-20. Unión del lado izquierdo y delantero con el soporte y el eslabón de la base.

Por último, unimos los eslabones del codo y el soporte central poniendo entre estas una pieza del 20 y uniéndolas mediante un tornillo de 12 mm.

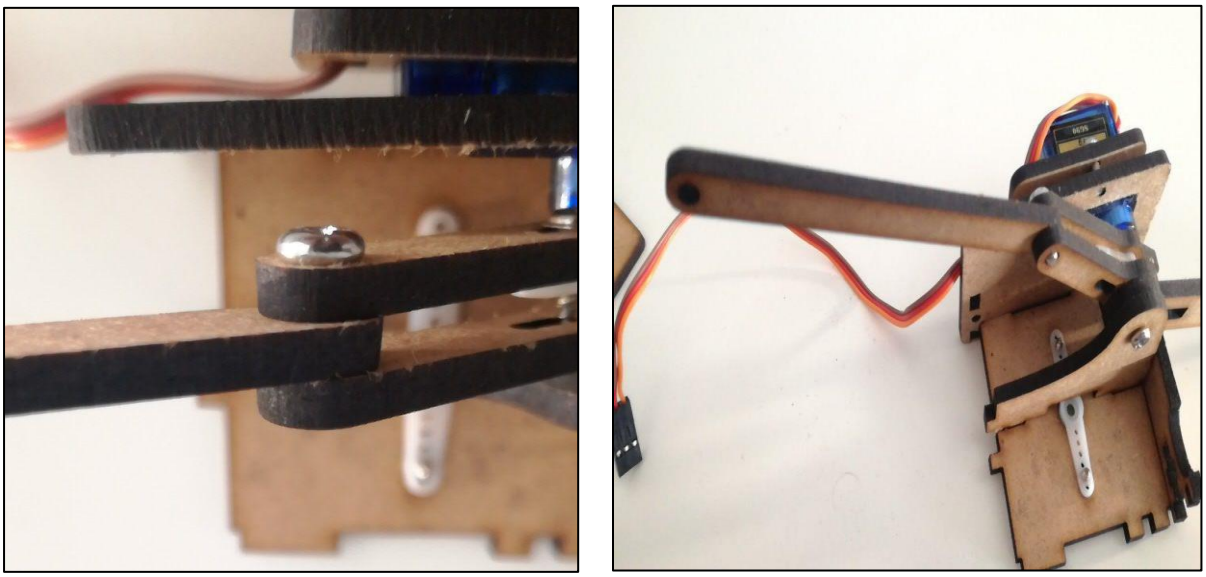


Figura 3-21. Unión de la articulación del codo con el soporte central.

3.1.4.2 Hombro

Con este conjunto y el conjunto del lado derecho, junto con la pieza 27 y 26, y usando 5 tornillos de 12 mm con sus tuercas, terminaremos de unir todas las partes del robot para, posteriormente, terminar el codo, unirlo al servo que moverá la base del robot y terminar el hombro.

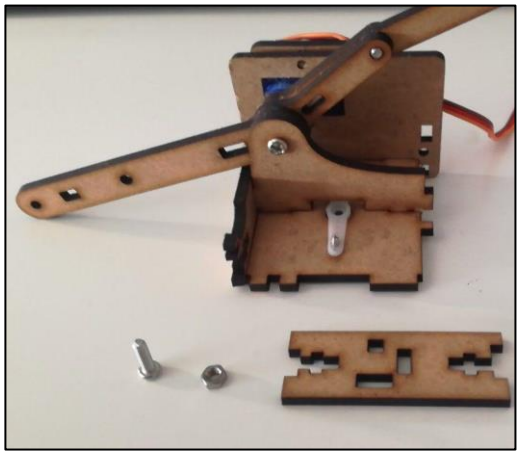


Figura 3-22. Elementos de la unión de la parte trasera con el eslabón de la base.

Lo primero que haremos será terminar de construir la parte central uniendo la parte trasera al eslabón de la base, encajando la pieza 26 a las piezas 7 y 12, y atornillamos de la misma forma que con la pieza 17.

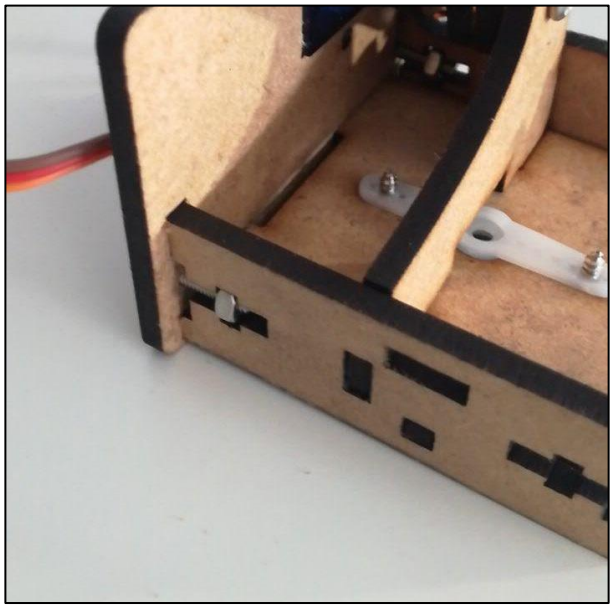


Figura 3-23.Union de la parte trasera con el eslabón de la base

Al unir el lado derecho al eslabón de la base, también uniremos el eslabón que controla el hombro con el soporte central. Cogemos la pieza 27 y la encajamos al eslabón que corresponde con la pieza 21 y la atornillamos con su tuerca como se ha estado haciendo en los últimos pasos. Encajamos el lado derecho en el hueco que queda y encajando su articulación que está controlada por el servo a la pieza 21 y atornillamos.

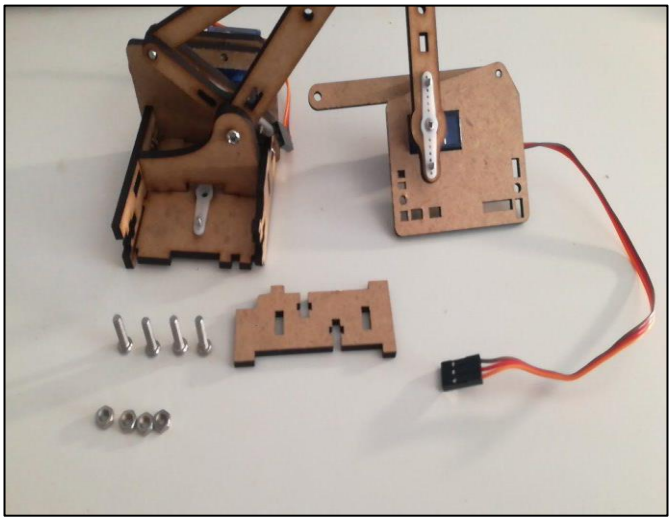
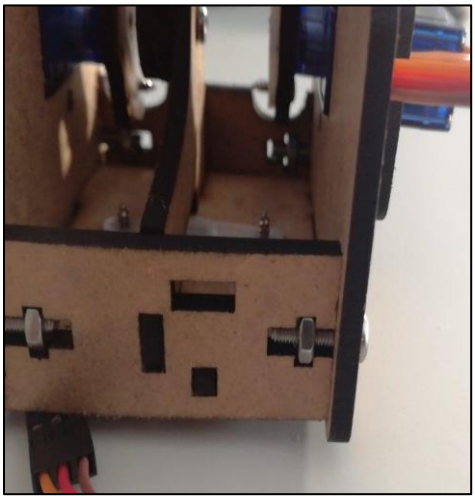
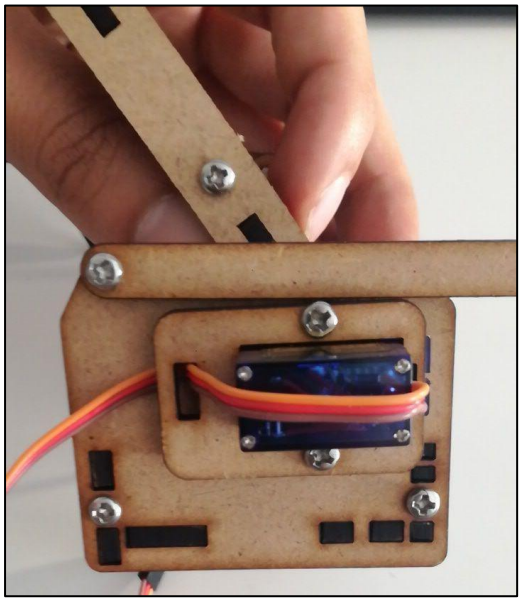


Figura 3-24. Elementos de la parte derecha con el eslabón de la base





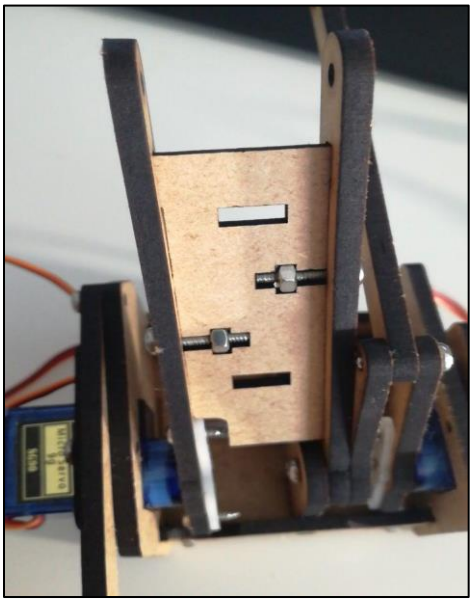


Figura 3-25. Unión del lado derecho al eslabón de la base

Con esto ya tendremos terminado la unión del robot con la base, solo faltaría unirlo a la articulación y ya el robot podría girar sobre su base habiendo conseguido ya la articulación q_0. Con esto, vamos a terminar las otras dos articulaciones de forma que al moverse el hombro también mueva el codo, para ello debemos unir los eslabones de cada lado. Dicho esto, nos ayudaremos del soporte central para llevar a cabo también este objetivo.

Para terminar de montar el codo, primero debemos unir los dos eslabones del lado izquierdo con la pieza 19 y dos tornillos de 6 mm, para finalizar con las piezas 9, 8, 22 y 20, dos tornillos de 10 mm y tres de 6 mm, terminaremos de unir todos los eslabones del robot para que se muevan conjuntamente; pero antes, vamos a atornillar el brazo a la base ya que más adelante será más difícil montarlo.

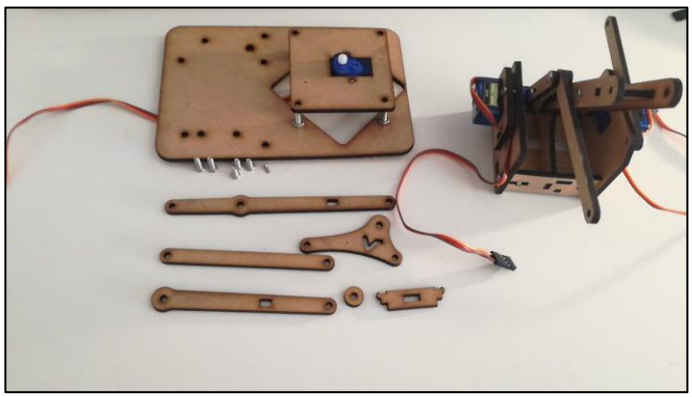


Figura 3-26. Elementos para unir las articulaciones.

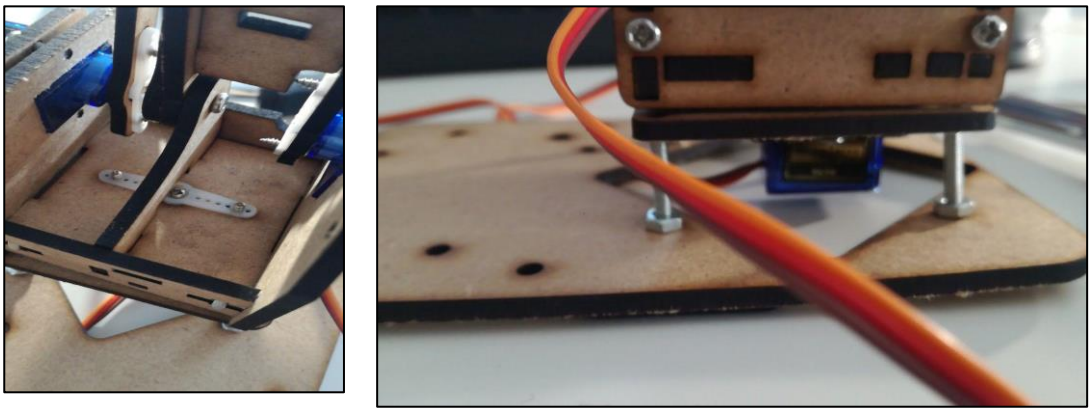


Figura 3-27. Unión del eslabón de la base con su articulación

Atornillamos la pieza 19 a los eslabones del lado izquierdo y central con tornillos de 6 mm, atornillando en el agujero de en medio de la pieza 19, la articulación central por el servo y en el extremo más cercano atornillamos el eslabón controlado por el servo. Con esto ya se puede apreciar el codo del robot acabado.

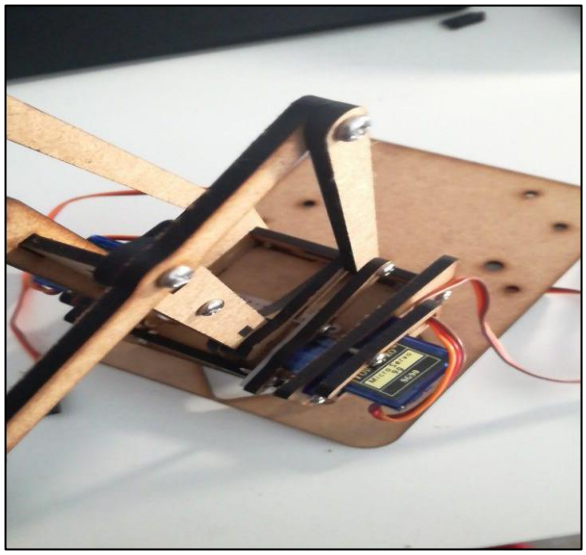


Figura 3-28. Codo del robot

Una vez terminado el codo vamos a unirlo al hombro antes de acabar con este. Unimos los eslabones de cada lado (codo y hombro) con la pieza 9 y añadiéndole al lado del hombro la pieza que faltaría para unirlo, que sería la pieza 22, sin atornillarlo todavía, ya que faltaría una pieza que se usará en los siguientes pasos. Así nos aseguraremos de que, al mover el hombro, el codo también se moverá con él.

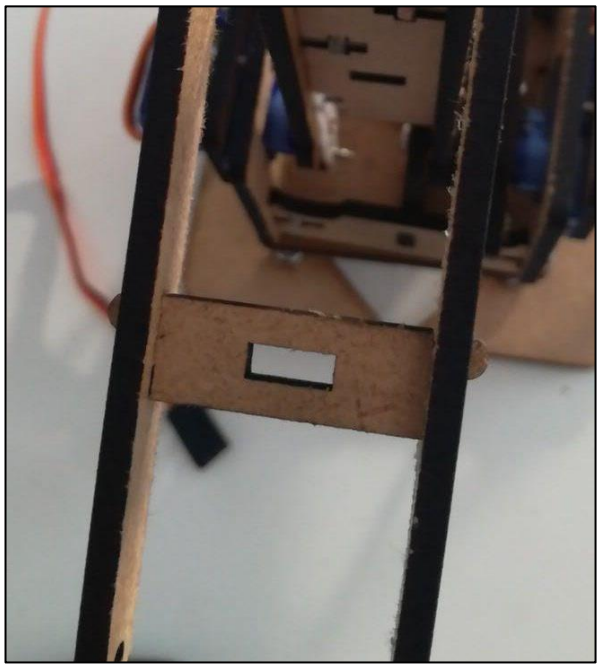


Figura 3-29. Unión de las articulaciones del codo y el hombro.

El siguiente paso será acabar el hombro y con ello todo el brazo del robot. Cogemos la pieza 8 y 20. Atornillamos la parte más alejada de la pieza 8 al eslabón suelto del lado del hombro con un rodamiento en medio con un tornillo de 10 mm, teniendo en cuenta que el extremo más ancho debe ir en la parte baja.

La parte baja de la pieza 8 la atornillaremos con un tornillo de 10 mm, debido a que tendrá que unir 3 piezas, y en el otro agujero de la pieza se atornillara con un tornillo de 6 mm la pieza 20. Una vez hecho esto, habremos acabado con el brazo y procederemos a dar el último paso.

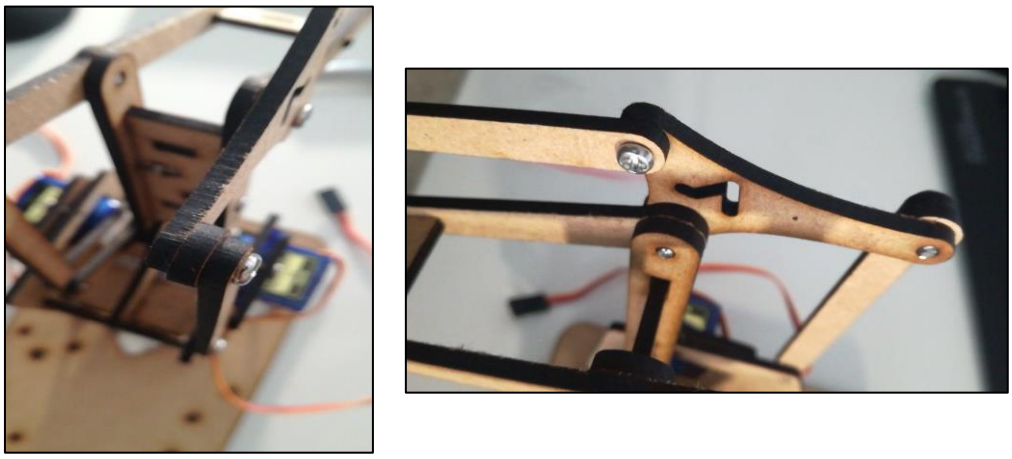


Figura 3-30. Hombro del robot

3.1.5 Pinza

Para acabar, construiremos la pinza del robot y la colocaremos en el brazo acabando, de esta forma, el montaje del brazo robótico. Las piezas que usaremos para la pinza serán la 3, 6, 10, 13, 14, 15, 18, 28 y 29 junto con 1 tornillo de 12 mm, 2 tornillos de 6 mm y 6 tornillos de 8 mm y el servo a usar en la pinza.

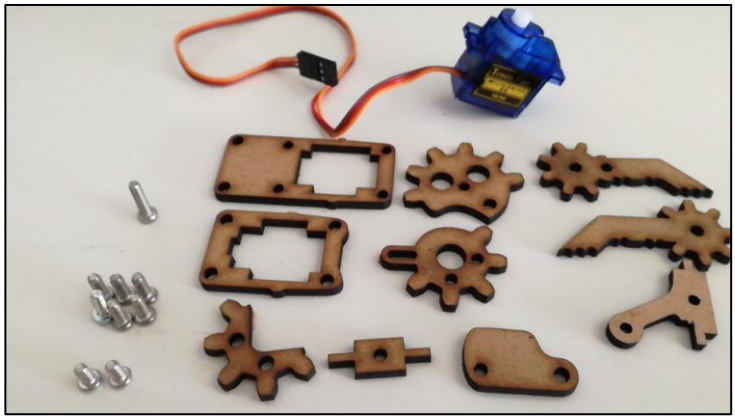


Figura 3-31. Elementos de la pinza

Cogemos las piezas 3 y 6 e introducimos la 6 por el servo para hacer presión al atornillarlo a la pieza 3 con 4 tornillos de 8 mm. Antes de atornillar, debemos meter también entre las dos piezas la pieza 28 y 29 de forma que la parte circular de la pieza 28 este mirando a la zona que sobra de la pieza y esta esté por el lado derecho. La pieza 29 se pondrá en el lado contrario. Es importante que se pongan ya que las usaremos para atornillarla al brazo.

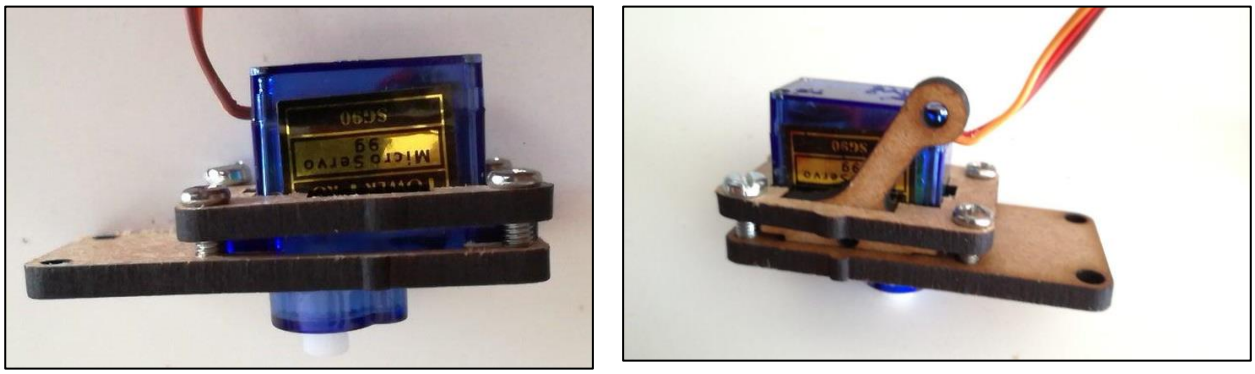


Figura 3-32. Articulación para abrir y cerrar la pinza.

Con las piezas 10 y 18 colocaremos las pinzas. En la pinza que tenga el agujero atornillaremos la de 12 mm, de forma que sobresalga el tornillo un poco. Esa parte debe ser también atornillada por el agujero de la pieza 10 que es más fino. El otro agujero con la otra parte de la pinza se atornillará con un tornillo de 8 mm.

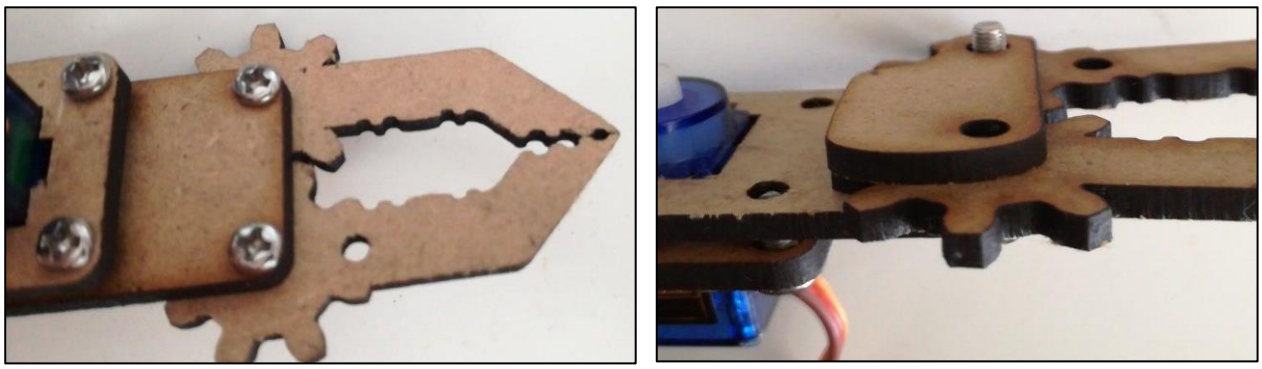


Figura 3-33. Agarre de las pinzas

El siguiente paso será atornillar el servo a los engranajes. Para ello usamos de nuevo uno de los accesorios que vienen con el servo para que lo gire cuando el servo lo haga. El engranaje que se atornillará al servo será el 25, atornillándose a su vez en ese engranaje la pieza 14 con tornillos de 6 mm para darle más superficie de paso. Por otro lado, se pondrá el otro engranaje (13) en el tornillo de 12 mm atornillándolo a su vez, con un tornillo de 8 mm, por el otro agujero, a la pinza para que se abra o cierre. Una vez hecho esto la pinza estará acabada y solo quedará unirla al brazo.

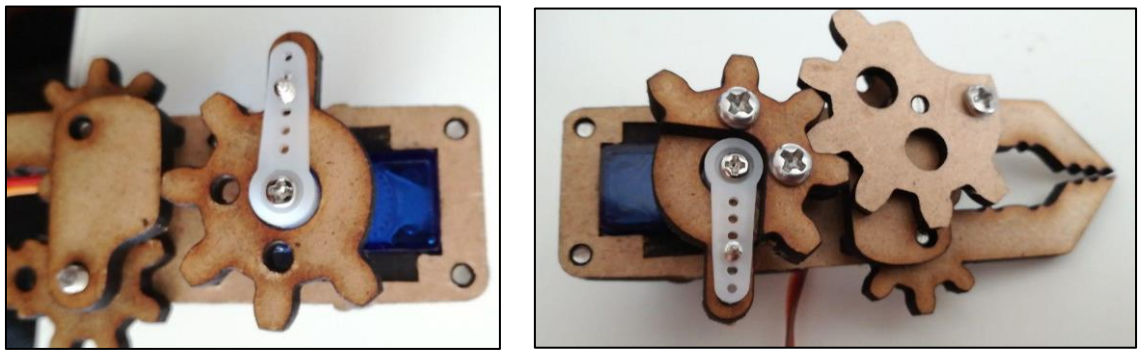


Figura 3-34. Unión de la articulación con el agarre de las pinzas mediante engranajes

Con los eslabones del brazo atornillaremos la pinza con tornillos de 8 mm. Y con esto ya el robot estará montado y podemos proceder al control del mismo.

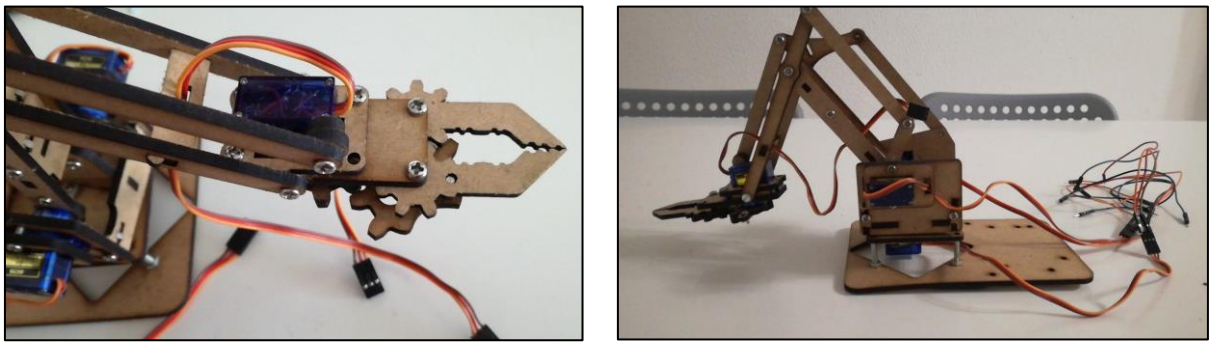


Figura 3-35. Robot manipulador a programar

3.2 Cinemática del robot

En este apartado calcularemos los modelos cinemáticos directo e inverso, ya que serán necesarios para la programación y control del mismo. Empezaremos calculando el modelo directo mediante Denavit-Hartenberg y seguidamente sacaremos a partir de ahí la cinemática inversa.

3.2.1 Calculo del modelo cinemático directo

Una vez montado el robot, empezaremos a estimar su modelo cinemático. El problema cinemático se reduce a encontrar la matriz de transformación homogénea que relaciona la posición y orientación del extremo del robot respecto a su sistema de referencia fijo (su base). Esta matriz A está en función de los parámetros de las articulaciones del robot. Para obtener esta matriz utilizaremos el método de Denavit-Hartenberg anteriormente explicado.

El robot que estamos intentando controlar, se trata de un robot manipulador de 3 GDL con articulaciones rotacionales. Por lo tanto, tenemos tres eslabones que los denominaremos L1, L2 y L3, y tres ángulos por articulación que nombraremos al ángulo de la base q_1, al del hombro q_2 y el codo q_3.

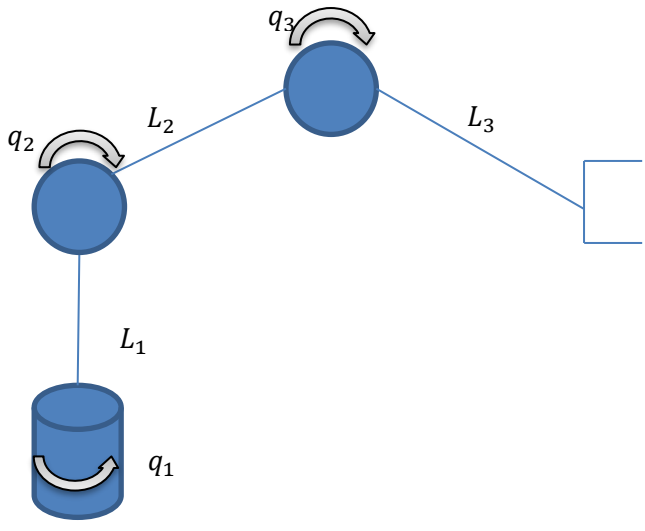


Figura 3-36. Asignación de parámetros del robot.

Siguiendo los pasos del Denavit-Hartenberg, hemos enumerado los eslabones y articulaciones desde la base hasta la pinza. El siguiente paso será localizar el eje de la articulación y asignarles los ejes Z a cada articulación.

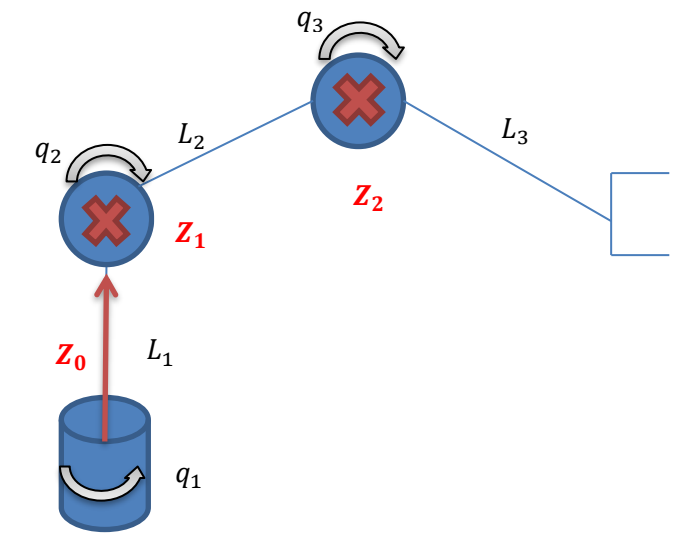


Figura 3-37. Coordenadas Z de cada articulación

Para acabar, colocaremos lo ejes de forma que se cumple que son ejes dextrógiros, y colocaremos el eje Z de la pinza que debe ser paralelo al de la última articulación.

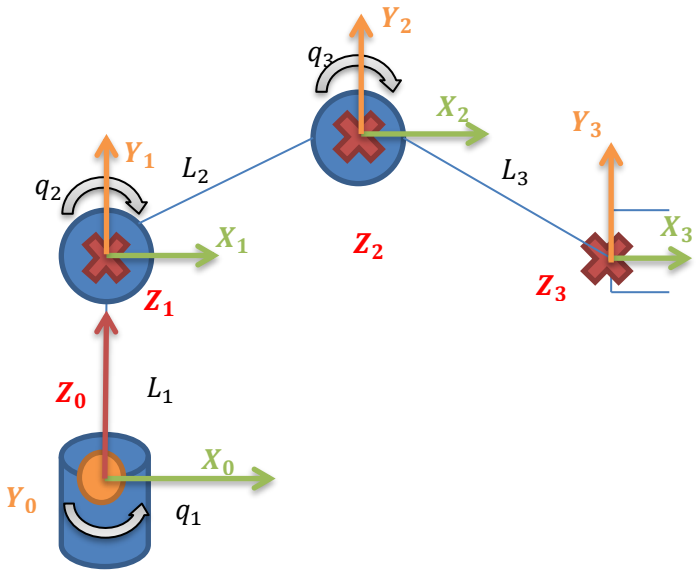


Figura 3-38. Coordenadas de referencia XYZ de cada articulación

Una vez puesto todos los sistemas de referencia, vamos a calcular la matriz de transformación que relaciona los parámetros del robot según el método de que estamos siguiendo. El siguiente paso es identificar cuanto hay que girar o a que distancia se encuentran entre los distintos ejes de X y Z entre las articulaciones. En la siguiente tabla se encuentran los valores en cada articulación siguiendo el significado de cada variable comentado en el apartado 2.2.1.3.

Articulación	θ	d	a	α
1	q_1	L_1	0	90º
2	q_2	0	L_2	0
3	q_3	0	L_3	0

Tabla 3-1. Parámetros de Denavit-Hartenberg

Con los parámetros calculados, sacamos las matrices de transformación entre cada uno de los ejes de referencia y las multiplicamos para obtener la matriz de transformación entre el eje de la pinza y el eje de coordenadas 0. Finalmente obtendremos el modelo cinemático directo.

$$ {}^{0}A_1 = \begin{pmatrix} \cos(q_1) & 0 & \sin(q_1) & 0 \\ \sin(q_1) & 0 & \cos(q_1) & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3-1} $$

$$ {}^{1}A_2 = \begin{pmatrix} \cos(q_2) & -\sin(q_2) & 0 & L_2\cos(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & L_2\sin(\mathrm{q}_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3-2} $$

$$ {}^{2}A_3 = \begin{pmatrix} \cos(q_3) & -\sin(q_3) & 0 & L_3\cos(q_3) \\ \sin(q_3) & \cos(q_3) & 0 & L_3\sin(\mathrm{q}_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3-3} $$

$$ T = {}^{0}A_1\,{}^{1}A_2\,{}^{2}A_3 \tag{3-4} $$

$$ \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{pmatrix} L_3\cos(q_1)\cos(q_2)\cos(q_3) - L_3\cos(q_1)\sin(q_2)\sin(q_3) + L_2\cos(q_1)\cos(q_2) \\ L_3\sin(q_1)\cos(q_2)\cos(q_3) - L_3\sin(q_1)\sin(q_2)\sin(q_3) + L_2\sin(q_1)\cos(q_2) \\ L_3\sin(q_2)\cos(q_3) + L_3\cos(q_2)\sin(q_3) + L_2\sin(q_2) + L_1 \\ 1 \end{pmatrix} \tag{3-5} $$

3.2.2 Calculo modelo cinemático inverso

En muchas aplicaciones, el problema cinemático inverso ha de resolverse en tiempo real. Al contrario de lo que ocurría en el problema cinemático directo, con cierta frecuencia la solución del problema cinemático inverso no es única, existiendo diferentes posiciones que pueden posicionan y orientan el extremo del robot del mismo modo. En estos casos una solución cerrada permite incluir determinadas reglas o restricciones que aseguren que la solución obtenida sea la más adecuada posible.

El modelo cinemático inverso podremos obtenerlo a partir del modelo cinemático directo ya calculado, sin embargo, vamos a utilizar el método geométrico. Los métodos geométricos permiten tener normalmente los valores de las primeras variables articulares, que son los que consiguen posicionar el robot. Para ello utilizan relaciones trigonométricas y geométricas sobre los elementos del robot. Se suele recurrir a la resolución de triángulos formados por los elementos y articulaciones del robot.

El procedimiento se basa en encontrar un número suficiente de relaciones geométricas en las que intervendrán las coordenadas del extremo del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos. Como se ve, el robot posee una estructura planar, quedando este plano definido por el ángulo de la primera variable articular q1.

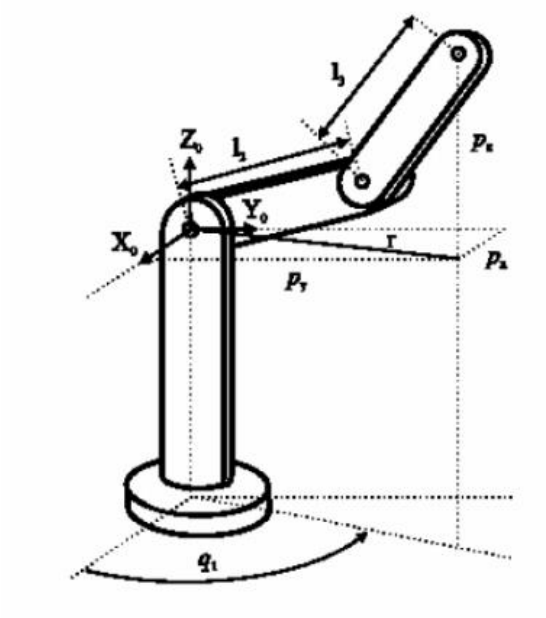


Figura 3-39. Valores geométricos del robot.

$$q_1 = arctg\left(\frac{P_y}{P_x}\right) \tag{3-6}$$

$$\left.\begin{array}{c} r^2 = p_x^2 + p_y^2 \\ r^2 + p_z^2 = L_2^2 + L_3^2 + 2L_3L_2\cos(q_3) \end{array}\right\} \rightarrow \cos(q_3) = \frac{p_x^2 + p_y^2 + p_z^2 - L_2^2 - L_3^2}{2L_2L_3} \tag{3-7}$$

$$\sin(q_3) = \pm\sqrt{1 - \cos^2(q_3)}$$

$$q_3 = arctg\left(\frac{\pm\sqrt{1 - \cos^2(q_3)}}{\cos(q_3)}\right) \tag{3-8}$$

Hasta aquí no hay ninguna restricción que podamos añadir ya que ninguno de estos valores tiene más de una solución posible. Con q_1 si nos encontramos con dos soluciones posibles.

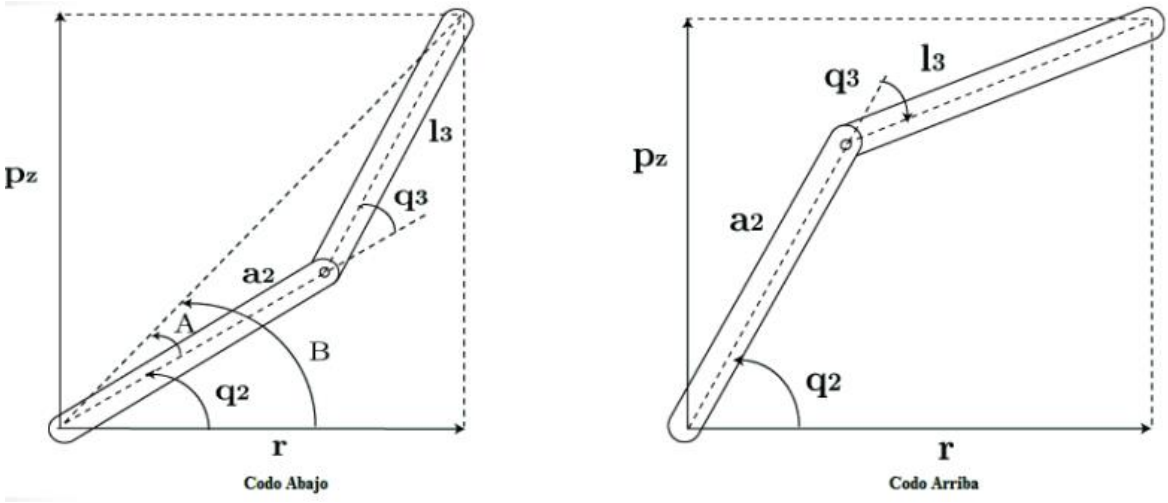


Figura 3-40. Posibles soluciones al MCI.

Nuestro robot tiene el codo arriba por lo que debemos de escoger esa solución de las dos posibles.

$$\begin{aligned} q_2 &= \beta + \alpha \quad \text{(Codo arriba)} \\ q_2 &= \beta - \alpha \quad \text{(Codo abajo)} \end{aligned} \tag{3-9}$$

$$\beta = arctg\left(\frac{p_z}{r}\right) = arctg\left(\frac{p_z}{\pm\sqrt{p_x^2 + p_y^2}}\right) \tag{3-10}$$

$$\alpha = arctg\left(\frac{L_3 \sin(q_3)}{L_2 + L_3 \cos(q_3)}\right) \tag{3-11}$$

4 ELECTRÓNICA

La parte electrónica del proyecto se divide en varias partes. Para hacerlo más sencillo de entender vamos a explicar cada conexión de los dispositivos por separado indicando en que pines del microcontrolador se han conectado. Debido al gran número de dispositivos que se van a usar, necesitaremos una batería adicional para darle carga a los servos y todo funcione correctamente.

4.1 Servomotores

Los servos usados en el robot, son lo Tower SG90. Estos servos son capaces de girar hasta 90º o 360º. Tienen un voltaje de funcionamiento de entre 4.8V hasta 6V. Un servo girará un cierto número de grados dependiendo de la cantidad de tiempo que recibe una señal en alto en un período determinado. A este tiempo que se mantiene a señal alta se le conoce como duty cycle. En el caso de este servo que gira hasta unos 180º, tenemos un período de 20 ms. Para que el servo se posicione a 90º, tiene que recibir un duty cycle de 1.5ms, a 0º un duty cycle de 1ms y a 180º a 2 ms. Dispone de un par de salida de 2.5 kg/cm y una velocidad de 0.1s/60º.

Para poder controlarlo se hará uso de señales PWM (Modulación por Ancho de Pulso).

- Cable rojo: Alimentación 4,8 - 6 V.
- Cable marrón: Tierra.
- Cable naranja: Señal de control.

Figura 4-1. Servomotor Tower SG90

Se usarán 4 servomotores en total, uno para la base, otro para el hombro, otro para el codo y el último para las pinzas. El servo de la base irá conectado al pin 13, el del codo ira conectado al 12, el hombro al 11 y la pinza al 10.

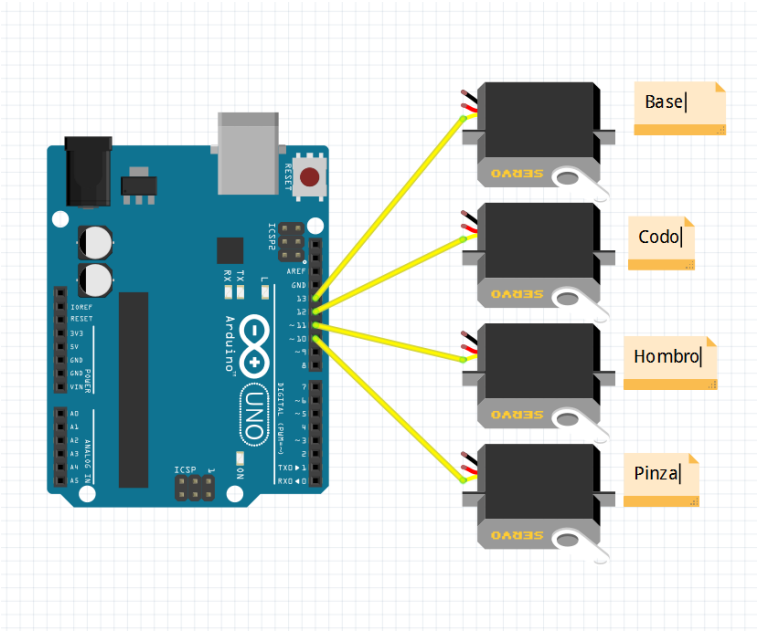


Figura 4-2. Esquemáticos servomotores

4.2 Joysticks

Otro dispositivo que usaremos durante el proyecto será un joystick para manejar el robot manualmente, siendo este modo uno de los tres que explicaremos más adelante. El joystick tiene 5 pines de conexión, un pin para el voltaje de 5V, otro para la tierra, dos pines que irán conectados a la parte analógica de la placa, debido a que nos darán la posición del joystick en los ejes XY, desde 0 a 1023, siendo un pin para el eje X y el otro para el eje Y. Por último, un pin para utilizar el botón del joystick que irá conectado a los pines digitales como entrada a la placa.

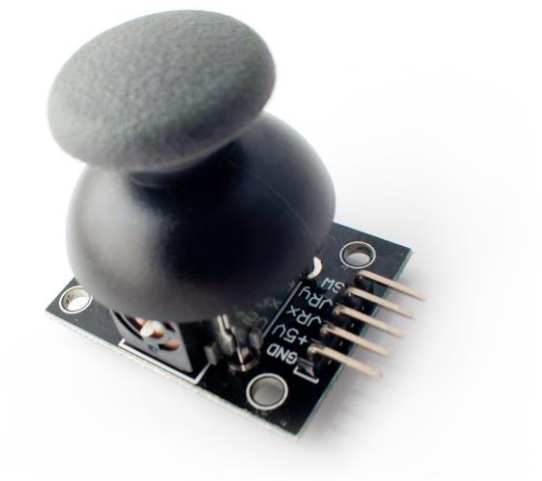


Figura 4-3. Joystick

En el proyecto se han usado 3 joysticks uno por articulación del robot, aunque se puede reducir a 2 sin ningún problema. Los joysticks se usarán para controlar el robot como si se tratara de un mando. Al tener 5 pines cada uno, de los cuales 3 hay que conectar a la placa, los vamos a conectar de forma que el joystick que controla la articulación 1 se conectará el pin de la X al pin analógico A0 y el de la Y al pin analógico A1. El pin del botón de este joystick ira al pin 2 y será el responsable de elegir si nos encontramos en modo de control con mando o control automático del robot.

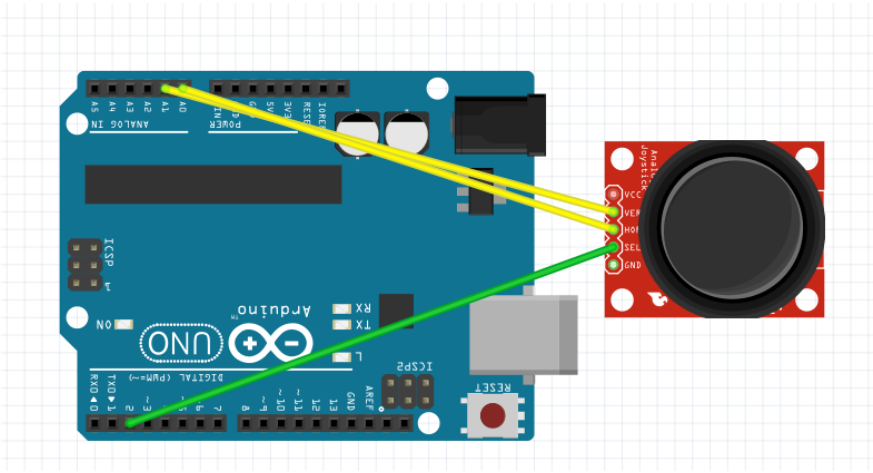


Figura 4-4. Joystick articulación 1.

Los pines del joystick de la articulación 2, se conectará el pin de la X al pin analógico A2 y el de la Y al pin analógico A3. El pin del botón ira conectado al pin 3 y se encargara de variar entre los dos modos que tiene el mando.

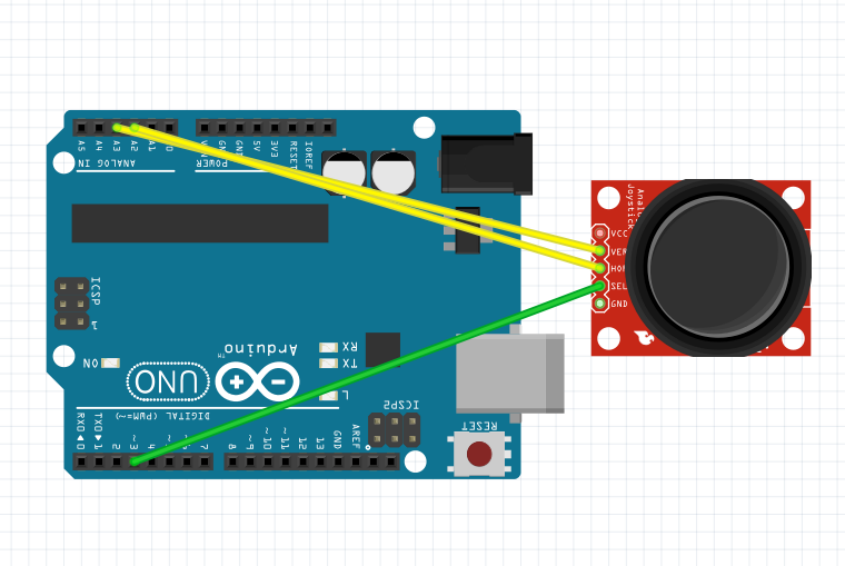


Figura 4-5. Joystick articulación 2

El joystick de la articulación 3 solo se conectará los analógicos. Conectándose el pin X al pin analógico A4 y el de la Y al pin analógico A5.

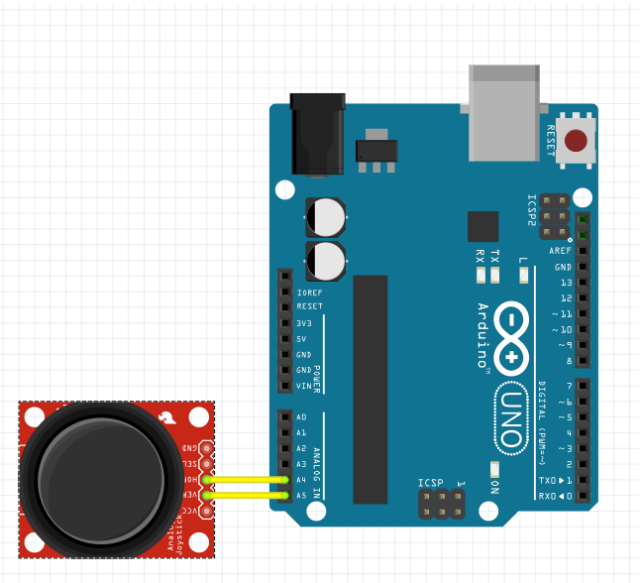


Figura 4-6. Joystick articulación 3

4.3 Otros dispositivos

Además de los nombrados anteriormente, usaremos algunos más comunes como botones y leds.

Figura 4-7. Leds y botones

Los Leds se usarán para indicarnos en qué modo de funcionamiento nos encontramos en ese momento. Se usarán dos Leds uno rojo y otro verde. Si el Led rojo está encendido, estaremos en modo automático si no estaremos en el modo mando. Este Led irá conectado al pin 4. Si el Led verde este encendido indicará que está en el control de mando controlando las articulaciones, si no estaremos en el control cartesiano de la pinza. Este Led irá conectado al pin 5.

Los botones extras que se usarán serán en total dos. Estos botones se usarán para abrir y cerrar la pinza encargándose cada botón de uno de las dos opciones. El botón que se encargará de abrir la pinza se conectaría pin 8 y el de cerrar al pin 9.

5 PROGRAMACIÓN

En cuanto a la programación usada, primero hablaremos de las simulaciones hechas antes de implementarlas en el robot, para asegurarnos que no hay ningún error en el cálculo de los modelos cinemáticos. Seguidamente se implementaría ya con las distintas funciones que le daremos al robot.

Las funciones que se implementarían serán tres modos distintos.

- Un control en el movimiento de los ángulos, haciendo que con los joysticks podamos mover cada una de las articulaciones del robot de forma independiente.
- Un control en el movimiento de la pinza. Usando la cinemática inversa, mediante los joysticks, controlamos la pinza variando cada uno de los ejes cartesianos.
- Un generador de trayectoria. Por puerto serie enviamos la posición y el tiempo que queremos que dure el movimiento, haciendo que el robot se mueva calculando la trayectoria que deberá seguir para cumplir con los requisitos puestos.

5.1 Programas usados

Para el desarrollo del Proyecto se han usado dos programas claves: Matlab para la simulación y Arduino para la programación del microcontrolador.

5.1.1 MATLAB

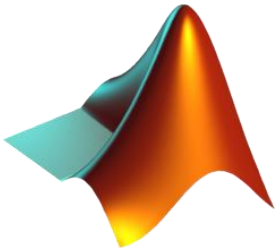


Figura 5-1. Logo Matlab

Es un Sistema de cómputo numérico que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio.

Entre sus prestaciones básicas se hallan la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. Dispone de dos herramientas adicionales, el cual solo usaremos una que se trata de Simulink.

Simulink es un entorno de programación visual de más alto nivel de abstracción que el lenguaje interpretado por Matlab. Simulink viene a ser una herramienta de simulación de modelos o sistemas, con cierto grado de abstracción de los fenómenos físicos involucrados en los mismos.

5.1.2 Arduino

Figura 5-2. Logo Arduino

Es una compañía de desarrollo de software y hardware de fuente abierta, así como una comunidad internacional que diseña y manufactura placas de desarrollo hardware, como la que usamos en el proyecto, para construir dispositivos digitales y dispositivos interactivos que puedan detectar y controlar objetos del mundo real.

El software de Arduino que vamos a usar, consiste de dos elementos: un entorno de desarrollo (IDE), y en el cargador de arranque que es ejecutado de

forma automática dentro del microcontrolador en cuanto este se enciende. Las placas Arduino se programan mediante un computador, usando comunicación serie.

5.2 Simulación

Para asegurarnos de que el resultado obtenido es el correcto, se comprueba su validez con la ayuda de la herramienta informática Matlab.

La herramienta de software Matlab ofrece, entre sus muchas funcionalidades, la posibilidad de realizar scripts escritos con su propio lenguaje de computación y con un asistente. Dicho esto, se ha programado tres scripts distintos en Matlab. Con estos programadas creados y la ayuda de Simulink, se han ido comprobando los resultados de los modelos y viendo si los resultados obtenidos son coherentes o no.

5.2.1 Función cinemática directa

El primer programa implementado se trata de la implementación del modelo cinemático directo. Este script lo hemos programado como una función de Matlab ya que se usará para simular el robot en el Simulink. En este programa ponemos como entrada los valores dados en cada articulación, y usando la ecuación (3-5) correspondiente al modelo directo, calculamos los valores de cada uno de los ejes cartesianos X, Y y Z siendo estos las salidas del programa. Los valores de los eslabones se han calculado como la distancia entre cada articulación, no como la suma de los eslabones que lo componen, ya que podría perjudicar el control debido a que intente llegar a lugares inaccesibles. Esto se debe a que estamos asumiendo que las articulaciones miden más o menos de lo que realmente valen, es decir, estaríamos diciendo que el eslabón es más grande o más pequeño de lo que es, lo que producirá un fallo en los cálculos que podrían ser considerables o no dependiendo de la magnitud del error producido con las distancias entre articulaciones.

```
function xyz = CinematicaDirecta(in)
q1       = in(1);           % Posición articular
q2       = in(2);           %
q3       = in(3);
l0=0.03; % Eslabón 1
l1=0.08; % Eslabón 2
l2=0.08; % Eslabón 3

  % A rellenar por el alumno

  x=l1*cos(q1)*cos(q2) + l2*cos(q1)*cos(q2)*cos(q3) - l2*cos(q1)*sin(q2)*sin(q3);
  y=l1*cos(q2)*sin(q1) + l2*cos(q2)*cos(q3)*sin(q1) - l2*sin(q1)*sin(q2)*sin(q3);
  z=l0 + l1*sin(q2) + l2*cos(q2)*sin(q3) + l2*cos(q3)*sin(q2);

xyz=[x;y;z];
```

Figura 5-3. Función cinemática directa

5.2.2 Función cinemática inversa

El segundo programa, muy similar al anterior, se trata de la implementación del modelo cinemático inverso. Este programa nos ayudará a la hora de decidir qué ángulos debe tomar el robot cuando le indicamos unas coordenadas a las que se debe mover, por ejemplo, en el generador de trayectoria, donde le indicamos la posición final a la que queremos que vaya. En este script tenemos como entrada la posición en coordenadas XYZ a la que se encuentra el robot o queremos que se encuentre.

```
function [q1,q2,q3] = Cinematica_Inversa( in )

px=in(1);
py=in(2);
pz=in(3);
```

Figura 5-4. Entradas de la función cinemática inversa

Utilizando las ecuaciones de 3-8 hasta 3-11 calcularemos los ángulos que deben girar las articulaciones para moverse a la posición deseada. Estas posiciones serán la salida del programa. Al igual que en el script del modelo cinemático directo, lo hemos programado como una función ya que lo usaremos en el siguiente programa y el más importante, que es el generador de trayectoria.

```
L1=0.03;
L2=0.08;
L3=0.08;
pz=pz-L1;


% Calculamos las variables articulares en función de la posición
cosq3 = (px^2 + py^2 + pz^2 - L2^2 - L3^2)/(2*L2*L3);
q1 = atan2(py,px);
q31 = atan2(sqrt(1-cosq3^2),cosq3);
q32 = atan2(-sqrt(1-cosq3^2),cosq3);

 if abs(q31-q3ref)<abs(q32-q3ref)
    q3=q31;
else
    q3=q32;
end

q21 = atan2(pz,(sqrt(px^2 + py^2))) - atan2((L3*sin(q3)),(L2+L3*cosq3));
q22 = atan2(pz,(-sqrt(px^2 + py^2))) - atan2((L3*sin(q3)),(L2+L3*cosq3));


q2=q21;

if abs(imag(q2))<1e-6
    q2=real(q2);
end

if abs(imag(q3))<1e-6
    q3=real(q3);
end


end
```

Figura 5-5. Ecuaciones de la función cinemática inversa

5.2.3 Generador Spline Cúbico

Como ya sabemos, hay distintas formas de hacer un generador de trayectoria. El que vamos a diseñar será el spline de tipo 3. Este generador de trayectoria, creará entre tramos una trayectoria cúbica que más se ajuste a los puntos definidos. Como vamos a querer que un robot haga un movimiento determinado eligiendo él la

trayectoria, vamos a darle un tiempo para que este se mueva e incluso le diremos cuando queremos que empiece a moverse. Por último, le diremos también cuántos puntos de muestreo queremos entre punto y punto. Por lo tanto, los parámetros de entrada serán los siguientes y en ese orden:

- Xini: Posición en X inicial del robot en la trayectoria.
- Yini: Posición en Y inicial del robot en la trayectoria.
- Zini: Posición en Z inicial del robot en la trayectoria.
- Xf: Posición en X final del robot en la trayectoria.
- Yf: Posición en Y final del robot en la trayectoria.
- Zf: Posición en Z final del robot en la trayectoria.
- N: Número de puntos de muestreo que se van a considerar a lo largo de la trayectoria.
- Tini: Tiempo en el que queremos que el robot empiece desde que enviemos el comando.
- D: Duración que queremos darle al movimiento del robot para que haga la trayectoria comenzando en tini.
- T: Tiempo actual.

```
function [out] = generadorspline(in)
xini=in(1);
yini=in(2);
zini=in(3);
xfin=in(4);
yfin=in(5);
zfin=in(6);
n=in(7);
tini=in(8);
d=in(9);
t=in(10);
```

Figura 5-6. Parámetros de entrada del generador de trayectoria.

Una vez dado estos parámetros comenzaremos a estimar las trayectorias entre los puntos escogidos. Siguiendo la definición descrita en el apartado 2.2.4.3, el generador creará las trayectorias entre los puntos escogidos siguiendo sus respectivas ecuaciones. Por lo que nuestro primer paso en el programa, antes de empezar a calcular las ecuaciones, es escoger los puntos que vamos a usar a lo largo de la trayectoria. Los puntos seleccionados se han metido en un vector del tamaño de n+2, añadiendo también la posición final e inicial. Para calcular los puntos intermedios, hemos calculado el incremento que habría entre puntos restando la posición final con la inicial y dividiéndolo entre el número de incrementos totales que sería n+1.

```
%duracion entre movimientos
dmv=d/(n+1);

x=[xini:(xfin-xini)/(n+1):xfin];
y=[yini:(yfin-yini)/(n+1):yfin];
z=[zini:(zfin-zini)/(n+1):zfin];

%si la posicion inicial y final es la misma llenamos la trayectoria con la
%misma posicion puntos+2 veces
if xini==xfin
    x=linspace(xini,xfin,(n+2));
end
if yini==yfin
    y=linspace(yini,yfin,(n+2));
end
if zini==zfin
    z=linspace(zini,zfin,(n+2));
end
```

Figura 5-7. Cálculo de las posiciones intermedias.

Calculadas ya los puntos por los que debe pasar el robot según el número de muestras, utilizaremos la cinemática inversa con estos puntos para saber a qué ángulo[1] deben estar las articulaciones para que el extremo del robot se posicione en ese punto. Este cálculo lo haremos cuando el tiempo en el que estemos sea menor que el tiempo que queramos que empiece el robot a moverse. Esta condición no es necesaria y perjudicará cuando lo hagamos en Arduino ya que debemos de poner un tiempo inicial mayor que 0 para que haga estos cálculos previos, si no, no los hará y el robot no se moverá como esperamos. El uso de esta condición simplemente es por el tiempo de cómputo, ya que, si no la ponemos, hará este cálculo cada vez que se meta en la función lo que requerirá más tiempo del necesario realmente. Los valores se guardarán en variables persistentes para que no desaparezca su valor una vez empiece a calcular las posiciones a las que se debe mover el robot.

```
if(t<tini)
for i=1:n+2
    [au, au2, au3]=Cinematica_Inversa([x(i) y(i) z(i)]);
    q1(i)=au;
    q2(i)=au2;
    q3(i)=au3;
end
```

Figura 5-8. Cálculo de los ángulos de las articulaciones en los puntos muestreados.

Una vez calculado los ángulos, lo haremos con las velocidades. En este caso las velocidades iniciales y finales serán 0, ya que se espera que en esos puntos el robot no se esté moviendo. A la hora de calcular las velocidades intermedias, hay que tener en cuenta si la posición en la que nos encontramos en un punto máximo, en tal caso la velocidad debería ser cero. Para ello, comprobamos si la pendiente cambia a los dos lados del punto. Si es así, entonces ponemos la velocidad de la articulación a 0. Si la pendiente no cambia, entonces calcularemos la velocidad con la ecuación (2-15).

[1] Matlab trabaja con los ángulos en radianes

```
for i=2:n+1
     if(sign(q1(i)-q1(i-1))~=sign(q1(i+1)-q1(i)))
        qp1(i)=0;
    else
        qp1(i)=((q1(i+1)-q1(i))/dmv+(q1(i)-q1(i-1))/dmv)/2;
     end
     if(sign(q2(i)-q2(i-1))~=sign(q2(i+1)-q2(i)))
        qp2(i)=0;
    else
        qp2(i)=((q2(i+1)-q2(i))/dmv+(q2(i)-q2(i-1))/dmv)/2;
     end
     if(sign(q3(i)-q3(i-1))~=sign(q3(i+1)-q3(i)))
        qp3(i)=0;
    else
        qp3(i)=((q3(i+1)-q3(i))/dmv+(q3(i)-q3(i-1))/dmv)/2;
     end
end
```

Figura 5-9. Cálculo de las velocidades de las articulaciones en los puntos muestreados.

Una vez supere el tiempo inicial indicado, comenzaremos con el cálculo del ángulo que debe ir tomando el robot en cada instante de tiempo. Utilizaremos las ecuaciones ya anteriormente indicadas, de la (2-9) hasta la (2-14), obteniendo de esta forma el ángulo de las articulaciones que debe tener cada uno en cada momento y la velocidad a la que debe de girar para cumplir todos los requisitos. Para saber sobre que tramo del muestreo nos encontramos en ese instante de tiempo y saber que ángulo de los calculamos debemos considerar, hemos calculado el tramo de la siguiente forma:

$$p = \left\lfloor \frac{t - tini}{dmv} + 1 \right\rfloor \tag{5-1}$$

```
if (t<(tini+d))
    p=floor((t-tini)/dmv)+1;

%calculamos los angulos en cada punto mediante el modelo inverso
    a1=q1(p);
    a2=q2(p);
    a3=q3(p);
    b1=qp1(p);
    b2=qp2(p);
    b3=qp3(p);
    c1=3*(q1(p+1)-q1(p))/(dmv^2)-(qp1(p+1)+2*qp1(p))/dmv;
    c2=3*(q2(p+1)-q2(p))/(dmv^2)-(qp2(p+1)+2*qp2(p))/dmv;
    c3=3*(q3(p+1)-q3(p))/(dmv^2)-(qp3(p+1)+2*qp3(p))/dmv;
    d1=-2*(q1(p+1)-q1(p))/(dmv^3)+(qp1(p+1)+qp1(p))/(dmv^2);
    d2=-2*(q2(p+1)-q2(p))/(dmv^3)+(qp2(p+1)+qp2(p))/(dmv^2);
    d3=-2*(q3(p+1)-q3(p))/(dmv^3)+(qp3(p+1)+qp3(p))/(dmv^2);
    taux=dmv*(p-1)+tini;
```

```
qr1=a1+b1*(t-taux)+c1*(t-taux)^2+d1*(t-taux)^3;
qrp1=b1+2*c1*(t-taux)^2+3*d1*(t-taux)^2;
qrpp1=2*c1+6*d1*(t-taux);

qr2=a2+b2*(t-taux)+c2*(t-taux)^2+d2*(t-taux)^3;
qrp2=b2+2*c2*(t-taux)^2+3*d2*(t-taux)^2;
qrpp2=2*c2+6*d2*(t-taux);

qr3=a3+b3*(t-taux)+c3*(t-taux)^2+d3*(t-taux)^3;
qrp3=b3+2*c3*(t-taux)^2+3*d3*(t-taux)^2;
qrpp3=2*c1+6*d1*(t-taux);
```

Figura 5-10. Cálculo de los ángulos y velocidades que debe ir tomando cada articulación

Una vez nos encontremos fuera del rango de tiempo especificado enviaremos el último valor del ángulo de cada articulación, y pondremos las velocidades a 0, debido a que el robot no debería moverse más. Mediante el Simulink comprobaremos si los resultados son correctos, que se verá en el apartado 6.1.

5.3 Programación del microcontrolador

Con la prueba realizada en Simulink, y habiendo comprobado que el robot realiza los movimientos que debe realizar, toca implementarlo en el microcontrolador con el programa de Arduino. Antes de explicar el algoritmo usado en el programa, destacar que se ha usado una librería externa para el control de la velocidad de los servos, ya que el ofrecido por Arduino solo se puede controlar la posición.

Los pines que se usarán para cada dispositivo ya se comentaron en el apartado 4, por lo que en principio se tendrían que indicar en el Arduino para inicializarlos con sus respectivas variables. El programa se ha dividido en varias funciones. Dos de ellas similares a los del apartado anterior que no varía que son las de los modelos inverso y directo y que, por lo tanto, no explicaremos, y otras tres funciones que serán los tres modos de funcionamiento que le vamos a dar al robot. Por último, es importante explicar la función que se usa para mover el servo. Esta función trabaja en grados, poniéndole a que grado queremos que se mueva el servo, internamente este creará la señal pwm adecuado para que el servo se mueva a ese ángulo. Además, nos permite, como se ha dicho anteriormente, indicar a qué velocidad queremos que se mueva en rad/s. Esta librería se llama VarSpeedServo.h. Sus funciones son similares a la librería proporcionada por Arduino pero añadiendo tiempos de esperas lo que le dará su valor de velocidad.

servo.write(ang,vel,true);

5.3.1 Función main

La función principal se encargará de decidir qué modo del robot se estará ejecutando. Mediante interrupciones del microcontrolador iremos cambiando los estados del modo. Habrá dos modos, si está funcionando autónomamente por comandos o por mando, y si estamos en el modo de mando si el control será cartesiano o angular. Estas interrupciones saltarán cuando se pulsen los botones de los joysticks. Uno será para cambiarlo de autónomo a mando y el otro para los modos del mando. Si estamos en el modo mando simplemente hará lo que la función de cada uno tenga programado.

```
void loop() {
  float refe = -3.14 / 4;
  float q[3];
  cinematicainversa(q,xini,yini,zini,refe);
  digitalWrite(4,ModoControl);
  if(ModoControl==LOW){
    //Serial.println("Control automatico desactivado");
    digitalWrite(5,ModoMando);
    if(ModoMando==HIGH){
      mando();
    }else{
      cinematicadirecta(q);
      cartesiano();

    }
  }else{
    //Serial.println("Control automatico activado");
  cinematicadirecta(q);
  if (flag==1){
      tact=millis();
      generadorspline(ang);
      base.write(posS1+(ang[0]-0),ang[3]*1000,true);
      codo.write(posS3+(ang[2]+71),ang[5]*1000,true);  //180-90
      hombro.write(posS2-(ang[1]-35),ang[4]*1000,true);
      Serial.print("q1=");
      Serial.print(ang[0]*3.14/180,8);
      Serial.println(" rad");
      Serial.print("q2=");
      Serial.print(ang[1]*3.14/180,8);
      Serial.println(" rad");
      Serial.print("q3=");
      Serial.print(ang[2]*3.14/180,8);
      Serial.println(" rad");
      Serial.print("velocidad q1=");
      Serial.print(ang[3]*1000,8);
      Serial.println(" rad/s");
      Serial.print("velocidad q2 =");
      Serial.print(ang[4]*1000,8);
      Serial.println(" rad/s");
      Serial.print("velocidad q3 =");
      Serial.print(ang[5]*1000,8);
      Serial.println(" rad/s");
      Serial.print("tiempo actual=");
      Serial.print((tact-t)/1000);
      Serial.println(" s");
      Serial.print("\n");

     if((tact-(t+tini))>=d){
       flag=0;
       xini=xfin;
       yini=yfin;
       zini=zfin;
     }
   }
```

Figura 5-11. Código main Arduino

Si estamos en el modo autónomo, esperará a recibir un comando por puerto serie. Si el comando recibido es ''open'' la pinza se abrirá, si es ''close'' la pinza se cerrará y por último ''mov xf, yf, zf, n, tin, d'' para que el robot se mueva a la posición indicada (xf, yf, zf) usando el generador de trayectoria simulado en Matlab. Una vez haya calculado la primera posición se enviará la posición a los servos para que este se mueva e ira a calcular la siguiente posición actualizando siempre el tiempo con la función millis();

```
if (Serial.available()) {
  option = Serial.readStringUntil('\n');
  if (option == "open") {
    pinza.write(40);
    Serial.println("Abierto");
  }
  if (option == "close") {
    pinza.write(100);
    Serial.println("Cerrado");
  }
  if (option[3] == ' ' && flag==0 && option[0]=='m' && option[1]=='o' && option[2]=='v' ) {
    byte prevPos = option.indexOf(' ');
    String data = option.substring(prevPos);

    prevPos = data.indexOf(',');
    help = data.substring(0, prevPos);
    help.toCharArray(aux,32);
    v[0]=atof(aux);
    prevPos++;

    byte currPos = data.indexOf(',', prevPos);
    int j;
    for(j=1;j<6;j++){
    v[j] = data.substring(prevPos, currPos).toFloat();
    prevPos = currPos + 1;
    currPos = data.indexOf(',', prevPos);
    }

   xfin = v[0];
   yfin = v[1];
   zfin = v[2];
   n = v[3];
   tini = v[4]*1000;
   d = v[5]*1000;
   flag=1;
   Serial.print("x inicial=");
   Serial.println(xini);
   Serial.print("y inicial =");
   Serial.println(yini);
   Serial.print("z inicial =");
   Serial.println(zini);
   Serial.print("x final=");
   Serial.println(xfin);
   Serial.print("y final =");
   Serial.println(yfin);
   Serial.print("z final =");
   Serial.println(zfin);
   Serial.print("muestras=");
   Serial.println(n);
   Serial.print("t inicial=");
   Serial.println(tini);
   Serial.print("duracion=");
   Serial.println(d);
   t = millis();
   tact=0;
   generadorspline(ang);
   Serial.print("tiempo de comienzo=");
   Serial.println(t/1000);
```

Figura 5-12. Lectura del puerto Serial

5.3.2 Función mando

En esta función controlaremos la velocidad angular del robot con el mando. En un principio se usaron tres joysticks en el que cada uno puede mover un ángulo del robot específico, pero se podrían usar dos ya que cada joystick tiene un eje X y un eje Y. Lo que hará esta función es ir leyendo de las salidas analógicas hasta que detecte en alguno de los joysticks un cambio en el eje X. Si desplazamos el joystick a la derecha ira sumando el incremento de los ángulos de la articulación correspondiente al joystick y a la izquierda los irá restando. Estos se irán sumando o restando de 1 en 1. Como los botones de los joysticks ya están asociados para que salten las interrupciones, se añadieron dos botones para este caso. Estos botones se encargarán de abrir y cerrar la pinza, además de poder hacerlo con el joystick también.

```
void mando(){
   int X1value = analogRead(XJ1);
   delay(100);
   int X2value = analogRead(XJ2);
   delay(100);
   int X3value = analogRead(XJ3);
   delay(100);
   int Y3value = analogRead(YJ3);
   if(X1value>1010){
    incl=incl+1;
   }else if(X1value<40){
    incl=incl-1;
   }
   posS1=posS1+incl;
    base.write(posS1);
    Serial.print("q1: ");
    Serial.print(posS1);
    Serial.println("°");
```

Figura 5-13. Lectura del joystick e incremento o decremento de la posición de la articulación q1

```
 if(digitalRead(Botoncerrar)==LOW){
   pinza.write(100);
   Serial.println("Cerrado");
}
if(digitalRead(BotonAbrir)==LOW){
  pinza.write(40);
    Serial.println("Abierto");
}
if(Y3value>1010){
  pospinza=pospinza+3;
  pinza.write(pospinza);
 }else if(Y3value<40){
  pospinza=pospinza-3;
  pinza.write(pospinza);
 }
delay(50);
```

Figura 5-14. Controles de la pinza

5.3.3 Función cartesiano

Esta función controlará las posiciones en cartesiano del extremo del robot. Se usará dos de los joysticks, uno se usará para controlar las coordenadas X e Y, y el otro para la coordenada Z. Se usará dos botones para abrir y cerrar la pinza como ocurre en la función mando. Como el robot tiene altura medidas en centímetros y en el modelo inverso las medidas están en metros, la cantidad que le restaremos o sumaremos a cada coordenada por los joysticks será de 0.01 m. Una vez restado o sumado, se enviará a la función de la cinemática inversa para que calcule los ángulos que debe girar cada articulación para colocarse en esa posición, controlando de esta forma el extremo del robot en coordenadas cartesianas.

```
void cartesiano(){
   float refe = -3.14 / 4;
   float q[3];

   int X1value = analogRead(XJ1);
   delay(100);
   int Y1value = analogRead(YJ1);
   delay(100);
   int X2value = analogRead(XJ2);
   delay(100);
   int Y3value = analogRead(YJ3);
   if(X1value>1010){
    xini=xini+0.01;
    Serial.print("X: ");
   Serial.println(xini);
   }else if(X1value<40){
    xini=xini-0.01;
    Serial.print("X: ");
    Serial.println(xini);
   }
```

Figura 5-15. Lectura del joystick analógico

```
cinematicainversa(q,xini,yini,zini,refe);

base.write(posS1+(q[0]-0));
hombro.write(posS2-(q[1]-35));
codo.write(posS3+(q[2]+71));
```

Figura 5-16. Escritura en los servos

5.3.4 Función generador spline

Esta función no varía mucho en cuanto al explicado en el simulador. Las únicas diferencias destacables son que, al acabar de moverse a lo largo de la trayectoria, va asignándole a la posición inicial la posición final deseada del robot, quitándolas como variable de entrada al comando que lo activa. Esta variable también se irá actualizando si la movemos con el mando siendo así una variable global. Otro cambio destacado es que los ángulos de la trayectoria que se calculan antes del mover el robot se cambiaron como variables globales para que no se borrara su valor. Hay una segunda forma de hacer esto, pero se ha dejado como futuras mejoras del proyecto y por lo tanto se explicará en su apartado. El resto de la función sigue el mismo algoritmo que en Matlab, primero calcula las posiciones intermedias según el número de muestras que se ha querido coger, calculamos los ángulos que debe tomar el robot para seguir la trayectoria, y vamos calculando en cada instante de tiempo la posición que debe ir tomando. Por último, si no se han calibrado los servos antes de montarlo, el ángulo que calcula la cinemática inversa no coincide con la posición del servo, por lo que se ha calcularía los incrementos de los ángulos y se han sumarían al ángulo del servo. Esto se debe a que la referencia angular que coge el servo no coincidiría con el ángulo real del robot.

```
qp1[n+1]=0;
qp2[0]=0;
qp2[n+1]=0;
qp3[0]=0;
qp3[n+1]=0;
qpp1[0]=0;
qpp1[n+1]=0;
qpp2[0]=0;
qpp2[n+1]=0;
qpp3[0]=0;
qpp3[n+1]=0;
// calculamos los puntos de muestreo intermedio para el calculo de las trayectorias
for (i = 0; i <= n+1; i++) {
    x[i] = i*(xfin - xini) /(n+1) + xini;
    y[i] = i*(yfin - yini) / (n+1) + yini;
    z[i] = i*(zfin - zini) / (n+1) + zini;
}
//posiciones a tomar
if (tact < (tini+t)) {
  for (i = 0; i <= n+1; i++) {
    float refe = -3.14 / 4;
    cinematicainversa(q,x[i], y[i], z[i], refe);
    q1[i] = q[0]*3.14/180;
    q2[i] = q[1]*3.14/180;
    q3[i] = q[2]*3.14/180;

  } // velocidades a tomar en cada punto de muestreo
  for (i = 1; i < n+1; i++) {
    if (((q1[i] - q1[i - 1]) > 0 && (q1[i + 1] - q1[i]) < 0) || ((q1[i] - q1[i - 1]) < 0 && (q1[i + 1] - q1[i]) > 0)) {
     qp1[i]=0;
    } else {
      qp1[i] = ((q1[i + 1] - q1[i]) / dmv + (q1[i] - q1[i - 1]) / dmv) / 2;
    }
    if (((q2[i] - q2[i - 1]) > 0 && (q2[i + 1] - q2[i]) < 0) || ((q2[i] - q2[i - 1]) < 0 && (q2[i + 1] - q2[i]) > 0)) {
      qp2[i]=0;
    } else {
      qp2[i] = ((q2[i + 1] - q2[i]) / dmv + (q2[i] - q2[i - 1]) / dmv) / 2;
    }
    if (((q3[i] - q3[i - 1]) > 0 && (q3[i + 1] - q3[i]) < 0) || ((q3[i] - q3[i - 1]) < 0 && (q3[i + 1] - q3[i]) > 0)) {
      qp3[i]=0;
    } else {
      qp3[i] = ((q3[i + 1] - q3[i]) / dmv + (q3[i] - q3[i - 1]) / dmv) / 2;
    }

  }
  qr1 = q1[0];
  qrp1 = 0;
  qrpp1 = 0;
  qr2 = q2[0];
  qrp2 = 0;
  qrpp2 = 0;
  qr3 = q3[0];
  qrp3 = 0;
  qrpp3 = 0;
  qp1[0]=0;
```

Figura 5-17. Inicialización de las velocidades y posiciones intermedias de la trayectoria.

```
} else if (tact < (tini + d + t)) {
  int p = floor((tact - (tini+t)) / dmv);

//funciones del generador de trayectoria
  a1=q1[p];
  a2=q2[p];
  a3=q3[p];
  b1=qp1[p];
  b2=qp2[p];
  b3=qp3[p];

  c1 = 3 * (q1[p+1] - a1) / (dmv * dmv) - (qp1[p+1] + 2 * b1) / dmv;
  c2 = 3 * (q2[p+1] - a2) / (dmv * dmv) - (qp2[p+1] + 2 * b2) / dmv;
  c3 = 3 * (q3[p+1] - a3) / (dmv * dmv) - (qp3[p+1] + 2 * b3) / dmv;
  d1 = -2 * (q1[p+1] - a1) / (dmv * dmv * dmv) + (qp1[p+1] + b1) / (dmv * dmv);
  d2 = -2 * (q2[p+1] - a2) / (dmv * dmv * dmv) + (qp2[p+1] + b2) / (dmv * dmv);
  d3 = -2 * (q3[p+1] - a3) / (dmv * dmv * dmv) + (qp3[p+1] + b3) / (dmv * dmv);
  float taux = (dmv*p + tini+t);

  qr1 = a1 + b1 * (tact - taux) + c1 * ((tact - taux) * (tact - taux)) + d1 * ((tact - taux) * (tact - taux) * (tact - taux));
  qrp1 = b1 + 2 * c1 * (tact - taux) + 3 * d1 * ((tact - taux) * (tact - taux));
  qrpp1 = 2 * c1 + 6 * d1 * (tact - taux);

  qr2 = a2 + b2 * (tact - taux) + c2 * ((tact - taux) * (tact - taux)) + d2 * ((tact - taux) * (tact - taux) * (tact - taux));
  qrp2 = b2 + 2 * c2 * (tact - taux) + 3 * d2 * ((tact - taux) * (tact - taux));
  qrpp2 = 2 * c2 + 6 * d2 * (tact - taux);

  qr3 = a3 + b3 * (tact - taux) + c3 * ((tact - taux) * (tact - taux)) + d3 * ((tact - taux) * (tact - taux) * (tact - taux));
  qrp3 = b3 + 2 * c3 * (tact - taux) + 3 * d3 * ((tact - taux) * (tact - taux));
  qrpp3 = 2 * c3 + 6 * d3 * (tact - taux);
} else {
  qr1=q1[n + 1];
  qr2=q2[n + 1];
  qr3=q3[n + 1];
  qrp1=0;
  qrp2=0;
  qrp3=0;
  qrpp1=0;
  qrpp2=0;
  qrpp3=0;
```

Figura 5-18. Calculo de las posiciones en cada instante de tiempo

6 RESULTADOS Y FUTURAS MEJORAS

Con el robot ya programado y el simulador, compararemos los resultados de los ángulos para ver si son correctos a la vez que se comprobará en el simulador si sigue la trayectoria correctamente al igual que en el robot. Seguidamente se dirán algunas mejoras que podría tener el proyecto para fortalecer más los conocimientos o utilizar más recursos que nos ofrece la placa.

6.1 Resultados

6.1.1 Generador de trayectoria

Lo primero que comprobaremos es el generador de trayectoria. Mediante el Simulink, comprobamos que los modelos estén bien calculados y el generador de trayectoria. La programación por bloques hecha mediante Simulink es la siguiente:

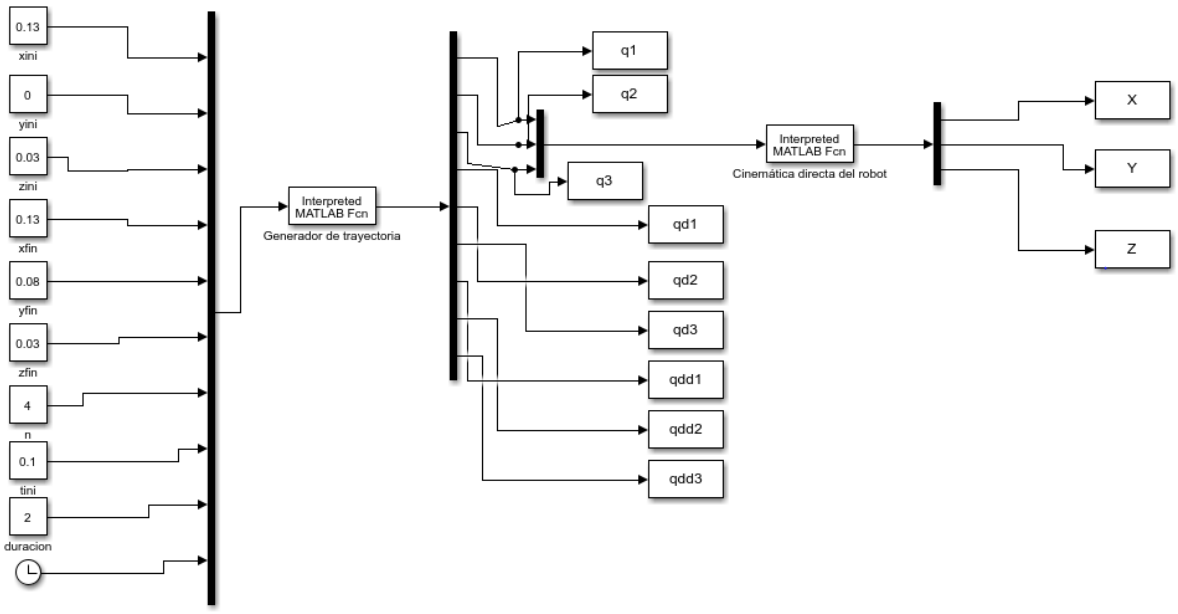


Figura 6-1. Programa por bloques de Simulink

Como se puede observar en la figura 6-1, las variables de entradas elegidas son:

- Xini=0.13
- Yini=0

- Zini=0.03
- Xf=0.13
- Yf=0.08
- Zf=0.03
- N=4
- Tini=0.1
- Duración=2

Con estos valores obtenemos

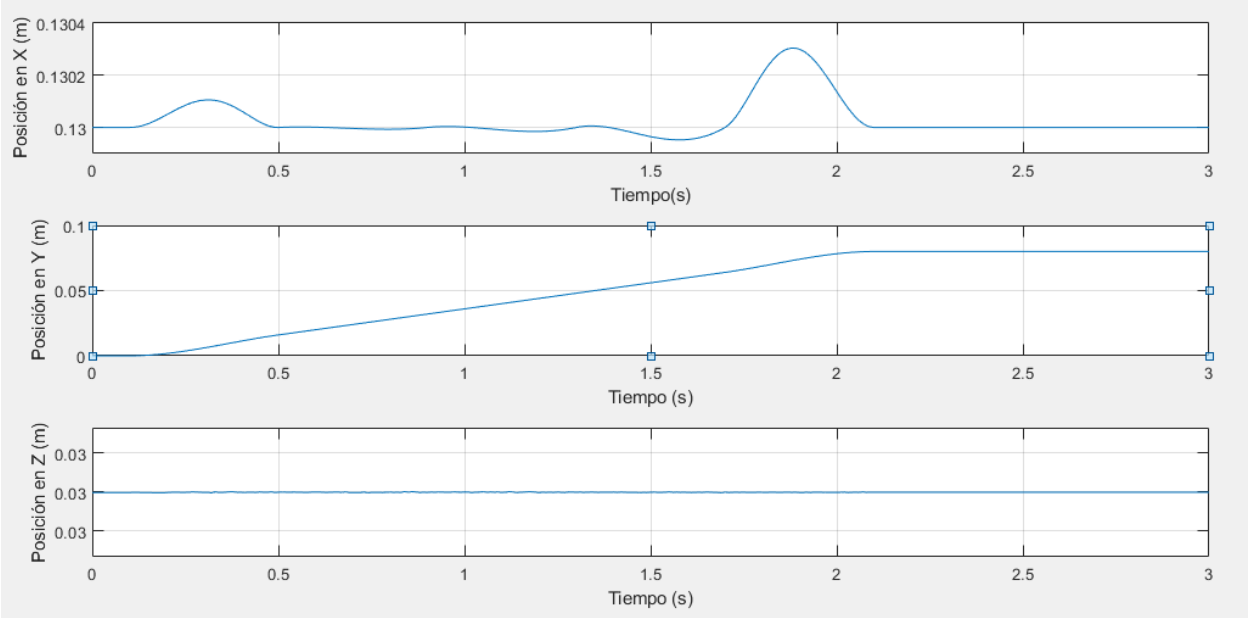


Figura 6-2. Resultados de la simulación

Se puede apreciar en la figura 6-2 como el robot va cambiando su posición hasta llegar a la posición deseada sin ningún problema. En las coordenadas X se desviará debido a que el robot no podría llegar a la posición indicada manteniendo siempre constante las coordenadas X. También se aprecia que los tiempos especificados se cumplen empezando en 0.1 segundo y acabando en 2.1 segundo.

Con estos resultados sabemos que el generador de trayectoria funciona correctamente. Ahora vamos a comparar los resultados de los ángulos obtenidos en esta simulación con los que nos dará el robot. Para ello vamos a comparar los resultados que nos da el programa en Arduino con los de la simulación en Matlab.

Entre las distintas muestras hay una duración de 0.4 segundos, ya que las muestras escogidas son 4. Sabiendo esto intentaremos coger valores intermedios entre cada tramo para que se pueda ir viendo la evolución en las distintas trayectorias y como mantiene el robot en un movimiento de ángulos constantes. Los primeros valores que veremos serán los obtenido antes de que pase el tiempo de inicialización puesto para que se calculen las trayectorias (t<0.1s). A la derecha se observarán los valores obtenidos en Arduino y a la izquierda se mostrará la gráfica generada en Matlab con los resultados que han ido saliendo de los ángulos durante la simulación.

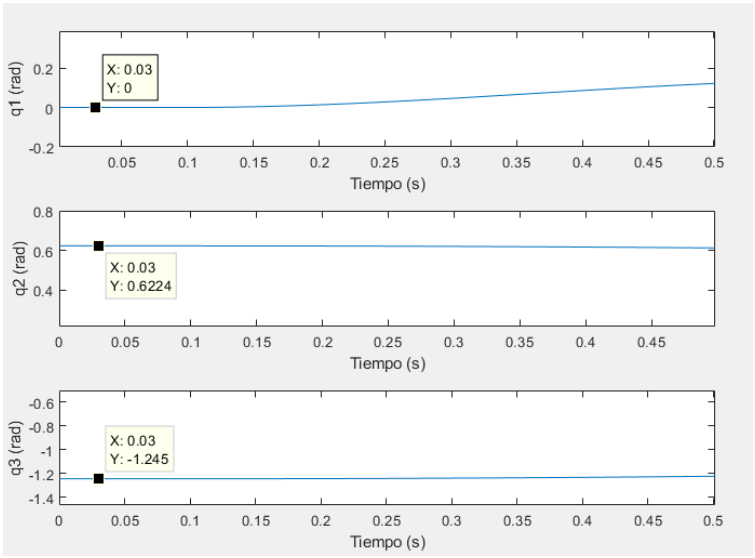


Figura 6-3. Valores para t<0.1s

Se puede observar que el robot espera hasta superar el tiempo de inicio para empezar a moverse. En Arduino se ha guardado en una variable a parte el valor del tiempo cuando se ha comenzado la generación de trayectoria. Vamos a observar ahora lo que ocurre si el robot supera el tiempo de inicio. El tiempo escogido ha sido el momento en el que t=0.19s.

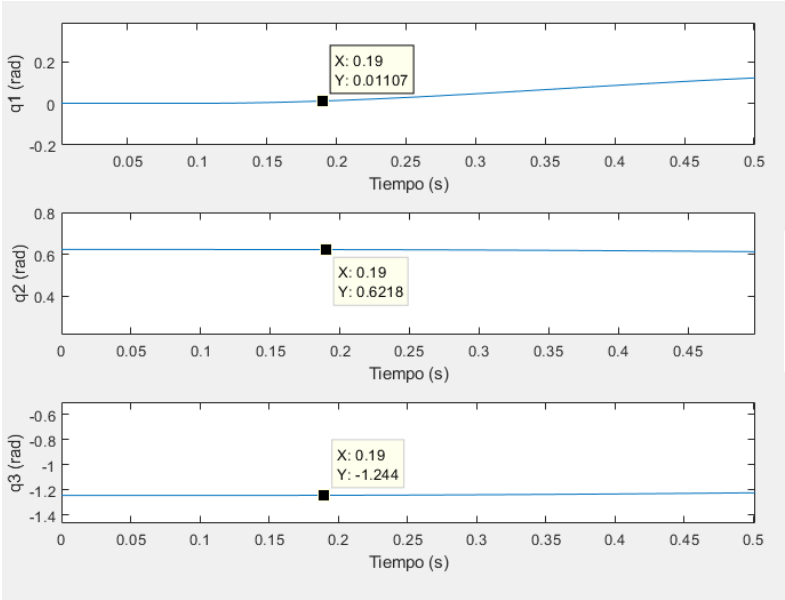
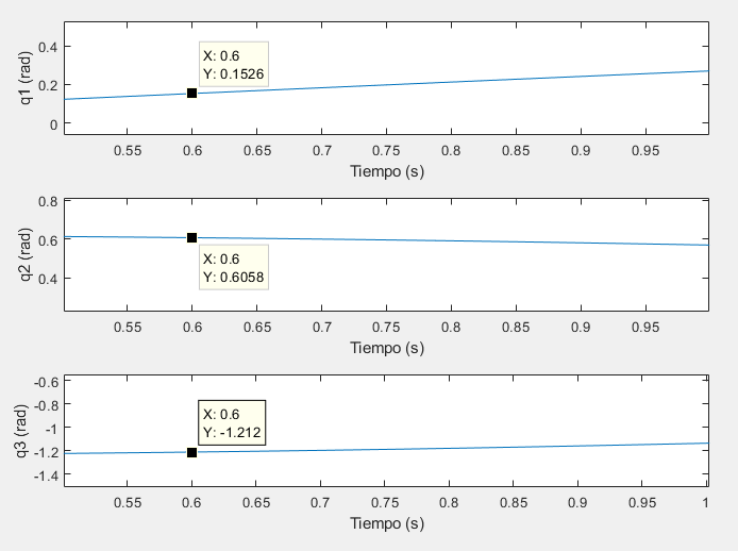


Figura 6-4. Valores para t=0.19s.

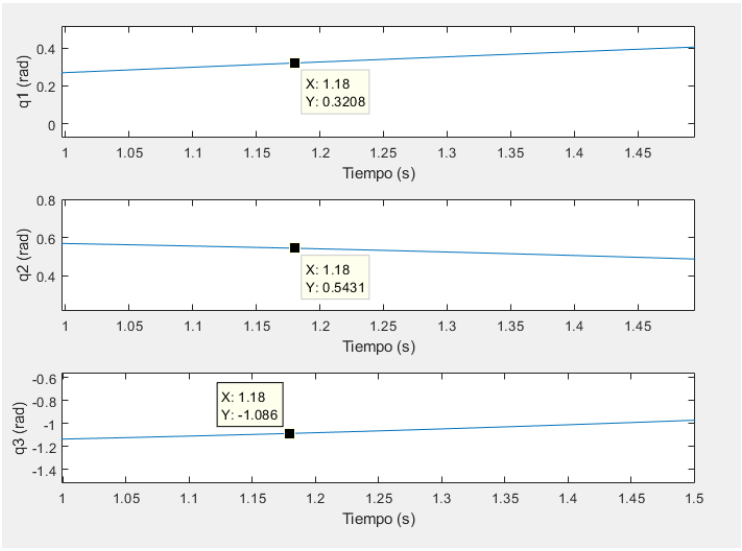
El robot va incrementando el ángulo de la base para moverse sobre el eje y. Los demás ángulos se mantienen constantes hasta que el robot se empiece a alejar de su posición en x o z para mantenerlo en el mismo valor. El siguiente tiempo escogido será en t=0.6s.




```
q1=0.15379129 rad
q2=0.60551414 rad
q3=-1.21102833 rad
tiempo actual=0.60
```

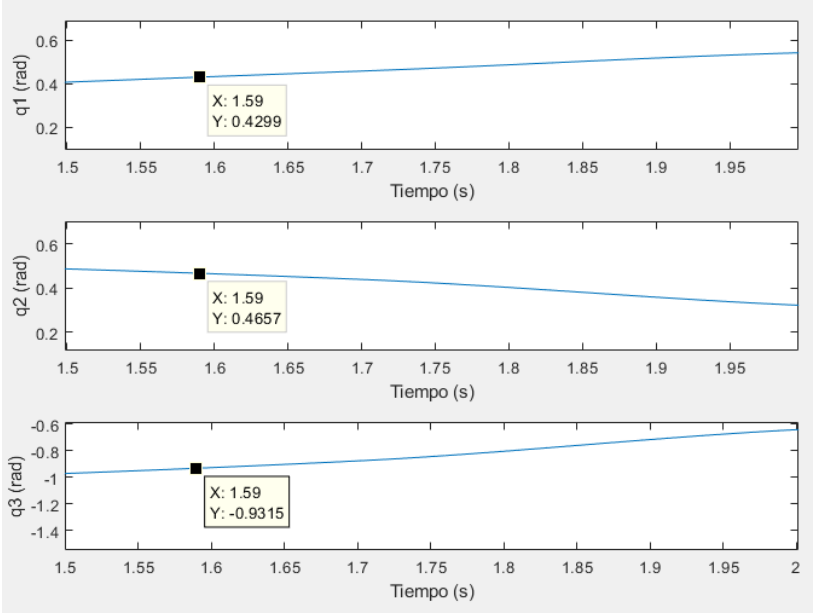
Figura 6-5. Valores para t=0.6s

Los ángulos del codo y el hombro se empiezan a mover debido a que, como se muestra en la figura 6-2, el robot se ha desplazado del eje X cuando debería haberse mantenido constante. Con esto vamos a ver el resto de tramos hasta que se llegue al tiempo de duración pedido que son los 2 segundos.



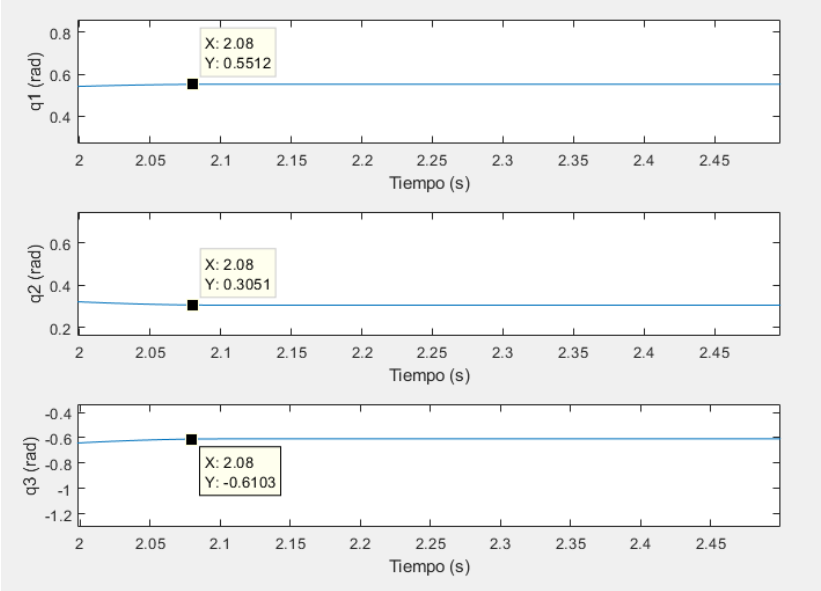

```
q1=0.32055804 rad
q2=0.54329671 rad
q3=-1.08659338 rad
tiempo actual=1.18
```

Figura 6-6. Valores para t=1.18s




```
q1=0.42986593 rad
q2=0.46573038 rad
q3=-0.93146076 rad
tiempo actual=1.59
```

Figura 6-7. Valores para t=1.59s




```
q1=0.55133028 rad
q2=0.30493330 rad
q3=-0.60986661 rad
tiempo actual=2.08
```

Figura 6-8. Valores para t=2.08s

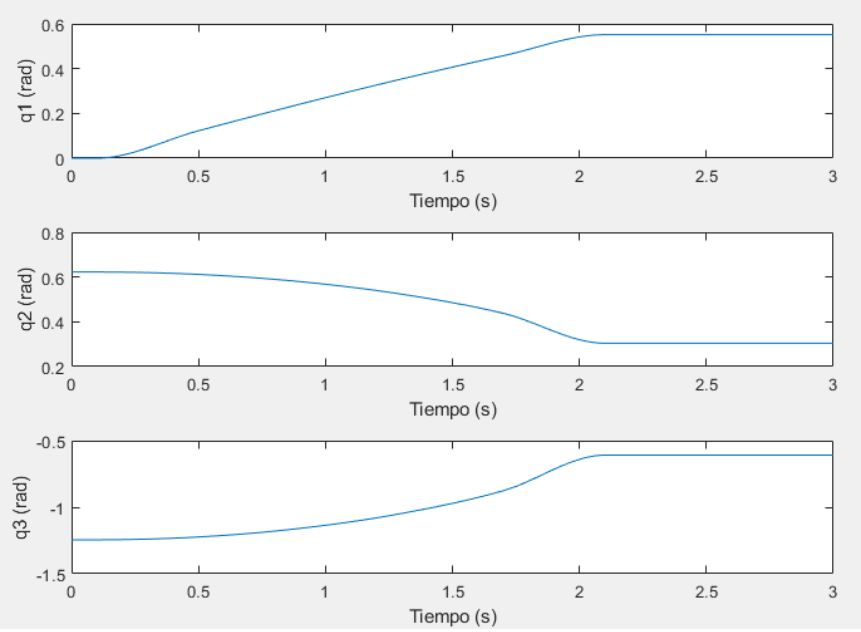
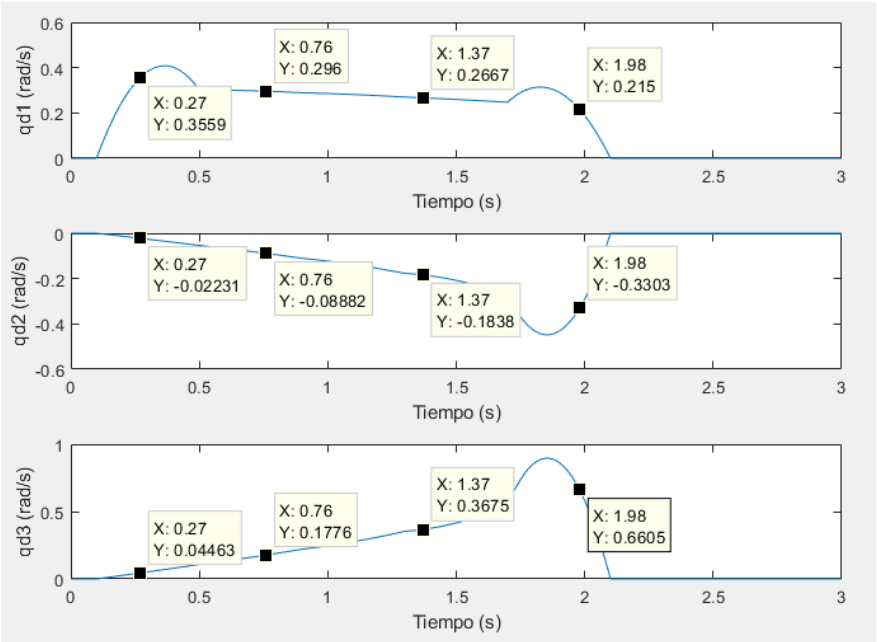


Figura 6-9. Gráfica de los ángulos obtenidos en la simulación

Como se ha observado, los valores obtenidos en Arduino no difieren mucho de los valores de la simulación, por lo que el generador de trayectorias funcionará correctamente y se podrá comprobar visualmente midiendo los desplazamientos reales del robot. Como también controlamos las velocidades[2], vamos a comprobar que estas también se cumplen.




```
velocidad q1=0.35703468 rad/s
velocidad q2 =-0.02244752 rad/s
velocidad q3 =0.04489504 rad/s
tiempo actual=0.27

velocidad q1=0.29609932 rad/s
velocidad q2 =-0.08852411 rad/s
velocidad q3 =0.17704820 rad/s
tiempo actual=0.76

velocidad q1=0.26692674 rad/s
velocidad q2 =-0.18325765 rad/s
velocidad q3 =0.36651551 rad/s
tiempo actual=1.37

velocidad q1=0.22135248 rad/s
velocidad q2 =-0.33941578 rad/s
velocidad q3 =0.67883148 rad/s
tiempo actual=1.98
```

Figura 6-10. Valores de la velocidad calculados durante el generador de trayectoria.

[2] En Arduino trabajamos con valores en microsegundos. Así que los resultados de las velocidades estarán en rad/ms por lo que habrá que pasarlos a segundos.

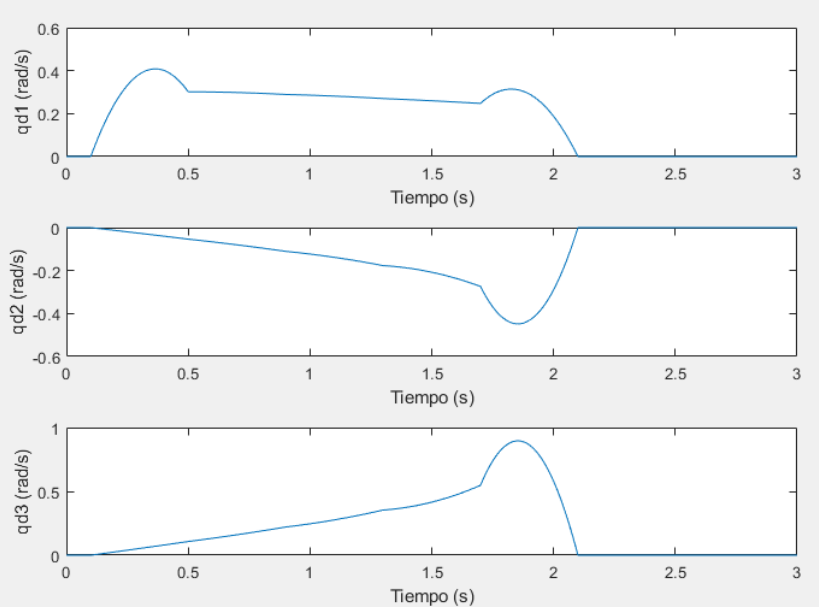


Figura 6-11. Gráfica de las velocidades angulares obtenidas en la simulación

6.1.2 Modo mando

Para comprobar que funcione correctamente vamos a hacer que el Arduino nos envié los valores por puerto serie de las variables que va cambiando. Así podremos observar cómo cambian los valores y ver los puntos máximos a los que es capaz de llegar.

```
q1: 80°
q1: 70°
q1: 80°
q1: 90°
q2: 150°
q2: 140°
q2: 130°
q2: 140°
q2: 150°
q3: 125°
q3: 135°
q3: 145°
q3: 135°
q3: 125°
```

Figura 6-12. Valores obtenidos en Arduino al controlar los ángulos con el mando

6.1.3 Modo cartesiano

Al igual que en el modo mando, veremos si los valores que nos salen son los deseados. En este caso se puede ver más visualmente con el robot si se encuentra en la posición que está escrita por pantalla. Este modo se puede usar para comprobar si el modelo cinemático inverso funciona, ya que enviaremos la posición a la que queremos movernos y nos devolverá el grado que se deben de mover las articulaciones.

```
X: 0.14
X: 0.15
Y: -0.01
Y: 0.00
Y: 0.01
X: 0.14
Z: 0.02
Z: 0.03
Z: 0.04
Z: 0.03
Z: 0.02
```

Figura 6-13. Valores obtenidos en Arduino al controlar la posición de la pinza en los ejes cartesiano con el mando

6.2 Futuras mejoras

El proyecto está pensado para fortalecer los conocimientos de los alumnos en el control y la programación de brazos manipuladores de una forma económica. Se escogió la madera por ese motivo para la construcción del robot, sin embargo, esto ocasionó que la parte del control dinámico fuera innecesaria, debido a que la madera es tan liviana que cualquier cambio dinámico no se notaría. Además, los servomotores tienen integrados un control que ya hacen el control dinámico por lo que aun no siendo un material liviano tampoco podríamos hacerlo. Por ello una de las futuras mejoras que se ha pensado, es buscar otro material barato y más pesado, con el que se pueda notar la dinámica del robot, y usar motores DC para aprender a controlarlos. Esta mejora abarcaría más de los conocimientos dados sobre el control de robots manipuladores. Como complemento a esta mejora, se podría usar un sensor táctil para la pinza que al detectar un objeto dejara de cerrarla para poder probar la dinámica del robot, viendo si es capaz de coger objetos y mantener la posición.

Otra posible mejora es el uso de los ángulos en el generador de trayectoria como global. Al hacer esto ocupamos mucha memoria ya que son varios ángulos y nos limita a usar cierta cantidad de muestras en el generador ya que la memoria se llenaría. Por eso, otra posible mejora sería usar la memoria EPROM que tiene cara microcontrolador para guardar ese tipo de variables y dejarlas como locales. Reduciendo la memoria del microcontrolador y aprendiendo a usar más recursos de los microcontroladores.

REFERENCIAS

[1] https://es.wikipedia.org/wiki/Cinem%C3%A1tica_inversa

[2] Fu, K.S.; González, R.C. y Lee, C.S.G. (1987). Robotics: Control, Sensing, Vision, and Intelligence. *McGraw-Hill*.

[3] http://proton.ucting.udg.mx/materias/robotica/r 166/r78/r78.htm

[4] Martínez A. G. M.; Jáquez O. S. A.; Rivera M. J. y Sandoval R. R. "Diseño propio y Construcción de un Brazo Robótico de 5 GDL".

[5] http://www.aurova.ua.es/robolab/ejsSystem.ht m

[6] MITSUBISHI (1996). Instruction Manual Industrial Micro-Robot System Model RV-M1. *Mitsubishi Electric Corporation. Tokio JAPON*.

[7] https://es.wikipedia.org/wiki/Cinem%C3%A1tica_directa

[8] Barrientos, A; Peñín, L.F; Balaguer, C. & Aracil, R. (2007)"Fundamentos de Robótica". 2ª Ed. *McGrawHill*

[9] Lung-Wen Tsai. "Robot Analysis: the mechanics of serial and parallel manipulators". *Wiley Interscience*.

[10] Murray, Li & Sastry. "A mathematical introduction to robotic manipulation". *CRC*.

[11] http://programacionextrema.es/2017/06/29/cinematica-del-robot/

[12] https://maferstech.com/robotica-algoritmo-de-denavit-hartenberg-caso-de-estudio-ssrms/

[13] https://www.ciencia-explicada.com/2013/02/parametrizacion-denavit-hartenberg-para.html

[14] https://www.areatecnologia.com/electricidad/servomotor.html

[15] http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

[16] http://www.ordenadores-y-portatiles.com/joystick.html

[17] https://es.wikipedia.org/wiki/Arduino_Uno

[18] Molina M. Diseño. (2015) Construcción del Prototipo de un Brazo Robótico con Tres Grados de Libertad, como Objeto de Estudio. *INGENIARE*.; 18(1): 87-94.

[19] https://www.arduino.cc/

[20] Chaos D, Chacon JnJ, Lopez-Orozco JA, Dormido S. (2013). Virtual and remote robotic laboratory using ejs, matlab and labview. *Sensors*.;13(2): 2595–2612.

GLOSARIO

MTH: Matriz de Transformación Homogénea

MCI: Modelo Cinemático Inverso

GDL: Grados de Libertad

PWM: Pulse Width Modulation