UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"DISEÑO E IMPLEMENTACIÓN DE UN ROBOT EDUCATIVO DE BAJO COSTE CON APLICACIONES VISUALES"

TRABAJO FIN DE GRADO

JULIO 2021

AUTORA: Delia Martínez Gálvez

DIRECTOR: Arturo Gil Aparicio

ÍNDICE

INDICE ILUSTRACIONES

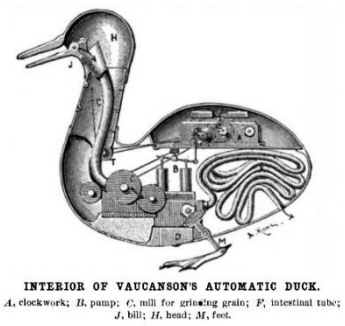
1. INTRODUCCIÓN

1.1. Antecedentes, Robótica

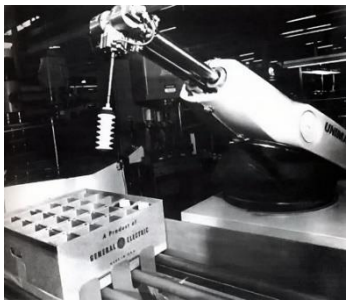
La RAE (Real Academia Española) define la palabra robot como *"máquina o ingenio electrónico programable que es capaz de manipular objetos y realizar diversas operaciones"*. La palabra robot se acuñó en 1921 cuando el escritor Karel Čapek utilizó en su obra R.U.R. (Robots Universales Rossum) la palabra checa "robota", la cual significa trabajos forzados o servidumbre. Se tradujo al inglés en 1923 como la conocemos hoy en día, "robot". En los años 50 Isaac Asimov la popularizó en un conjunto de relatos titulados "I, Robot".

Por otra parte el ingeniero español Leonardo Torres Quevedo acuñó la palabra "automática" para dar nombre a la acción de hacer automáticas las tareas. Según la RAE (Real Academia Española) se define "automático" como *"un mecanismo o un aparato que funciona en todo o en parte por sí solo"*.

A lo largo de la historia se han creado una gran variedad de robots para diferentes tareas, desde un pato mecánico capaz de comer y mover las alas (1738-Digesting Duck), primer robot programable industrial (1961- Unimate), primer robot con 6 ejes (1973-Famulus), hasta un robot humanoide capaz de simular expresiones, reconocer y recordar caras (2015- Sophia).



1.Digesting Duck (fuente: Wikipedia)



2.Unimate (fuente: Wikipedia)

3.Sophia (fuente: Wikipedia)

En la actualidad los robots se usan para funciones muy diversas desde labores domésticas hasta la realización de tareas peligrosas.

1.1.1. Tipos de robots

- Según su cronología

  - 1ª Generación, son robot con un simple sistema de control, de secuencia variable o fija.

  - 2ª Generación, son robots que aprenden una serie de movimientos que un operador ha ejecutado previamente y este ha memorizado.

  - 3ª Generación, son robot con sensores, un ordenador ejecuta unas órdenes en un programa que este envía al robot para que los realice.

- Según su estructura, base estática

  - Manipuladores tipo serie:

Robot cartesiano: cuenta con 3 articulaciones prismáticas y coincidentes con los ejes cartesianos. Se suele utilizar para trabajos de "pick and place".

Robot cilíndrico: sus ejes forman un sistema de coordenadas cilíndricas. Se suele utilizar para ensamblaje o soldaduras por puntos.

Robot esférico o polar: sus ejes forman un sistema de coordenadas polares. Se suele utilizar para manipulación en máquinas herramientas, varios tipos de soldaduras. Un ejemplo sería el robot industrial nombrado antes, Unimate.

Robot SCARA: tiene dos articulaciones rotatorias paralelas. Se utiliza para trabajos de "pick and place".

- Robot paralelo: son cadenas cinemáticas cerradas cuyo extremo móvil, está conectado a la base con varias cadenas cinemáticas en serie independientes. Tiene como aplicaciones la transferencia de piezas a alta velocidad, simuladores de vuelo, maquinado de piezas, etc.

4.Robot cartesiano (fuente: www.intraautomationsl.com)

5.Robot paralelo (fuente: www.abb.com)

6.Robot SCARA (fuente: www.epson.com)

- Según su estructura, móviles
  - Robots móviles, estos son capaces de desplazarse por el entorno mientras realizan una tarea o para realizarla, pueden ser vehículos móviles, caminantes, trepadores, humanoides (estos actualmente se utilizan solo en experimentación), submarinos o unmanned aerial vehicles.

8.Robot trepador (fuente: www.igus.com)

7.Robot humanoide (fuente: www.efe.com)

Este proyecto se centra en los robots de base estática, concretamente en los brazos robóticos. Un brazo robótico consta de varias partes interconectadas mediante articulaciones que dotan al robot de una serie de movimientos tanto rotacionales como de desplazamiento lineal y movimiento traslacional.

1.2. Objetivos

Este Trabajo de Final de Grado tendrá un doble objetivo.

- En la fase de desarrollo y construcción del brazo robótico de bajo coste, el objetivo será conocer las características y uso de la plataforma Arduino; conocer el funcionamiento de motores de paso a paso, motores DC, servomotores y su control mediante el programa para ordenador; conocer las posibilidades que brinda MATLAB para diseñar algunas funciones para el control del brazo robótico dentro de este entorno; valorar qué tipo de motor utilizar para dotar de movimiento al brazo robótico; así como diseñar la conexiones entre componentes y elegir los materiales necesarios para llevarlo a cabo.

- Cuando ya esté finalizado, el objetivo del brazo robótico será para uso educativo en un aula de prácticas, y tendrá aplicaciones visuales, de seguimiento de objetos y búsqueda activa de los mismos. Por lo que será necesario conocer y saber utilizar un tipo de marca con las que el robot pueda reconocer los objetos a través de una cámara.

1.3. Análisis del problema y soluciones

Uno de los objetivos principales del TFG es que el brazo robótico sea de bajo coste ya que en caso de rotura o desgaste de una pieza en el aula de prácticas, no sea costosa su reparación o sustitución. Por esta razón la opción elegida para construir las piezas del brazo robótico es imprimirlas en una impresora 3D, ya que el material empleado es PLA y este tiene un bajo coste y es fácil de conseguir.

Una de las características principales de un brazo robótico son sus articulaciones con las que consigue moverse el brazo, para ello se integran unos motores en las articulaciones y se consigue la rotación entre piezas. Actualmente existen varias opciones con las que dotar de movimiento a esas articulaciones, siguiendo la condición del bajo coste, se ha elegido el servomotor ya que se comporta como un motor paso a paso, sin necesidad de un sensor externo que nos indique en qué posición se encuentra, su control

será más sencillo mediante MATLAB, además de trabajar con corriente continua y tener un tamaño reducido.

Para el control de estos servomotores se ha elegido un microcontrolador de código abierto, Arduino Uno. Esta placa es de bajo coste en comparación con otros controladores. Cuenta con 14 pines digitales de entrada y salida que permite enviar la señal a los 7 servos con los que contará el brazo robótico, esta señal será enviada desde el ordenador mediante un cable USB, por lo que facilita también uso.

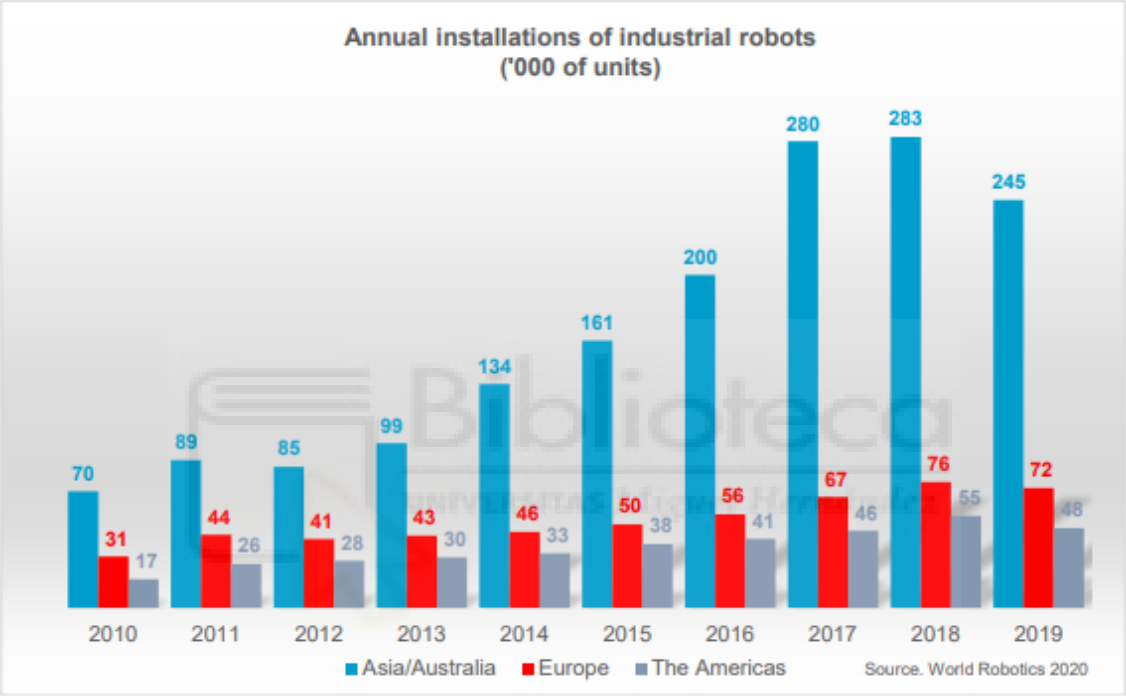
Para transmitir esas órdenes mediante USB al microcontrolador contaremos con el programa de ordenador MATLAB, el cual cuenta con la toolbox ARTE, creada por el profesor Arturo Gil Aparicio de la Universidad Miguel Hernández. Esta toolbox está enfocada para la manipulación de brazos robóticos.

Para el procesamiento de las imágenes captadas por la cámara, instalada en el extremo del robot, se utilizará un programa para la detección de marcas desarrollado en OpenCV usando la librería ArUco. Adaptado para su uso en MATLAB por Luis Miguel Jiménez García de la Universidad Miguel Hernández.

2. ESTADO DEL ARTE

2.1. Robots de educación

Acorde a la información presentada por la IFR (International Federation of Robotics), la instalación de robots industriales está creciendo notablemente año tras año. Sobre todo en la parte de Asia, desde 2013 ha sido el mayor mercado mundial de robots industriales, aportando el 38% de la instalaciones totales entre 2017 y 2018. Solo en 2019 se instalaron más de 140 mil unidades.



9. Instalaciones anuales de robots industriales (fuente: www.ifr.org)

Debido a este crecimiento es necesario que los ingenieros se formen en el conocimiento y control de estos robots industriales. Para ello es necesario que las universidades formen a los futuros ingenieros en estos campos, ya que serán los encargados de diseñar y programar estos robots.

Para que el proceso de formación sea correcto y completo, es necesario que no solo se aprenda la teoría de los robots, sino que además tengan una formación práctica para terminar de comprender la materia y poner en prácticas soluciones a posibles conflictos o problemas.

Esta necesidad de formar a los ingenieros con casos prácticos acarrea una serie de inconvenientes, ya que los robots industriales tienen un alto coste. Para controlar este robot industrial es necesario un software a la altura, esto implica que el software también será caro y requerirá de una curva de aprendizaje lenta. Debido a esta curva de aprendizaje de los alumnos es necesario que tenga una celda de seguridad tanto para el robot como para los alumnos.

La universidad Miguel Hernandez concretamente en el caso de ISA UMH, cuentan con un robot industrial Fanuc, programable mediante paletera. Como ya se ha comentado este robot requiere de una celda de seguridad y en caso de rotura o mal uso es costosa su reparación.

A instancia de los directores del presente proyecto, se planteaban formas de contar con una plataforma que pudiera ser fácilmente integrada con MATLAB, específicamente con el Toolbox ARTE. Como funciones adicionales pudiera capturar imágenes en tiempo real y procesarlas. Ser intrínsecamente seguro para el uso en el laboratorio por los estudiantes. Además de un uso sencillo para suavizar la curva de aprendizaje.

2.2. Robots descartados

Los robots de código abierto son robots cuyos documentos, planos, código fuente o planos han sido publicados o liberados como modelos de código abierto. Para este proyecto se ha decidido utilizar un brazo robótico de código abierto para poder centrarnos en la programación del mismo. Aun así no se ha descartado el realizar cambios al diseño utilizado en función de las necesidades del proyecto.

En una búsqueda de brazos robóticos de código abierto son 6 grados de libertad se redujo a los siguientes robots:

2.2.1. The Annin Robotics AR3[1]

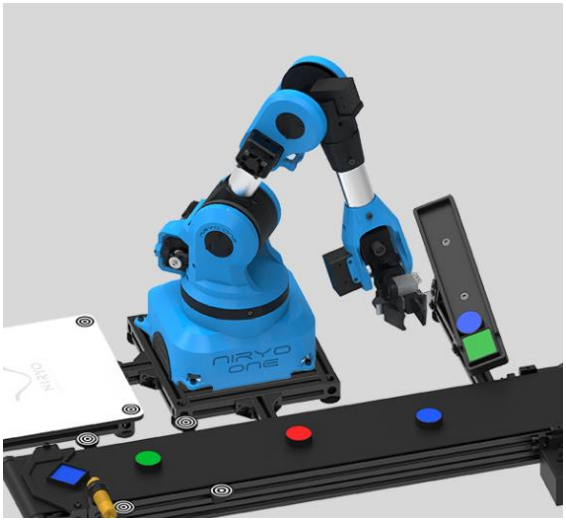
10.AR3 de Annin Robotics (fuente: www.anninrobotics.com)

Este es un brazo robótico de 6 grados de libertad y código abierto, está compuesto de piezas impresas en 3D y piezas de aluminio. Para mover sus articulaciones utiliza motores paso a paso con sus correspondientes controladores, para poder controlarlos utiliza una placa Arduino Mega. Este robot tiene un aspecto más profesional e industrial y lo hacía interesante como propuesta para este proyecto.

Se ha descartado este diseño por las piezas de aluminio, ya que serían más difíciles de conseguir, aunque estén disponibles en la página oficial para comprar el kit completo, no es el objetivo del proyecto. Otra razón es que tiene diseño con unas características que excede las necesidades del proyecto. El diseño cuenta con una caja a parte donde se encuentran algunas conexiones y útiles necesarios para el robot, provocando que el robot necesite más espacio, algo que no nos conviene en un aula de prácticas.

[1] https://www.anninrobotics.com/

2.2.2. Niryo One[2]



11. Nyrio One (fuente: www.niryo.com/niryo-one)

Niryo es un brazo robótico de origen francés, creado con una finalidad educativa y de código abierto. En la página oficial se puede comprar el kit completo para montar el brazo robótico, además de los accesorios para diferentes tareas como visión por una cámara, ventosa de vacío o pequeñas funciones industriales. También se pueden encontrar en la página web tutoriales de cómo montar cada parte del robot y manuales para su correcto uso. Su refinado diseño y la multitud de funciones disponibles lo hacía candidato para este proyecto.

Al final este robot fue descartado, ya que sería complejo conseguir y adaptar las piezas de aluminio del brazo y comprar el kit completo no sería el objetivo del proyecto e incrementaría el coste.

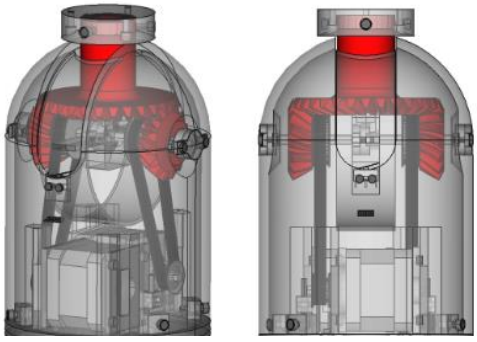
[2] https://niryo.com/niryo-one/

2.2.3. Thor[3]

12.Thor (fuente:www.hackaday.io/project/12989-thor)

Thor es un brazo robótico de código abierto con 6 grados de libertad. Este robot es el que más se asemeja a los vendidos para tareas industriales (yaw-roll-roll-yaw-roll-yaw), puede levantar objetos de hasta 750 gramos. Comenzó siendo un proyecto final de carrera que hasta hoy su diseñador ha ido mejorando. Está impreso en 3D y controla los motores paso a paso con una placa Arduino Mega. En su página web se puede encontrar la lista de componentes y un video explicativo de cómo montar el brazo robótico.

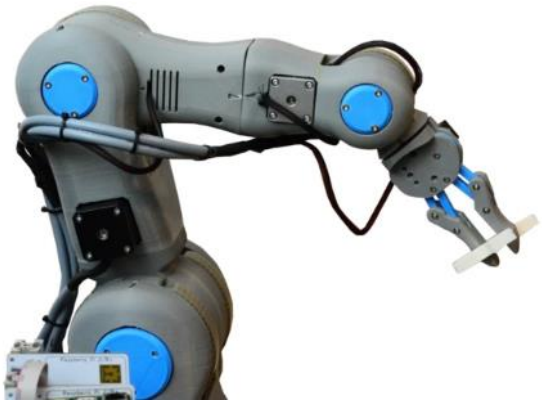
Se ha descartado este robot por tener un montaje bastante complejo ya que en su interior cuenta con mecanismos de engranajes y correas que para su uso en el laboratorio de prácticas dificultaría su ajuste, o en caso de rotura, el arreglo. Además, al tener ocultos los mecanismos los alumnos no podrían apreciar el interesante funcionamiento de los engranajes interiores.



13. Mecanismo interior Thor (fuente: www.hackaday.io/project/12989-thor)

[3] https://hackaday.io/project/12989-thor#menu-description

2.2.4. RBX1 de Roboteurs[4]



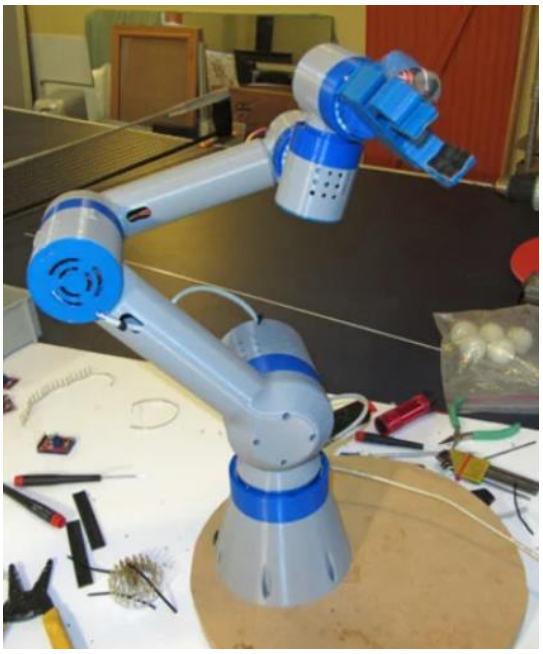
14.RBX1 (fuente: www.roboteurs.com)

Este brazo robótico está construido íntegramente con componentes impresos en 3D. Es necesario disponer de una impresora 3D para imprimir las piezas y en la página web puedes comprar todos los componentes restantes. Funciona con 7 motores paso a paso, un servomotor para la pinza del extremo y está controlado con una Raspberry Pi.

Este brazo robótico fue descartado por tener un mecanismo con correas en su interior, al igual que el robot Thor, que complica su construcción y cuenta con numerosas piezas que en caso de rotura sería complejo acceder a ellas.

[4] https://roboteurs.com/

2.2.5. WE-R2.4 [5]



15.WE-R2.4 (fuente: www.thingiverse.com/thing:3327968)

Este brazo robótico sigue en constante desarrollo, cuenta con 6 ejes con motores paso a paso con placas Pololu controladas mediante un firmware personalizado. En su página web se puede encontrar textos explicativos de cómo montar el robot y videos con pruebas de movimiento.

Su diseño sencillo y fácil construcción lo hizo candidato para este proyecto, además que tuviera una estructura tan estrecha facilitaba sus movimientos y conseguía más rango de actividad a la vez que reducía su peso.

Aunque este robot se ajusta a los objetivos del proyecto, se descartó por utilizar motores paso a paso y no servomotores, que sería la opción ideal.

[5] https://www.thingiverse.com/thing:3327968

3. DISEÑO DE UN ROBOT DE BAJO COSTE

3.1. Impresora 3D

En 1976 se inventó la primera impresora de inyección de tinta, pero no fue hasta 1992 que se empezó a fabricar el primer prototipo de impresora 3D. Este primer prototipo consistía en un láser UV que iba solidificando un fotopolímero capa a capa hasta formar piezas complejas, pero tenían algunas imperfecciones. Actualmente las impresoras 3D son máquinas capaces de crear diseños en 3D hechos a ordenador.

Las impresoras 3D se podrían clasificar en 3 categorías dependiendo del modo en que crea las figuras 3D:

- Impresoras 3D de tinta: estas impresoras usan una tinta aglomerante para espesar un polvo que general mente puede ser a base de celulosa o escayola. Al utilizar este tipo de tinta se pueden emplear diferentes colores. Este método permite crear figuras o piezas de forma más rápida y económica pero las piezas serán más frágiles.

- Impresoras 3D láser: consiste en una base que se sumerge en resina en estado líquido y va saliendo lentamente del recipiente mientras el láser va solidificando la base capa a capa.

- Impresora 3D de inyección de polímero: estas impresoras consisten en una o más cabezales de extrusión caliente por el que fluye un filamento fundido de algún tipo de polímero, ABS, PLA, Nailon, TUVALU, etc. Este filamento se va depositando capa a capa en una base, normalmente, de cristal. Aunque es

17.Impresora 3D láser (fuente: www.3r3dtm.com)

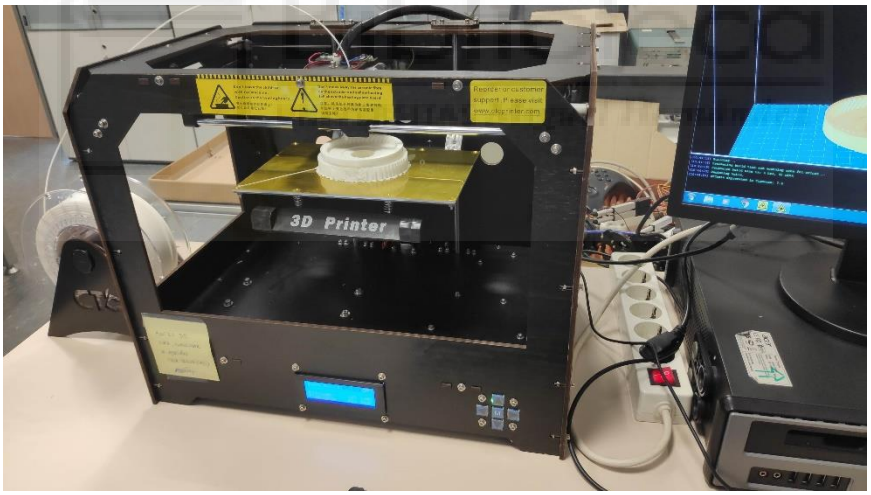
16.Impresora 3D polímero (fuente: www.amazon.com)

más costoso que la impresión por láser o de tinta, las piezas son más resistentes y pueden ser pulidas tras la impresión.

En este proyecto se ha utilizado una impresora 3D de inyección por la posibilidad de la posterior modificación de las piezas (puliéndolas). Además de su fácil manejo y accesibilidad ya que en el campus de la Universidad Miguel Hernández disponemos de varias.

3.1.1. Impresora 3D CTC

Esta impresora cuenta con un doble extrusor para poder tener dos filamentos preparados para impresión. Puede cargar piezas para su impresión desde una tarjeta SD, ya que cuenta con un lector de tarjetas SD, o bien conectando la impresora con un cable USB al ordenador. Como el resto de impresoras, esta cuenta con motores paso a paso, controlados mediante una placa Arduino, para que el extrusor se mueva en los ejes x e y para cubrir todo el plano horizontal.



18.Impresora 3D CTC (fuente: propia)

El funcionamiento de la impresora es el siguiente. El rollo de filamento PLA, del color que sea necesario, se coloca en un soporte para que vaya girando conforme la impresora va necesitando más filamento. Ese filamento pasa por un extrusor que lo calienta, a una determinada temperatura programable, y mientras se derrite el filamento lo va depositando en una "cama" o plataforma, a una temperatura indicada previamente, dibujando la forma que tenga esa capa de la pieza. Al final la impresora va formando la pieza capa a capa, una sobre otra, hasta conseguir la forma deseada.

3.1.2. Filamento PLA

19.Filamento PLA (fuente. www.impresoras3d.com)

En este proyecto se ha utilizado como material para imprimir en 3D las piezas un filamento de PLA. El PLA (ácido poliláctico) es un tipo de polímero biodegradable que se obtiene a partir del ácido láctico. Este ácido se puede fermentar desde cultivos como el maíz. Esta característica hace que tenga un bajo coste. Este filamento se funde a temperaturas bajas, entre 180ºC y 220ºC, su temperatura de transición vítrea, es decir cuando pasa de estado duro y relativamente quebradizo a viscoso, esta entre 60ºC y 65ºC, esto lo convierte en un material muy útil. En cambio tiene una alta fricción por lo que el extrusor podría quedar obstruido.

3.1.3. ReplicatorG

20.Logo ReplicatorG (fuente: wwww.oshl.edu.umh.es)

Para este proyecto se ha utilizado el software ReplicatorG para enviar desde el ordenador la información necesaria a la impresora mediante cable USB. Es un software libre para impresoras 3D que cuenta con una interfaz sencilla. A este programa se le carga una pieza en archivo con extensión ".stl". Permite mover o rotar la pieza en 3D para situarla sobre la plataforma, generar un efecto espejo o cambiarle la escala.

Cuando hemos cargado la pieza se genera el "GCode", este archivo contiene un código fundamental para el funcionamiento de las impresoras 3D. Consiste en un listado de instrucciones sencillas que debe transmitir la placa Arduino a los motores, con los desplazamientos en los ejes x, y, z para conseguir imprimir cada capa de la pieza. Este archivo "GCode" también contiene las temperaturas que debe alcanzar el extrusor o la plataforma para poder imprimir correctamente. Generalmente cada letra de este código tiene un significado:

Letra	Significado
G	Movimiento
X	Distancia horizontal
Y	Distancia vertical
Z	Profundidad
F	Tasa de alimentación
S	Velocidad del eje
T	Selección de herramienta
M	Múltiples funciones
I y J7	Centro del arco
R	Radio del arco

En este caso para general el "GCode" se ha configurado los siguientes parámetros:

- La calibración "Thing-O-Matic" que divide el objeto en 3D en cientos o miles de capas que va dibujando una tras otra. Se ha utilizado el extrusor derecho. En algunas piezas se ha usado un soporte que crea la propia impresora para las partes voladizas de la figura 3D. La pieza tendrá un 10 % de relleno en las partes sólidas. Cada capa tiene 0,27mm de altura, una velocidad de alimentación de 35 mm/s y una temperatura del extrusor de 190ºC. En este caso el programa no deja configurar la temperatura de la plataforma donde se imprime la pieza antes de generar el "GCode", por lo que es necesario cambiarla después de generar el código, de 110ºC a 60ºC. Para ello es necesario cambiar la línea *M109 S110 T0 (set HBP temperature)* por *M109 S60 T0 (set HBP temperature)*.

- En cuanto al filamento de PLA que se ha utilizado tiene 1,75mm de diámetro.
- Y el extrusor que derrite el PLA de esta impresora 3D tiene 0,4 mm de diámetro.

3.2. Modelo 3D del brazo robótico

Como ya hemos comentado anteriormente, se hizo una búsqueda de un diseño en 3D de código abierto para el brazo robótico de este proyecto. Tras preseleccionar algunos diseños y finalmente descartar los que no se ajustaban a las necesidades que buscábamos elegimos un diseño.

La página *créate.arduino*[6] es una plataforma para explorar y crear proyectos basados en Arduino. Esta página está impulsada por los servicios web de Amazon. Dentro de esta página cuentan con un apartado, *Arduino Project Hub*, donde se pueden encontrar diferentes proyectos utilizando Arduino creados por usuarios de la página.

Otra página donde este usuario subió parte del proyecto es *hackaday.io*[7] la cual es una comunidad pensada para compartir proyectos y aprender sobre ellos.

21.Logo Arduino.create (fuente: www.Arduino.create.cc)

22.Logo hackaday.io (fuente: www. hackaday.io)

En combinación de estas dos páginas el usuario Danny Van den Heuvel publicó el diseño del brazo robótico de 6 grados de libertad que vamos a utilizar en el proyecto.

[6] https://create.arduino.cc/projecthub
[7] https://hackaday.io/

Se decidió utilizar este diseño por ser compacto y estar adaptado para el uso de servomotores controlados por una placa Arduino directamente, además de su reducido tamaño, apropiado para un aula de prácticas. Aunque el autor del diseño también incluyó una paletera personalizada para controlar las articulaciones del brazo robótico, para este proyecto se ha utilizado solo el diseño en 3D de las piezas.

23.Modelo 3D Robot (fuente: www.hackaday.io)

El diseño también tiene una excepción, ya que la base cuenta con una reductora, la cual descarta el uso de un servomotor para esta articulación y nos condiciona a usar un motor paso a paso.

3.3. Cinemática Directa

La cinemática directa de un robot nos dice, conociendo los ángulos de un robot, dice en qué posición y orientación se encuentra el efector final del brazo robótico respecto a la posición y orientación de su base. Para saber esto es necesario calcular los parámetros DH.

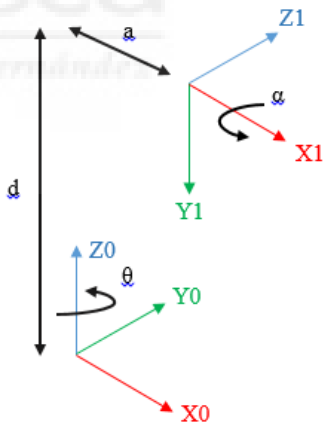
3.3.1. Parámetros Denavit-Hartenberg

En 1955 Jacques Denavit y Richard Hartenberg crearon estos 4 parámetros asociados para normalizar los marcos de coordenadas para los vínculos espaciales. Más tarde en 1981 Richard Paul comprobó y demostró que se pueden utilizar para el análisis cinemático de los sistemas robóticos.

Para colocar esto ejes en nuestro robot se tiene que seguir una serie de reglas, ya que el eje Z_i siempre se colocará alineado con el eje del movimiento de la articulación i+1. Después se colocará el vector X_i, el cual tiene que cumplir que sea paralelo al vector normal común, calculado como $X_n=Z_i \times Z_{i-1}$. Los signos de giro para los ejes se podrán calcular con un producto vectorial o una forma más práctica, ya que seguirán "la regla de la mano derecha" o "del sacacorchos", consiste en colocar el pulgar paralelo a la dirección y sentido del vector y con el resto de dedos envolver el vector, esto nos dará el sentido de giro de ese vector. El sistema de referencia de ultimo efector del brazo robótico se ha hecho coincidir con el sistema de referencia de la cámara USB que nos proporciona el programa para la detección de marcas ARUCO, para facilitar los cálculos.

Los 4 parámetros se calculan como:

- θ: ángulo de giro en el eje Z de origen.
- d: desplazamiento en el eje Z de origen.
- a: desplazamiento en el eje X de destino.
- α: ángulo de giro en el eje X de destino.

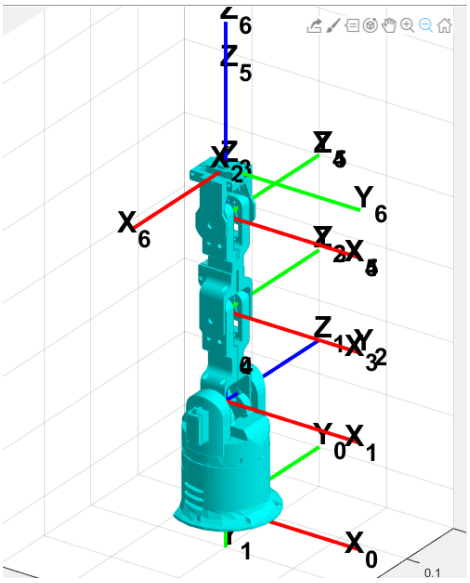


24.Ilustración parámetros DH

(fuente: propia)

Con estas medidas, para nuestro brazo robótico quedarían los siguientes parámetros DH.8

		θ	d	a	α
0→1	$^{0}A_1$	$q1$	0.146	0	$-\frac{\pi}{2}$
1→2	$^{1}A_2$	$q2-\frac{\pi}{2}$	0	0.123	0
2→3	$^{2}A_3$	$q3+\frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
3→4	$^{3}A_4$	$q4$	0.13	0	$-\frac{\pi}{2}$
4→5	$^{4}A_5$	$q5$	0	0	$\frac{\pi}{2}$
5→6	$^{5}A_6$	$q6-\frac{\pi}{2}$	0.065	0	0



25.Ejes referencia para parámetros DH

(fuente: propia)

Las matrices $^{i-1}A_i$ representan la transformación de coordenadas desde el sistema de coordenadas i-1 al i. La posición del efector final será:

$$A_i^{i-1} = \begin{pmatrix} cos\,\theta_i & -cos\,\alpha_i\,sin\,\theta_i & sin\,\alpha_i\,sin\,\theta_i & a_i\,cos\,\theta_i \\ sin\,\theta_i & cos\,\alpha_i\,sin\,\theta_i & -sin\,\alpha_i\,cos\,\theta_i & a_i\,sin\,\theta_i \\ 0 & sin\,\alpha_i & cos\,\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_6^0 = A_1^0 * A_2^1 * A_3^2 * A_4^3 * A_5^4 * A_6^5$$

3.4. Cinemática Inversa

La cinemática inversa de un robot nos proporciona los parámetros a los que tiene que estar cada articulación para que el efector final del robot llegue a una posición orientación determinada.

Para poder obtener los valores de las articulaciones es necesario conocer una matriz de posición y orientación del efector final:

$$T = \begin{pmatrix} nx & ax & ox & px \\ ny & ay & oy & py \\ nz & az & oz & pz \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$$

Siendo *R(n, a, o)* la matriz de rotación del sistema de coordenadas del efector final del brazo respecto de la base del robot, y *t(p)* la matriz de traslación o el desplazamiento del sistema de coordenadas del efector final respecto de la base en cada uno de los 3 ejes x, y, z. Para formar la matriz completa rellenamos con 0 y el factor de conversión para la escala que en este caso ser 1.

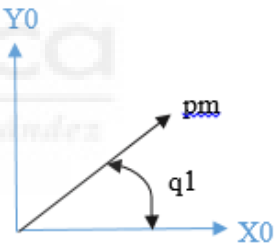
Partiendo de una T conocida y las medidas del robot, se calcula cada parámetro de cada articulación de forma geométrica:

- Primera articulación (vista desde arriba):

$$t = [px \quad py \quad pz] \quad W = [ox \quad oy \quad oz]$$

$$L6 = d(6) \qquad pm = t' - L6 * W$$

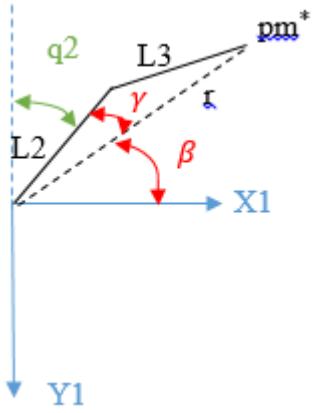
$$q1 = atan\frac{Y_{pm}}{X_{pm}} \qquad q1^* = q1 + \pi$$



26.Calculo q1 (fuente: propia)

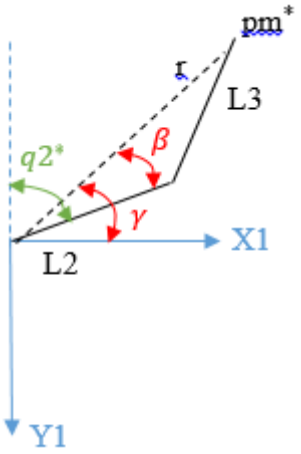
L6 es la distancia desde la muñeca del robot hasta la punta del mismo.

- Segunda articulación (codo arriba y codo abajo):



28.Calculo q2 codo arriba

(fuente: propia)



27.calculo q2 codo abajo*

(fuente: propia)

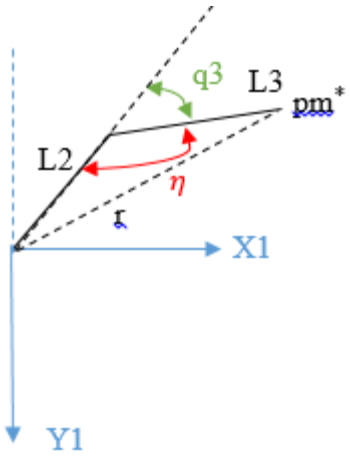
$$L2 = a(2) \quad L3 = d(4) \quad r = \sqrt{L2^2 + L3^2} \quad pm^* = (A_1^0)^{-1} * pm$$

$$\gamma = cos^{-1}\left(\frac{-L3^2 + L2^2 + r^2}{2 * L2 * r}\right) \quad \beta = tan^{-1}\left(\frac{Y_{pm*}}{X_{pm*}}\right)$$

$$q2 = \frac{\pi}{2} - \gamma - \beta \qquad q2^* = \frac{\pi}{2} + \gamma - \beta$$

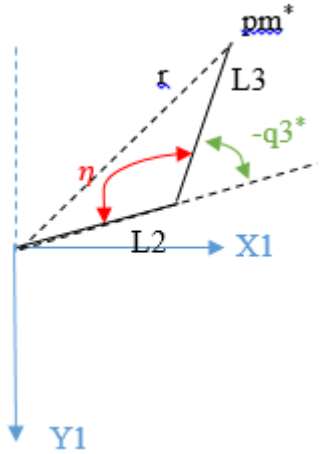
L2 y L3 son medidas del robot, las podemos obtener de los parámetros DH.

- Tercera articulación (codo arriba y codo abajo):



30.Calculo q3 codo arriba

(fuente: propia)



29.Calculo q3 codo abajo*

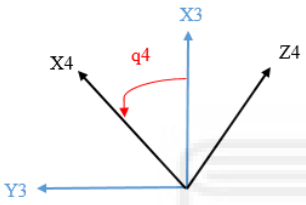
(fuente: propia)

$$\eta = cos^{-1}\left(\frac{-r^2 + L2^2 + L3^2}{2 * L2 * L3}\right)$$

$$q3 = \pi - \eta \quad q3^* = \eta - \pi$$

- Ultimas 3 articulaciones

Con las ecuaciones anteriores podemos conocer la posición de *q3*.El valor de la muñeca es independiente del valor de *q4*, *q5* y *q6*. Para calcular esto hay que colocarlos correctamente, *Z3* y *Z5* deben formar un plano del que *Z4* es perpendicular. Y *Z5* es paralelo a *Z6*, por tanto:



31.Calculo q4 (fuente: propia)

$$\overline{Z4} = \frac{\overline{Z3} * \overline{Z6}}{|\overline{Z3} * \overline{Z6}|}$$

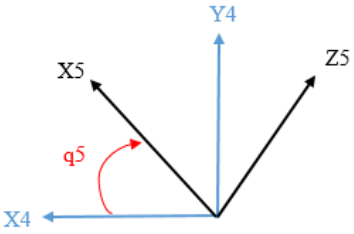
$$Z3 = A_3^0 \quad Z6 = T = [ox \quad oy \quad oz]$$

$$a = \overline{Z4}' * (-\overline{Y3}) \quad b = \overline{Z4}' * \overline{X3}$$

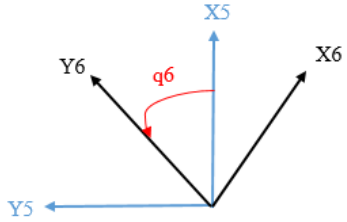
$$q4 = atan(\frac{b}{a})$$

$$a = \overline{Z5}' * \overline{Y4} \quad b = \overline{Z5}' * (-\overline{X4})$$

$$q5 = atan(\frac{b}{a})$$



32.Calculo q5 (fuente: propia)



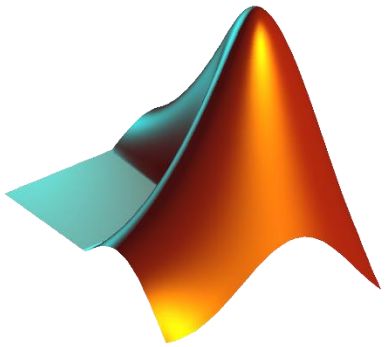
33.Calculo q6 (fuente: propia)

$$a = cos^{-1}(q6) = (-\overline{Y6}) * (-\overline{X5})$$

$$b = sin^{-1}(q6) = (-\overline{Y6}) * (-\overline{Y5})$$

$$q6 = atan(\frac{b}{a})$$

3.5. MATLAB



34.Logo MATLAB (fuente: www.matlab.mathworks.com)

Para el control por ordenador del brazo robótico se va a usar el programa MATLAB, abreviatura de MATtrix LABoratory. Este es un sistema de cómputo numérico el cual cuenta con su propio lenguaje de programación, lenguaje M. También ofrece un entorno de desarrollo integrado. MATLAB está disponible para plataformas Windows, macOS, Unix y GNU/Linux.

El lenguaje propio de MATLAB es interpretado y se puede ejecutar tanto desde un archivo script, archivos *.m, como en el entorno interactivo. Permiten operaciones de cálculo lambda, funciones, programación orientada a objetos, matrices y vectores. Y también posee herramientas y funciones para trabajar con datos en 3D y 2D.

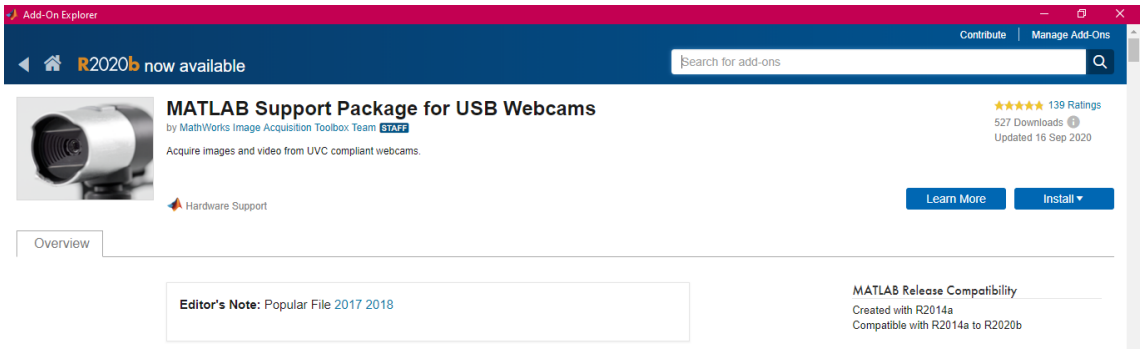
Este software se utiliza en cursos de matemáticas como en formación de ingeniería, también se emplea en centros de investigación para el desarrollo de productos tecnológicos.

3.5.1. Toolbox y packages

MATLAB cuenta con *Toolbox*, que son básicamente cajas de herramientas, las cuales cuentan con un conjunto de funciones que extienden las capacidades de MATLAB.

Para este proyecto es necesaria la instalación de los *Hardware Support Package*:

- *MATLAB Support Package for USB Webcam*[8]: Este *Toolbox*, nos proporcionará las herramientas necesarias para procesar las imágenes en directo captadas por la cámara USB (Compatible con MATLAB R2014a y posteriores).



35.Toolbox USB Webcam (fuente: propia)

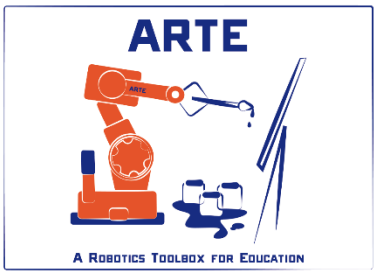
- *Legacy MATLAB and Simulink Support for Arduino*[9]: Este package nos facilitará unas funciones para que la placa Arduino Uno se comunique con MATLAB y se puedan controlar los servomotores. Ya no está recomendado el uso de esta versión, pero la nueva versión del paquete manda la señal a los servomotores con un rango de 0 a 1 (min. y máx.), mientras que esta versión utilizada en este proyecto, envía los datos con un rango de 0 a 180 grados, lo cual nos es más cómodo para su uso en las funciones del brazo robótico.

36.Toolbox Arduino (fuente: propia)

[8] https://es.mathworks.com/matlabcentral/fileexchange/45182-matlab-support-package-for-usb-webcams

[9] https://es.mathworks.com/matlabcentral/fileexchange/32374-legacy-matlab-and-simulink-support-for-arduino

3.5.2. ARTE[10]



37.Logo ARTE (fuente: www.arvc.umh.es/arte)

Una de las *Toolbox* de MATLAB que vamos a aprovechar es ARTE. Este *Toolbox* fue creado por el profesor de la Universidad Miguel Hernández Arturo Gil junto a los estudiantes de la asignatura robótica de 2012.

El acrónimo ARTE (*A Robotics Toolbox For Education*) significa que es una caja de herramientas de robótica para la educación. Este *Toolbox* está orientad a manipuladores robóticos. Las características principales de ARTE son:

- Representar visualmente el sistema Denavit-Hartenberg de un robot.
- Simular cualquier robot industrial dentro de MATLAB.
- Dispone de gráficos 3D de una gran cantidad de robots industriales y se pueden añadir más a la biblioteca de manera sencilla.
- Se puede graficar las fuerzas y pares de cada articulación. Y observar el trazado, velocidad y aceleración de un movimiento del robot.
- Cuenta también con un intérprete de MATLAB a RAPID que permite programar un robot usando código MALAB.
- En su página web se pueden encontrar videos explicativos y sesiones prácticas para el uso de esta librería.

[10] http://arvc.umh.es/arte/index_en.html

3.6. Arduino

38.Logo Arduino (fuente: www.arduino.cc)

Un microcontrolador es básicamente un circuito integrado programable, el cual es capaz de ejecutar unas instrucciones grabadas en su memoria. Consta de 3 principales unidades funcionales de un ordenador: memoria, periféricos de entrada/ salida y unidad central de procesamiento.

En 2005 se inició el proyecto *Arduino* como algo enfocado a estudiantes, ya que antes de eso usar microcontroladores era algo caro para estudiantes promedio. Arduino es una compañía desarrollo de software y hardware. También manufactura placas de desarrollo de hardware para poder crear dispositivos digitales que controlen objetos.

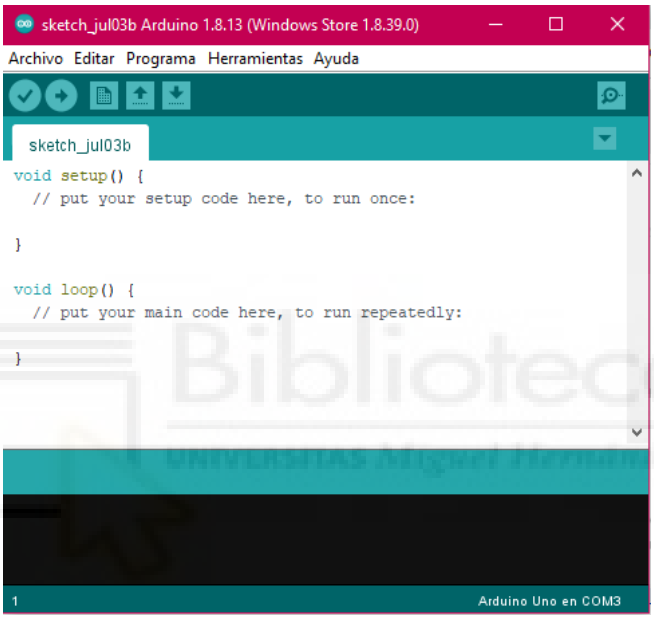
3.6.1. Placa Arduino Uno

En este proyecto vamos a utilizar una placa *Arduino Uno*, es una placa con 6 pines analógicos y 14 pines digitales, los cuales serían suficientes para controlas los 6 servomotores con los que contaría el brazo robótico del proyecto. Esta placa puede ser alimentada desde una batería externa de 9 voltios o por un cable USB desde un ordenador.

39.Arduino Uno (fuente: www.electronicaembajadores.com)

3.6.2. IDE Arduino

Para programar este dispositivo se utiliza el software de Arduino, este está compuesto por un entorno de desarrollo integrado (IDE) propio. Este lenguaje de programación de Arduino es una adaptación de C++. El propio entorno de desarrollo integrado nos facilita algunos ejemplos de códigos para el funcionamiento de Arduino. Antes de subir un código a la placa Arduino el propio IDE comprueba si tiene errores de escritura o en el código.



40.Ventana inicio IDE Arduino (fuente: propia)

3.7. Motor paso a paso

3.7.1. Funcionamiento

Los motores paso a paso (stepper) son dispositivos electromecánicos que transforman unos impulsos eléctricos en un desplazamiento angular. Su uso es ideal para situaciones movimientos que requieren mucha precisión. Son capaces de quedar fijos en una posición si una o más de las bobinas están recibiendo energía.

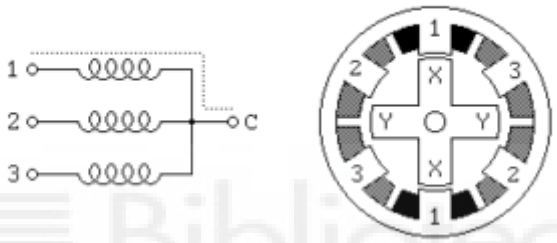
Su funcionamiento consiste básicamente un estator embobinado con material ferromagnético y un rotor que puede girar libremente dentro de ese estator. Con una

determinada secuencia que alimenta una bobina tras otra, genera que el rotor vaya girando dependiendo de la bobina o bobinas alimentadas.

3.7.2. Tipos de motores paso a paso

- Motores paso a paso de reluctancia variable

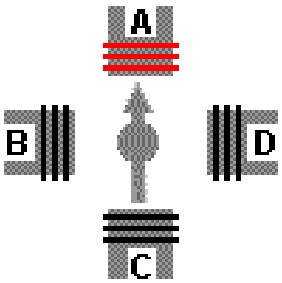
Este motor no usa un campo magnético permanente, por lo que este puede moverse sin limitación. Se usa generalmente para movimientos que no necesitan un gran par de fuerza y que tengan un ángulo de actuación más reducido.



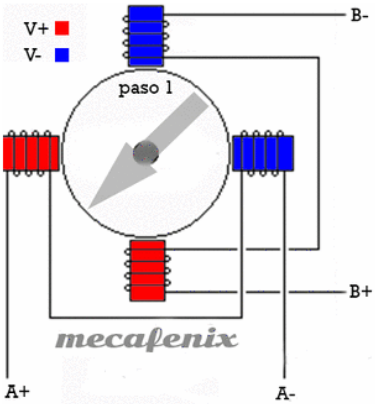
41. Motor reductancia variable (fuente: www.ingmecafenix.com)

- Motor paso a paso de imán permanente

En este caso existen dos variantes, los unipolares y bipolares. Los motores paso a paso unipolares suelen tener 5 o 6 cables de entrada, uno de alimentación y el resto para alimentar las bobinas. Este tipo es el más sencillo de controlar. Los motores paso a paso bipolares cuentan con 4 cables de salida. En su interior tienen una conexión en H por cada bobina, lo cual hace que su tarjeta controladora sea más compleja.



42. Motor unipolar (fuente: www.ingmecafenix.com)



43.Motor bipolar (fuente: www.ingmecafenix.com)

- Motores paso a paso híbridos

Este tipo de motor es el resultado de la combinación de los anteriores tipos de motores paso a paso. En este caso se alimenta tanto las bobinas como el imán.

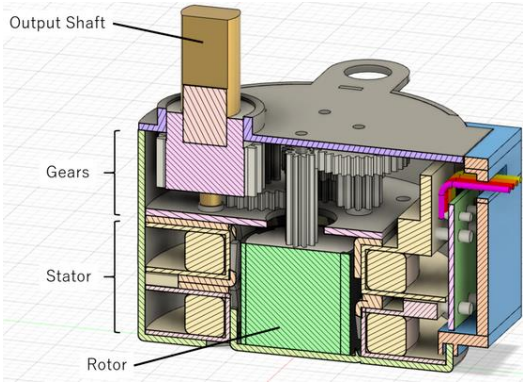
44.Motor paso a paso hibrido (fuente: www.biqtronica.com)

3.7.3. Motor paso a paso utilizado

45.Motor paso a paso 28BYJ-48 (fuente: tienda.bricogeek.com)

El 28BYJ-48 es un motor paso a paso unipolar de 4 fases, 64 pasos por vuelta, por lo que cuenta con 5 cables de salida, uno de alimentación y dos para cada bobina.

En este motor se alimentan las dos bobinas con una secuencia determinada, esto hace que el rotor del centro se mueva, que a su vez mediante una serie de engranajes mueve el eje de salida.



46. Motor 28BYJ-48 interior (fuente: https://cookierobotics.com/042/)

Para controlar este motor paso a paso, se necesita un driver. Para este proyecto se ha utilizado un driver ULN2003. Que acondicionará los pulsos recibidos por los pines de entrada junto a la alimentacion y tierra para que el servo se mueva correctamente.

Tamaño	35mm x 31mm x 11mm
Peso	30 g
Ángulo de Paso de par	5.625 / 64
Fuerza(torque)	0,3 Kg/cm

3.8. Servomotores

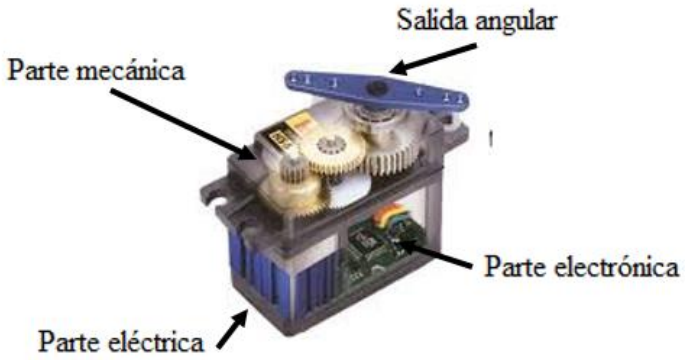
3.7.4. Funcionamiento

Existe la posibilidad de tener el funcionamiento de los motores paso a paso, desplazamiento angular, sin necesidad de utilizar un dispositivo externo, los servomotores.

Los servomotores son motores paso a paso de alto rendimiento, estos ya llevan integrados un dispositivo o *encoder* para conocer en qué posición se encuentran. Normalmente se utilizan para el control de posición en lazo abierto, ya que debido a su

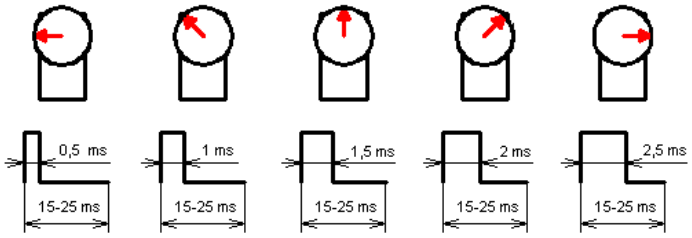
funcionamiento, la señal de actuación recibida nos dice el número de pasos que debe dar el motor.

Un servomotor cuenta con una parte electrónica que controla los ángulos de salida en función de la señal de entada. Una parte eléctrica, (motor de corriente continua) y una parte mecánica, una caja de engranajes, encargada de potenciar el par y reducir la velocidad del motor.



47.Partes servomotor (fuente: wwwpanamahitek.com)

Para hacerlo funcionar cuenta con 3 cables de entrada, alimentación o positivo (rojo) y neutro o negativo (negro o marrón), y el cable (amarillo, blanco o naranja) por el que llega la señal de control, pulsos con un determinado ancho (PWM). Como se muestra en la ilustración 45, se hace una equivalencia entre el ancho del pulso recibido y el ángulo final de salida del servomotor.



48.Equivalencia pulso ángulo

3.7.5. Tipos de servomotores

- Los servomotores que giran 360º en ambos sentidos y de forma continua. En estos la tensión de entrada determina la velocidad de giro. Si se hubiese utilizado este tipo de servomotor en este proyecto, hubiese sido necesario un sensor externo que nos dijera en qué posición angular se encuentra el servo en

cada momento, ya que las funciones utilizadas en MATLAB solo transmiten ancho de pulso, no velocidad de giro para un servomotor.

- Los servomotores que tiene un giro limitado de 180º. Estos servos funcionan con el método explicado antes, dependiendo del ancho del pulso recibido se posicionan en un ángulo u otro, debido a la forma de transferir la información desde MATLAB estos son los utilizados en este proyecto.

3.7.6. Servomotores utilizados

En este proyecto vamos a utilizar los siguientes servomotores:

- Servomotor 1

49. Servomotor HD-1711MG (fuente: www.tienda.bricogeek.com)

El servomotor HD-17711MG de la marca Power HD, es lo que se conoce como miniservo analógico, cuenta con engranajes de metal, el eje de salida está sostenido por un balero que optimiza el rodamiento y evita que se desgaste. Su tamaño y conector para la señal y alimentación son estándar compatible con todos los controladores de servomotores. Sus características más importantes son las siguientes:

Tamaño	29.5 x 11.6 x 30.2 mm
Peso	18.5 g
Velocidad de rotación	0.13 sec/60º
Fuerza(torque)	3Kg/cm

- Servomotor 2

50. Servo FS5115M-FB

(www.tienda.bricogeek.com)

Este es un servomotor analógico con piñones metálicos ofreciendo una mayor durabilidad. Este servo es de tamaño estándar y comporta como uno estándar, salvo que cuenta con un tercer cable de feedback, el cual nos proporciona información de si el servo tiene que llegar a una posición y realmente no está llegando.

Tamaño	40.8 x 20.1 x 38 mm
Peso	60 g
Velocidad	0.16 sec/60°
Torque	15.5Kg/cm

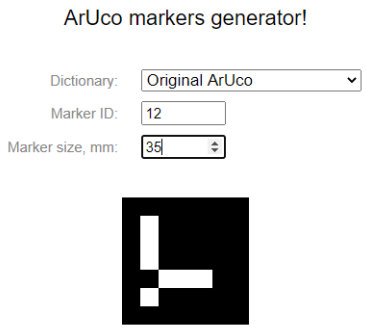
3.9. Marcas ArUco

Las marcas ArUco son un patrón o elemento que nos permite conocer la posición y orientación de un objeto respecto de la cámara con la que se está captando la imagen de la marca. Estas marcas están almacenadas en la librería ArUco.

Las marcas se detectarán con un programa de detección de marcas desarrollado por OpenCV[11] utilizando la librería ArUco.

[11] https://opencv.org/

Para generar marcas de esta librería disponemos de una página web[12] que lo hace automáticamente. Tiene 3 opciones para generar la marca: Un *Dictionary*, se ha elegido las marcas originales ArUco; Un *Marker ID* en el que se selecciona qué marca ArUco con una ID asociada se quiere imprimir; y un *Marker size* (en mm) para seleccionar el tamaño de las marcas. Una vez configurada la marca, podemos guardar la imagen como SVG o como PDF, se ha elegido esta segunda opción en el proyecto para imprimir la marca y que la cámara USB instalada en el robot fuera capaz de visualizarla.



51.ArUco markers generator
fuente: www.chev.me/arucogen/)

3.10. Módulo de cámara

Para que el brazo robótico de este proyecto pueda cumplir su función de "tener visión" y poder reconocer las marcas ARUCO es necesario que cuente con una cámara. La cámara no debe ser de grandes dimensiones ni pesada ya que esto incrementaría el coste y añadiría un peso extra al extremo del robot y se necesitarían motores más potentes.

MATLAB dispone de una serie de toolbox para poder recoger imágenes de cámaras. En concreto el que se va a utilizar puede comunicarse y recoger imágenes de cámaras conectadas mediante USB, esto hace que la conexión cámara ordenador sea la más sencilla.

[12] https://chev.me/arucogen/

Tras una búsqueda de cámaras idóneas se ha optado por el módulo de cámara USB HBV-1609[13]. Este módulo de cámara tiene una lente de 2 millones de píxeles, una gran angular de 120º para que el brazo robótico tenga un amplio rango de visión. Cuenta también con un chip OV2643, que puede almacenar imágenes. Su tamaño es de 35x31.5x19.3mm y con un peso de 23 gramos.

52. Módulo cámara USB HBV-1609 (fuente: www.amazon.com)

Una vez instalada la cámara en el extremo del robot e instalados el toolbox nombrados anteriormente en el manual, ya estará disponible para su utilización.

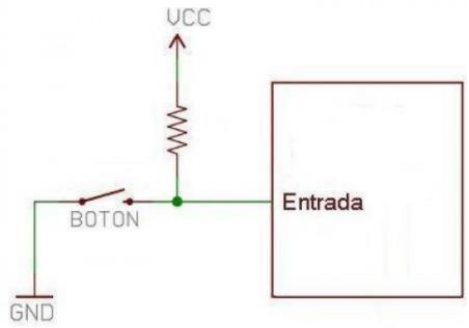
3.11. Placa

Para tener una buena organización del cableado y conexión del mismo con la placa Arduino y alimentación, se fabrica una placa con un circuito impreso o PCB (Printed circuit board) adaptado a las necesidades del proyecto.

Un circuito impreso es un medio para conectar eléctricamente y sostener los componentes electrónicos a través de pistas o rutas grabadas en una lámina de cobre sobre un sustrato no conductor.

[13] https://www.amazon.es/M%C3%B3dulo-HBV-1609-millones-p%C3%ADxeles-angular/dp/B07VQGFJC1/ref=sr_1_21?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=modulo+camara+usb&qid=1600858503&sr=8-21

La función de esta placa en este proyecto es conectar las señales de los motores del brazo robótico con la placa Arduino Uno y la alimentación. También cuenta con una resistencia pull-up para un mejor uso del final de carrera, necesario para el punto de calibración de la base.



53. Resistencia Pull-up (fuente:www.5hertz.com)

Esta resistencia pull-up asegura que no entre ruido por la señal del final de carrera. Por ello en el código se especifica que mientras la entrada sea 1 la base girará hasta que esté calibrada, se accione el final de carrera y la entrada sea 0, ya que la entrada estará conectada directamente a tierra.

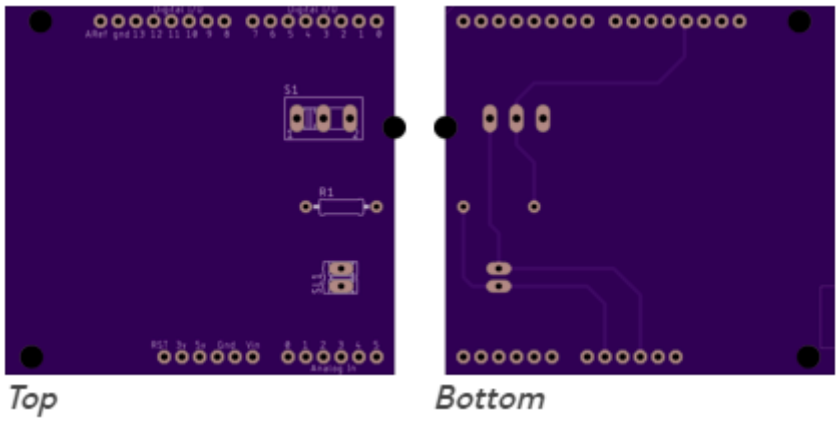
Hay diferentes métodos para crear un circuito impreso en una placa. En este caso se va a utilizar una insoladora. Al usar una placa sensibilizada positiva, la resina no es soluble al exponerla a la luz ultravioleta. Por ello se necesita que las pistas o rutas del circuito sean opacas. Se imprime este circuito en una transparencia de acetato, dejando las pistas opacas y el resto del diseño transparente.

Se coloca la transparencia de acetato sobre la placa virgen de cobre, ajustando el dibujo del circuito a la placa. Ahora se introduce la placa en una maquina insoladora, consiste en una caja opaca con tubos fluorescentes con una gran cantidad de radiación ultravioleta. Al terminar la placa se sumerge en un líquido revelador

Después se introduce en baño químico donde comienza a disolverse el cobre donde no hay pistas o rutas, dejando el dibujo del circuito impreso. Como último paso será necesario eliminar el resto de resina que queda sobre las pistas o rutas, se puede utilizar un disolvente común o volver a introducir la placa en la insoladora sin tapar ninguna zona.

Para terminar será necesario hacer taladros a la placa para introducir y conectar los componentes.

Para este proyecto se ha escogido la opción de pedir el circuito impreso a una empresa y así asegurar el buen acabado de la placa.



54.Placa final (fuente: propia)

3.12. Conexiones

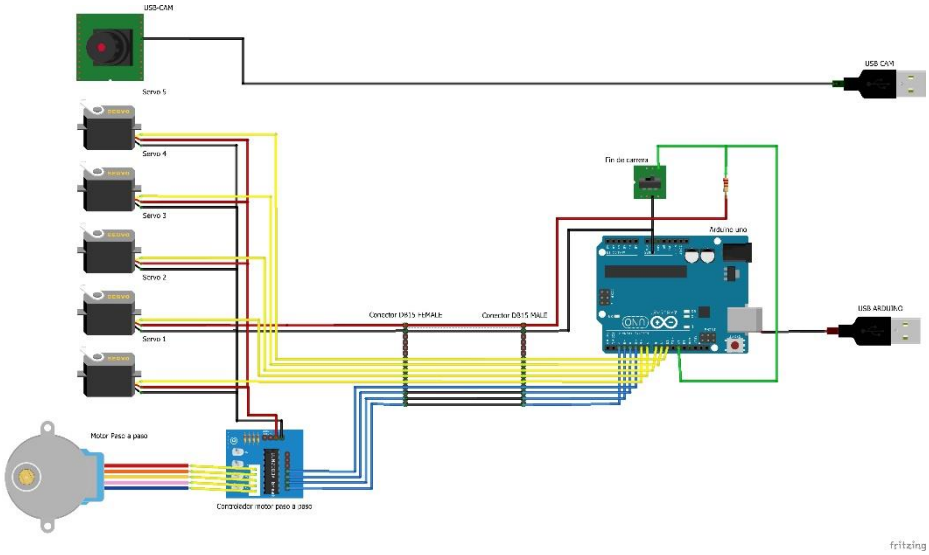
Las conexiones del brazo robótico con el ordenador mediante una placa Arduino serán sencilla e intuitiva.

La alimentación del brazo robótico se genera en una fuente de alimentación externa, esta llega a todos los componentes junto con la negativa, recordar también conectar todas las señales negativas juntas incluso la de la placa Arduino Uno. La señal de cada servo se conecta al pin correspondiente de la placa Arduino Uno. También cuenta con una placa para incluir una resistencia pull-up para el conector final de carrera.

En cuanto al módulo de la cámara irá directamente conectado por cable USB al ordenador que maneje el robot.

Para que el cableado del robot sea más limpio y organizado se ha decidido incluir un conector BD15 en la base del robot para conectarse al Arduino, así facilita el manejo del robot cuando no se utilice.

El esquema de conexiones quedaría de la siguiente manera:



55.Esquema de conexiones (fuente: propia)

4. EXPERIMENTOS

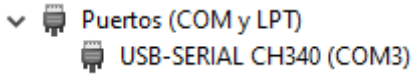
Para verificar que todo funcionase correctamente, han sido necesarias varias pruebas de movimiento y visión, las cuales se explican en los siguientes apartados.

4.1. Prueba servomotores

Es necesario tener instalado el IDE de Arduino y Matlab, toda la instalación se explica en el manual.

Para comprobar el correcto funcionamiento de los servomotores empezaremos con unas líneas de código simples:

Primero se declara el objeto Arduino en el puerto de entrada, en el caso práctico de este proyector es el puerto ‘COM3’ y lo llamaremos ‘ard’. Para conocer el puerto de entrada al que se está conectando la placa Arduino, el IDE Arduino lo puede reconocer automáticamente, o desde el “administrador de dispositivos”.



56.Puerto Arduino (fuente: propia)

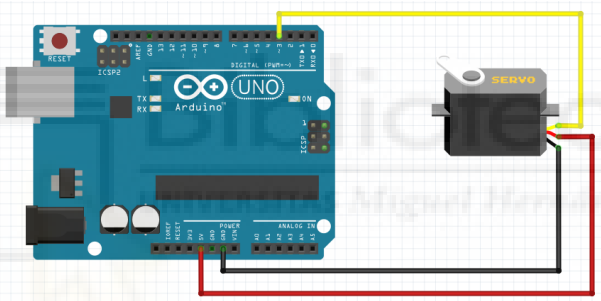
Matlab cuenta con un Arduino “ficticio” para comprobar el funcionamiento del código, para ello es necesario cambiar el puerto por ‘Demo’.

Se accede al objeto Arduino llamado ‘ard’, y se declara que se va a conectar un Servo a un pin, en este caso pin 3. Después se da el valor del ángulo, en este caso 90º.

```
Untitled.m
1 -    puerto='COM3';
2 -    ard=arduino(puerto);
3 -    pin=3;
4 -    ard.servoAttach(pin);
5 -    angulo=90;
6 -    ard.servoWrite(pin,angulo);
```

57. Ejemplo código servo (fuente: propia)

La conexión para esta prueba es:



58.Conexión servo (fuente: prueba)

Si todo está correctamente conectado e instalado el servo se moverá a 90 grados.

El toolbox de ARTE calcula los ángulos de las articulaciones del robot en radianes, por lo que será necesario convertirlos en grados con la función “rad2deg”. A parte la función “ard.servoWrite” funciona de 0 a 180 grados, mientras que ARTE calcula los grados de las articulaciones entre -90º y +90º. Por lo tanto será necesario sumarle 90 grados.

```
 9 -    ang_radianes=1.5708;
10 -    ang_grados=rad2deg(ang_radianes);
11 -    ang_grados_final=ang_grados+90;
12 -    ard.servoWrit(pin,ang_grados_final);
```

59.Conversión ángulos servo (fuente: propia)

ARTE construye un objeto robot tipo matriz, con los parámetros del robot, para poder hacer los cálculos. Será necesario acceder a los grados de las articulaciones mediante “robot.q”.

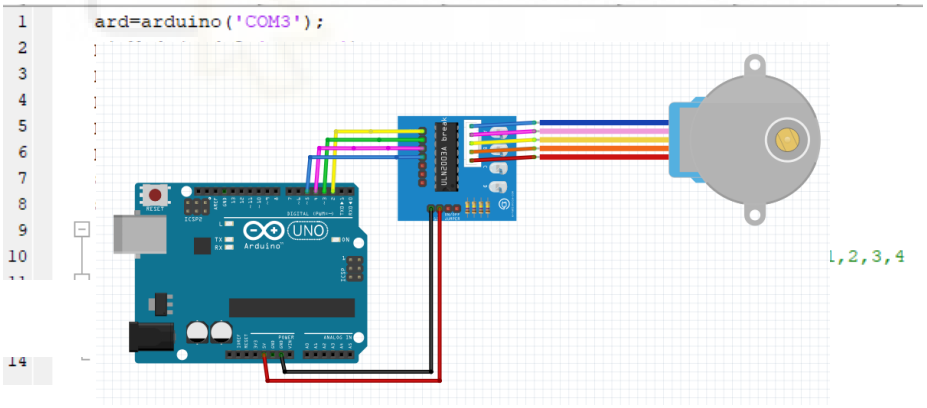
Para comprobar que el servo se mueve correctamente, se ejecutará esas líneas cada vez que movemos el robot en la simulación.

```
5 -     angulo_radianes=robot.q;
```

60.Acceso a los datos robot (fuente: propia)

4.2. Prueba motor paso a paso

Al igual que para la prueba con el servomotor, hay que declarar el objeto Arduino en el puerto de entrada correspondiente. Después declarar que cada pin es de salida y determinar la secuencia de alimentación de cada pin. Por último se genera un bucle que vaya pasando por cada secuencia.



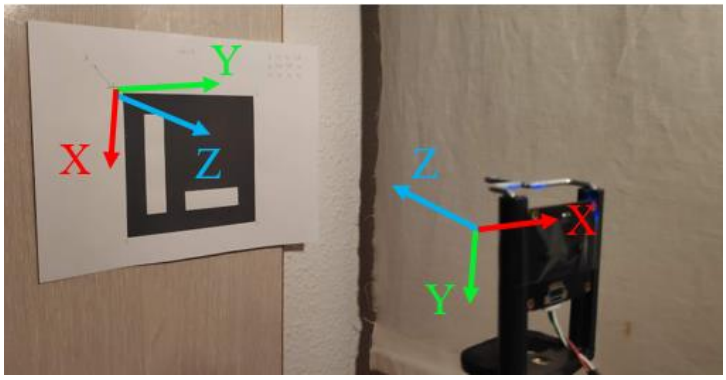
62.Esquema conexión stepper (fuente: propia)

El esquema de conexión para esta prueba:

Si todo está conectado correctamente el motor paso a paso comenzará a dar vueltas en sentido horario.

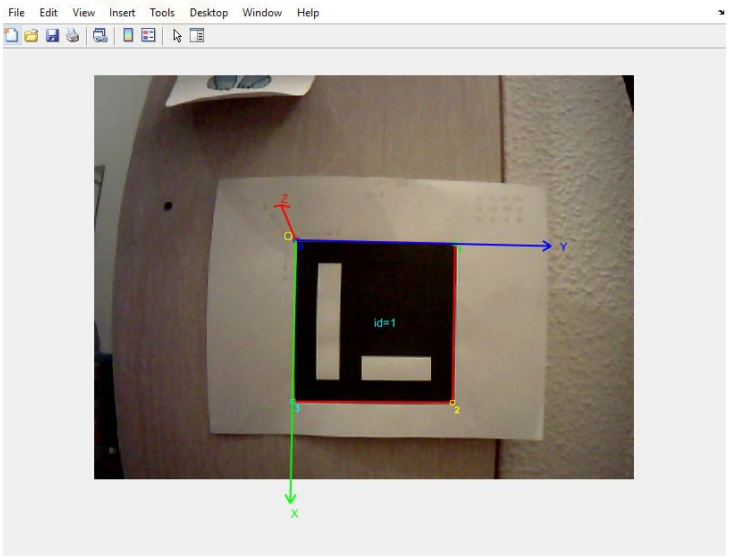
4.3. Prueba detección marcas ArUco

Para que el brazo robótico tuviera "visión" se va a utilizar un programa de detección de marcas codificadas. Es necesario descargar la carpeta "PracticaPOSEMarcas" para poder hacer la prueba de detección y posterior modificación para adaptar el código para su uso con el brazo robótico.



63.Colocación Marca ARUCO (fuente: propia)

Para la prueba es necesario imprimir la marca aruco que contiene la carpeta con el nombre "marker_id1", después se ejecuta el código "pose1.m". Se abrirá una ventana con la imagen en tiempo real capturada por la cámara USB. Se coloca la marca ARUCO impresa previamente en el campo de visión de la cámara.



64.Visión cámara (fuente: propia)

Cuando se termine de ejecutar el código, mostrará la matriz de rotación del origen de referencia de la marca ARUCO respecto del origen de la cámara. También mostrará la matriz de traslación de la marca ARUCO respecto de la cámara.

$$Rotación = \begin{bmatrix} -0.0181 & 0.9977 & 0.0647 \\ 0.9995 & 0.0198 & -0.0262 \\ -0.0274 & 0.0642 & -0.9976 \end{bmatrix} \quad Traslación = \begin{bmatrix} -43.3277 \\ -19.3901 \\ 395.9999 \end{bmatrix}$$

En este caso el resultado muestra que la marca ARUCO estará aproximadamente a 396 mm de distancia en el eje Z. Con estas dos matrices sabemos la posición y orientación exacta de la marca ARUCO.

Se ha adaptado este código para que pueda funcionar con el brazo robótico. Más adelante se explicará el funcionamiento de las funciones creadas.

4.4. Prueba movimiento hacia marca ArUco

La librería ARTE de Matlab permite mover el brazo robótico a un punto concreto. Para llegar a él es necesario saber la posición y orientación del punto respecto a la base del robot. La función "POSE" proporciona la posición y orientación de la marca ARUCO respecto de la cámara USB (Línea 2). Se enlazar esa relación con el extremo del brazo robótico y después a la base del robot.

Primero es necesario formar una matriz con los datos proporcionados por la función POSE, posición y orientación de la marca ARUCO respecto de la cámara (línea 4 – 6). Después, hay que referenciar la cámara en el extremo del robot a la base del brazo robótico (línea 8 - 9).Se referencia la marca ARUCO a la base del robot (línea 11). Finalmente para conocer la posición de cada articulación del brazo robótico se calcula la cinemática inversa (línea 13).

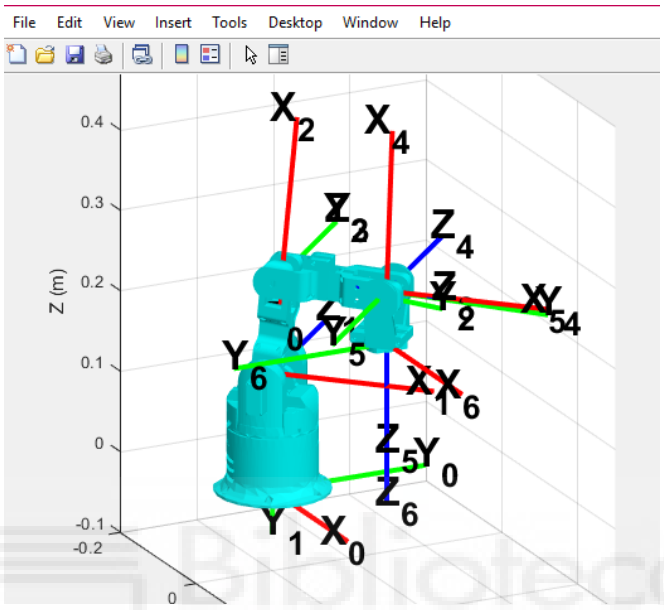
```
1   %Función POSE
2   [R, t]=POSE_elegir;
3   %Marca ARUCO - Cámara USB
4   T_marca_cam(1:3,1:3)=R;
5   T_marca_cam(1:3,4)=t./1000;
6   T_marca_cam(4,:)=[0 0 0 1];
7   %Cámara USB - Base robot
8   T_0=directkinematic(robot,robot.q);
9   T_cam_base=T_0;
10  %Marca ARUCO - Base robot
11  T_marca_base=T_cam_base*T_marca_cam;
12  %Posición robot
13  q_marca=inversekinematic(robot,T_marca_base);
```

65.Código prueba movimiento (fuente: propia)

Para comprobar el funcionamiento se crea una matriz de rotación ideal:

$$R = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \qquad T = \begin{bmatrix} -100 \\ 100 \\ -265 \end{bmatrix}$$

Se comprueba que el brazo robótico se mueve como corresponde:



66.Simulación prueba movimiento (fuente: propia)

5. Manual

En las siguientes páginas se explicará cómo poner en marcha el brazo robótico, los primeros pasos de la configuración y control, para poder localizar y coger objetos con marcas ARUCO.

Antes de todo es necesario tener instalado MATLAB y el IDE Arduino.

5.1. Calibración cámara

Para la calibración de la cámara es necesario tener descargado la caja de herramientas con el nombre *toolbox_calib*. Hay imprimir el patrón *pattern.pdf*, dentro de la carpeta *calibration_pattern* para poder calibrar la cámara.

El primer paso para calibrar la cámara del robot, es ejecutar el programa *capture.exe*. En el caso de que haya más de dos cámaras conectadas al ordenador, es posible que al ejecutarlo nos pregunte qué cámara queremos utilizar, en ese caso se elige

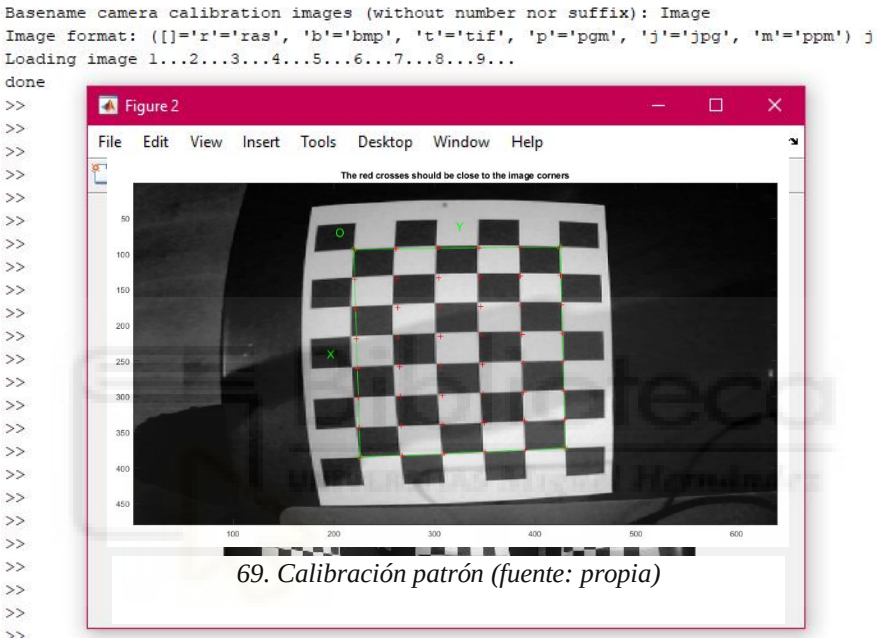
67.Elegir cámara USB (fuente: propia)

la cámara USB que esté conectada al brazo robótico. Otro caso es que coja por defecto la cámara que no corresponde (por ejemplo ordenadores portátiles con cámara integrada), en ese caso será necesario deshabilitar la otra cámara desde el "Administrador de dispositivos del ordenador".

Aparecerá una ventana con lo que está captando en tiempo real la cámara USB del brazo robótico y otra ventana con las instrucciones para utilizarlo. A continuación se coloca el patrón, impreso previamente, delante de la cámara y se va pulsando la tecla espacio para ir tomando varias imágenes con el patrón en diferentes posiciones. Deberá aparecer el mensaje *"Saving image window in file: Image1.jpg"* cada vez que se haga una instantánea. Cuando se hayan tomado suficientes imágenes como para tener diferentes

puntos de referencia en el espacio, se pulsa la tecla Q para salir del programa. Las imágenes generadas se autoguardarán en la misma carpeta del programa.

A continuación se copia todas esas imágenes creadas, a la carpeta *Imágenes calib*, dentro de la caja de herramientas *toolbox_calib*. En la línea de comandos de Matlab se ejecuta "*Calib_gui*", se abrirá un menú, pulsar "*Imagen names*" para cargar las fotos tomadas anteriormente. El programa pedirá el nombre de las imágenes que se deben cargar, sin el sufijo del número, y el formato en el que se guardaron. Se abrirá una ventana con las imágenes que ha cargado.



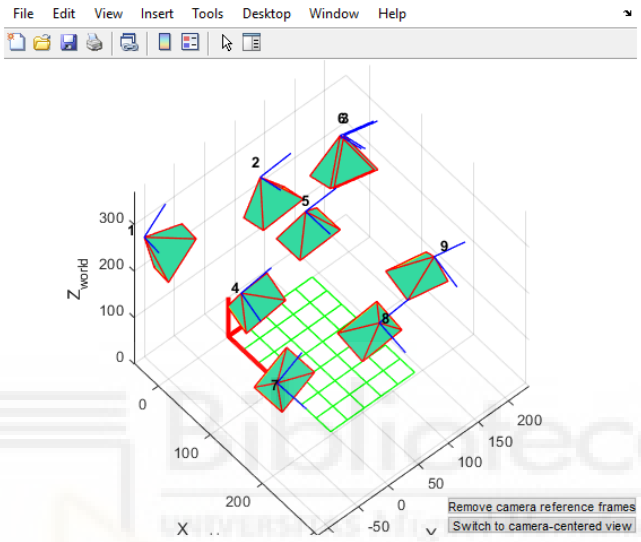
69. Calibración patrón (fuente: propia)

68. Carga imágenes calib_gui (fuente: propia)

Ahora para calibrar la cámara es necesario seleccionar las 4 esquinas del patrón, para ello es necesario volver al menú de *calib_gui* y pulsar en *"Extract grid corners"*. A continuación nos pedirá el número de imágenes a procesar, y se elegirán todas para tener una mayor precisión. Pedirá el número de cuadrados que hay en el eje x y en el eje y, además de lo que mide cada cuadrado que en este caso práctico es 28mm. Todo este proceso de calibración se explica en la página web del toolbox[14].

[14] http://www.vision.caltech.edu/bouguetj/calib_doc/

El programa calculará los parámetros de calibración de la cámara, junto con unas distorsiones. Todos estos datos de la cámara calibrada se generan al pulsar *"Calibration"* en el menú anterior *"calib_gui"*. Se generan los documentos *Calib_Result.m*, *calib_Results.mat* y *calib_data.mat* para su posterior uso en el reconocimiento de las marcas ARUCO. Este menú de *calib_gui* nos deja hacer una simulación en 3D para visualizar los puntos y ángulos desde que se han tomado las imágenes del patrón para su calibración, pulsando *"Show extrinsic"*:



70. Puntos toma imágenes cámara (fuente: propia)

Por ultimo pulsamos en SAVE para guardar todos los cálculos realizados en MATLAB y poder usarlo posteriormente.

IMPORTANTE: Una vez terminada la calibración de la cámara es necesario quitar el *toolbox_calib* del path de MATLAB donde se esté trabajando, ya que pude entrar en conflicto con otros programas, concretamente con el control del brazo desde la función *teach*.

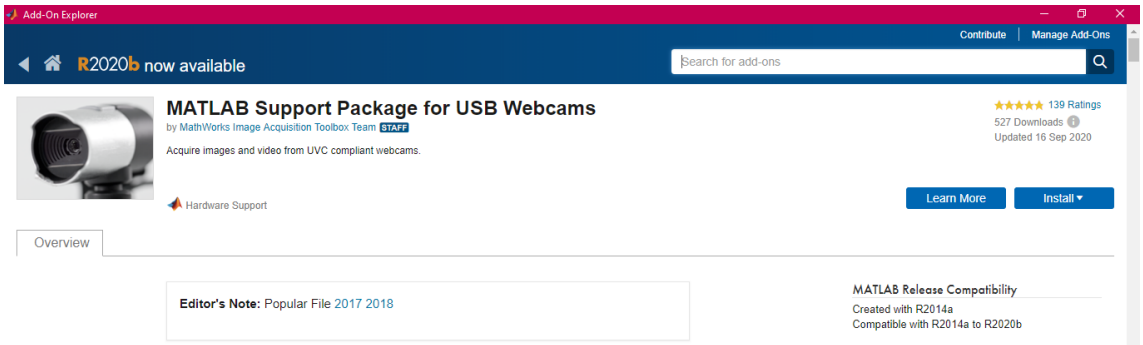
5.2. Configuración en MATLAB

Para que el tolbox de ARTE pueda controlar el brazo robótico será necesario instalar ciertos paquetes en MATLAB. Para la instalación de éstos paquetes, abrimos MATLAB, en el apartado *Enviroment* > *Add-Ons* > *Get Hardware Support Packages* y en el buscador, escribimos los nombres de los dos paquetes necesarios.

71.Instalar Hardware Support Packages (fuente: propia)

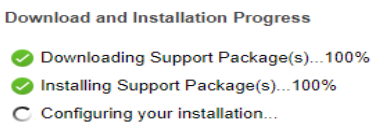
Los dos paquetes necesarios para el correcto funcionamiento son los nombrados anteriormente:

- MATLAB Support Package for USB Webcam



72.Toolbox USB Webcam (fuente: propia)

Para instalar este paquete pulsaremos “install” y comenzará la instalación:



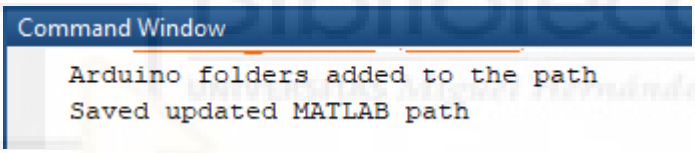
73.Instalación de paquete (fuente: propia)

- Legacy MATLAB and Simulink Support for Arduino

74.Toolbox Arduino (fuente: propia)

Para instalar este toolbox es necesario pulsar add.

Una vez instalados esos dos paquetes, en el *Command Window* (ventana de comandos) de MATLAB y se ejecuta *"install_arduino"*, al escribir esto estaremos añadiendo las carpetas con las funciones necesarias para que MATLAB reconozca los tipos de datos Arduino. Al escribirlo debería aparecer un mensaje como el siguiente:



75.Mensaje final install_arduino en MATLAB (fuente: propia)

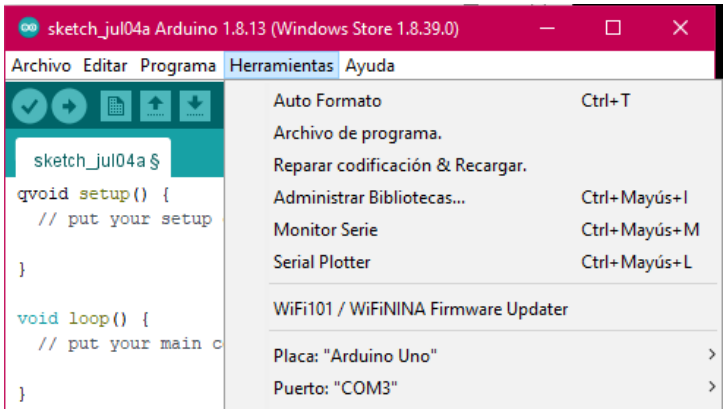
Con estos paquetes instalados MATLAB ya estará listo para su uso en este proyecto.

5.3. IDE Arduino

Para instalar el IDE de Arduino se accede a la página web oficial de Arduino[15] y se descarga el software idóneo para las características de nuestro ordenador. Al ejecutar el IDE de Arduino aparecerá esta ventana o escritorio:

76.Ventana inicio IDE Arduino (fuente: propia)

Una vez ejecutado el IDE de Arduino, es necesario configurarlo para el uso de la placa Arduino Uno, para ello, primero se conecta la paca Arduino Uno a un puerto USB del ordenador, a continuación se pulsa *Herramientas>Placa>Arduino Uno* y se indica qué placa vamos a utilizar. Se vuelve a pulsar *Herramientas>Puerto* y se elige el puerto



77.Pestaña herramientas IDE (fuente: propia)

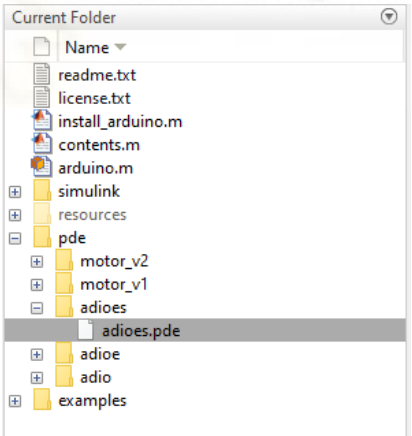
[15] https://www.arduino.cc/en/main/software

USB al que se haya conectado la placa Arduino, la mayoría de las ocasiones el mismo IDE detecta a qué puerto se ha conectado la placa.

Para obtener el código que usa la placa para enlazarse con MATLAB es necesario volver a la página de descarga del paquete *Legacy MATLAB and Simulink Support for Arduino* (descargado en el apartado anterior). En esta página aparece el botón *"Open Folder",* al pulsarlo automáticamente nos abrirá en MATLAB la carpeta con los recursos necesarios.

78.Open Folder Paquete (fuente: propia)

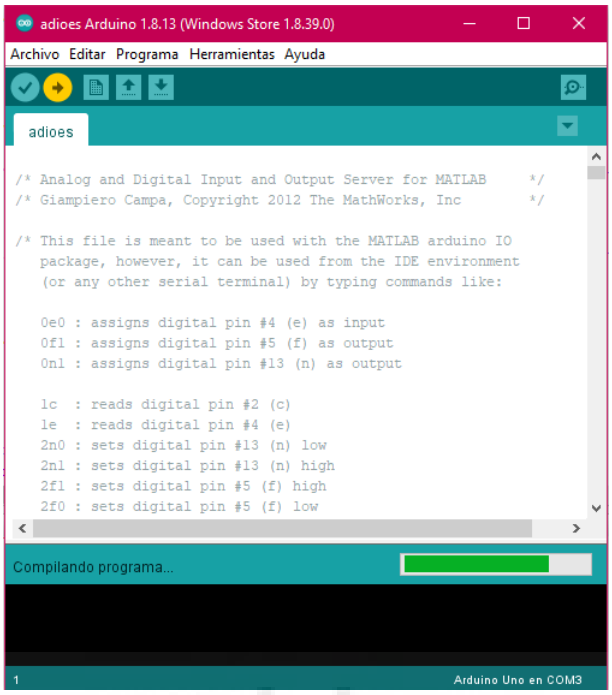
En este directorio pinchamos en la carpeta *"pde"* y luego en la carpeta *"adioes"*. El archivo .pde que encontramos en esta carpeta es el código que necesitamos instalar en la placa Arduino. Lo copiamos y pegamos en escritorio del ordenador u otro lugar que se prefiera, pero sin perderlo su localización ya que nos hará falta más adelante.



79.Archivo adioes.pde (fuente: propia)

Volvemos al IDE de Arduino y pulsamos *Archivo>Abrir* y localizamos el archivo *"adioes.pde"* que acabamos de mover de lugar. A continuación se pulsa *"Subir"* y el IDE comenzará a compilar y subir a la placa Arduino el código.

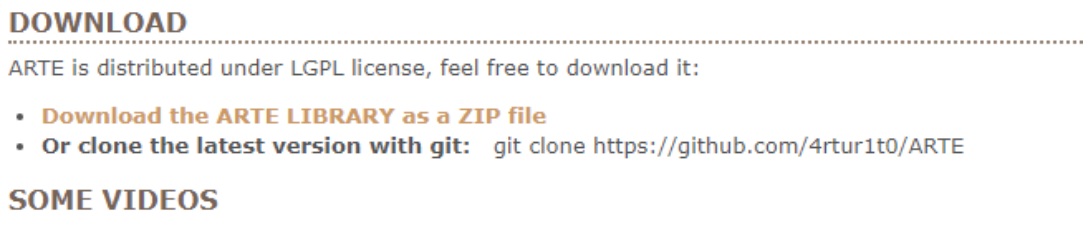
Cuando nos muestre el mensaje *"Subido"*, la placa Arduino Uno estará lista para comunicarse con MATLAB.



80.Compilando código Arduino (fuente: propia)

5.4. Inicialización Robot

Se entra en la página oficial de ARTE[16], accediendo al apartado DOWNLOAD se descarga un archivo ZIP. Una vez descargado se descomprime el archivo y pega en la misma carpeta donde se vaya a trabajar con los archivos del brazo robótico.

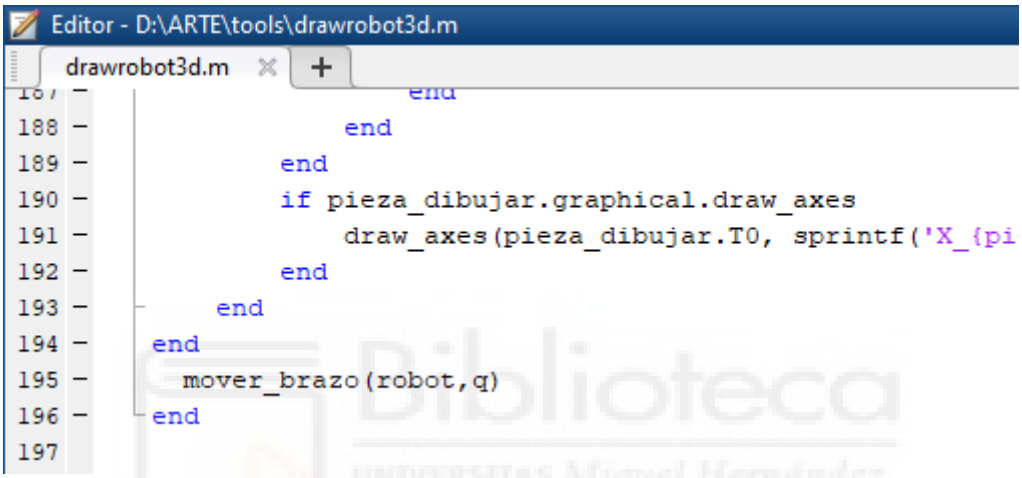


81. Download ARTE (fuente: https://arvc.umh.es/arte/index_en.html)

[16] https://arvc.umh.es/arte/index_en.html

Una vez pegado en la carpeta donde se va a trabajar. En la línea de comandos se ejecuta la función "*init_lib*", esta creará las variables y estructura necesaria para trabajar con ARTE.

Para que este brazo robótico funcione correctamente es necesario modificar el archivo original "drawrobot3d" que viene por defecto en ARTE. Para ello se abre por la línea de comandos la función, escribiendo "*open drawrobot3d*". En la penúltima línea se escribe la función diseñada para hacer funcionar el brazo robótico "*mover_brazo (robot,q)*".



82. Modificación función drawrobot3d (fuente: propia)

Esta función simula en 3d el robot en cada posición que va tomando, al incluir la función *mover_brazo*, se mueve al mismo tiempo el brazo robótico real. En caso de no querer mover el brazo robótico basta con dejar la función drawrobot3d tal cual se descarga sin modificar nada y solo hará una simulación del brazo robótico cargado.

Para cargar los parámetros del brazo robótico se ejecuta en la línea de comandos la función "*robot=load_robot*" y se abrirá una ventana donde se debe seleccionar el archivo .m con los parámetros del robot "*parameters.m*". Automáticamente cargará los parámetros del robot.

5.5. Funcionamiento del brazo robótico

Si se ha activado la función "*mover_brazo*" la ejecutará. El código de esta función primero comprobará si ya se ha utilizado el puerto USB seleccionado, si no lo ha hecho, conectará la placa Arduino Uno. Después cargará las constantes necesarias para el

correcto funcionamiento del brazo robótico. Asignará un modo de funcionamiento a cada pin, ya sea modo “output” o “input” dependiendo de si es una salida de señal para cada servomotor o la entrada de la señal del final de carrera del calibrado. A continuación Calibra la base del robot. Una vez calibrada ejecuta el bucle para actualizar la posición de cada articulación enviando la información a cada servomotor por su correspondiente pin.

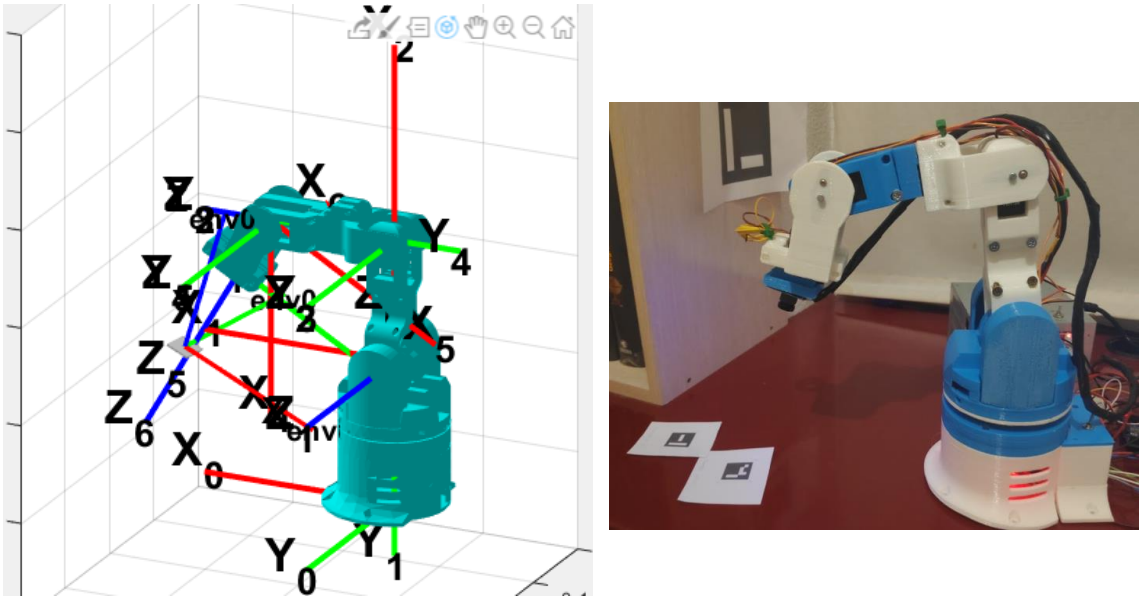
Como resultado el brazo robótico se moverá a la vez que la simulación de ARTE en Matlab.

5.6. Utilización funciones

- Elección de marca ArUco a coger

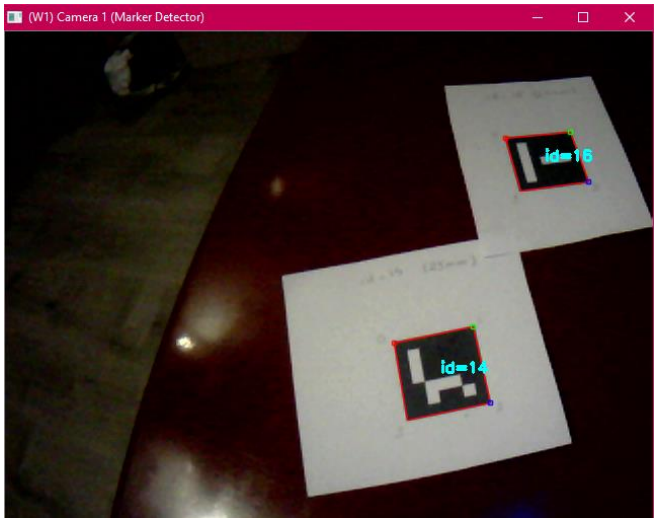
Para ejecutar esta función s escribe en la línea de comandos “*Elegir_marca*”.

El funcionamiento de esta función comienza haciendo una serie de comprobaciones: comprobar si se han cargado los parámetros del brazo robótico; se cargará un archivo .stl de la marca ARUCO de 25mm; directorio donde se encuentra el archivo “markers.exe” necesario para la correcta ejecución de la función. Después coloca el brazo robótico apuntando hacia la superficie donde se encuentre y ejecuta la función “*POSE_elegir*” para localizar la marca ARUCO.



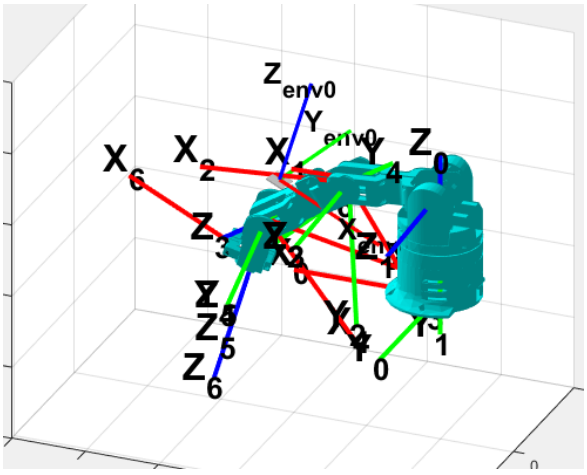
83.Robot apuntando a superficie (fuente: propia)

La función “POSE_elegir” da opciones de si se quiere localizar una marca en tiempo real o en una imagen almacenada. Detectará cuantas marcas ARUCO hay en la imagen tomada, si solo hay calculará sus coordenadas, si hay más de una, mostrará cuales ha encontrado y dará opción a elegir cual se quiere localizar.



84.Visión robot (fuente: propia)

Una vez localizada hace una serie de transformaciones de las coordenadas para poder utilizar los datos dados por la función. Y ejecuta la cinemática directa del robot para alcanzar la marca ARUCO.

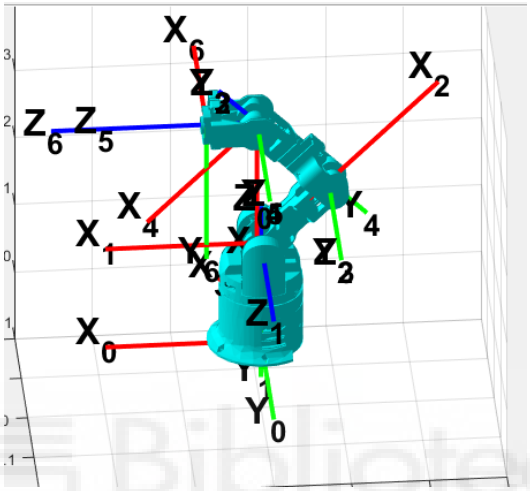


85.Robot localizando marca ARUCO (fuente: propia)

- Seguimiento a tiempo real de marcas ArUco

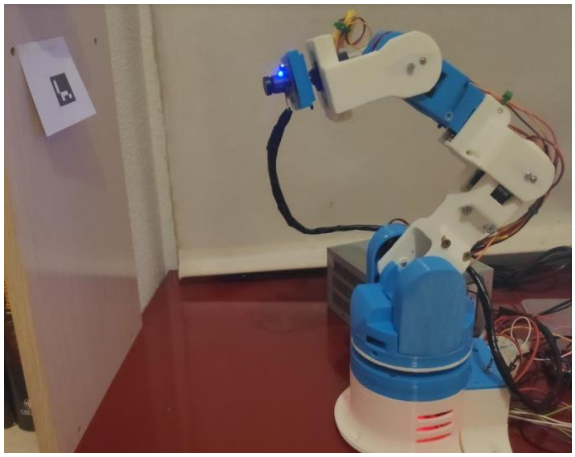
Para ejecutar esta función es necesario escribir en la línea de comandos "*Seguir_tiempo_Real*".

Al igual que la anterior función comenzará haciendo una serie de comprobaciones para la correcta ejecución. Una vez hechas esas comprobaciones cargará unas constantes necesarias.



86.Robot posición barrido (fuente: propia)

El brazo robótico comenzará moverse haciendo un barrido para encontrar una marca ARUCO. Una vez encontrada la marca, comenzará a seguirla a una distancia determinada. Si la pierde del campo de visión volverá a la posición inicial y comenzará a hacer el barrido.



87.Robot apuntando marca ARUCO (fuente: propia)

6. BIBLIOGRAFIA

[1] Wikipedia. Robótica

https://es.wikipedia.org/wiki/Rob%C3%B3tica

[2] Wikipedia. Brazo robótico.
https://es.wikipedia.org/wiki/Brazo_rob%C3%B3tico

[3] Robots paralelos. Autor: Juan Pérez.
https://es.slideshare.net/htrmoreno/robots-paralelos

[4] Wikipedia. Arduino Uno

https://es.wikipedia.org/wiki/Arduino_Uno

[5] Proyectoidis. Unimate

http://proyectoidis.org/4000-pound-unimate/

[6] Wikipedia. Robot Sophia

https://es.wikipedia.org/wiki/Sophia_(robot)#/media/Archivo:Sophia_at_the_AI_for_Good_Global_Summit_2018_(27254369347)_(cropped).jpg

[7] MATLAB

https://matlab.mathworks.com/

[8] Impresora3d. Historia sobre impresoras 3D

https://www.impresoras3d.com/breve-historia-de-la-impresion-3d/

[9] Areatecnologia. Tipos de impresoras.

https://www.areatecnologia.com/informatica/impresoras-3d.html#Tipos_de_Impresoras_3D

[10] Tresdpro. Material PLA.

https://tresdpro.com/que-es-el-material-pla/

[11] Panamahitek. Funcionamiento Servomotor.

http://panamahitek.com/que-es-y-como-funciona-un-servomotor/

[12] Wikipedia. Servomotor

https://es.wikipedia.org/wiki/Servomotor

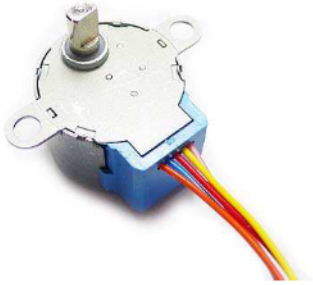
[13] Parámetros de DHL:
https://es.qwe.wiki/wiki/Denavit%E2%80%93Hartenberg_parameters#Denavit%E2%80%93Hartenberg_matrix

[14] Robot Analysis: The Mechanics of Serial and Parallel Manipulators. Lung-Weng Tsai (1999). John Wiley & Sons, Inc. ISBN: 978-0471325932

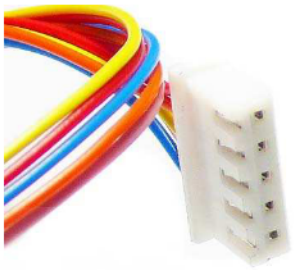
[15] Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Peter Corke. ASIN : B072FM8F5C

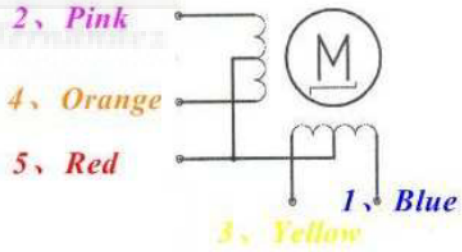
28BYJ-48 – 5V Stepper Motor

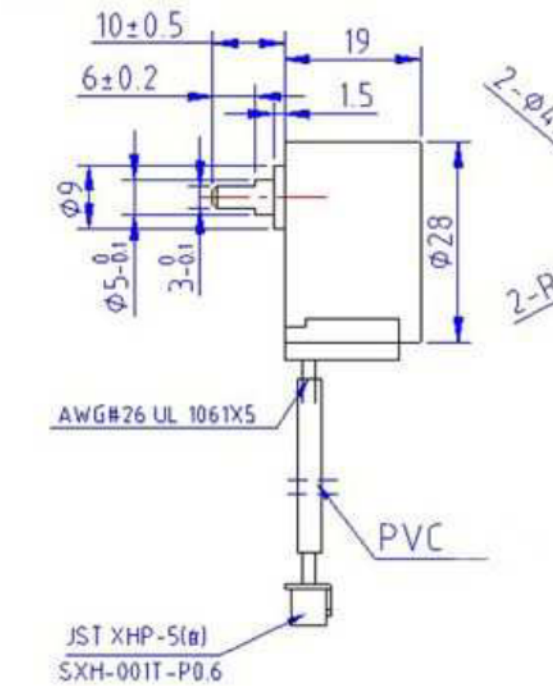
The 28BYJ-48 is a small stepper motor suitable for a large range of applications.

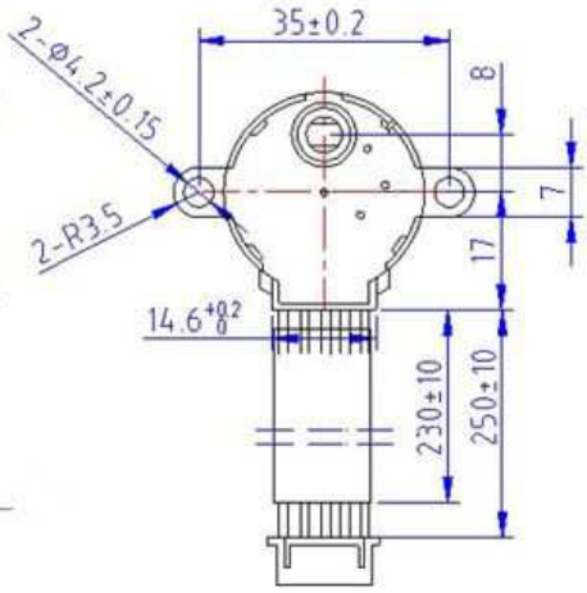


Rated voltage :	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25℃)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MΩ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz,No load,10cm)
Model	28BYJ-48 – 5V









FEETECH RC Model Co.,Ltd.

产品规格书

Specification of Product

产品名称：6V 15公斤模拟舵机 Product Name：6V 15kg.cm Analog Servo
产品型号 Model No.FS5115M

1.使用环境条件 Apply Environmental Condition

No.	Item	Specification
1-1	存储温度 Storage Temperature Range	-30℃～80℃
1-2	运行温度 Operating Temperature Range	-15℃～70℃

2.测试环境 Standard Test Environment

No.	Item	Specification
2-1	温度 Temperature range	25℃ ±5℃
2-2	湿度 Humidity range	65%±10%

3.机械特性 Mechanical Specification

No.	Item	Specification
3-1	尺寸 Size	A：40.8mm B：20.1mm C：38mm D：49.5mm
3-2	重量 Weight	58.5g
3-3	齿轮类型 Gear type	Metal Gear
3-6	机构极限角度 Limit angle	200° ±5°
3-7	轴承 Bearing	2 Ball bearings
3-8	出力轴 Horn gear spline	25T
3-9	摆臂 Horn type	Plastic,POM
3-10	外壳 Case	Nylon & Fiberglass
3-11	舵机线 Connector wire	300mm ±5 mm
3-12	马达 Motor	Metal brush motor
3-13	防水性能 Splash water resistance	NO

4.电气特性 Electrical Specification (Function of the Performance)

No.	工作电压 Operating Voltage Range	4.8V	6V
4-1*	静态电流 Idle current(at stopped)	5mA	7mA
4-2*	空载速度 No load speed	0.18sec/60°	0.16 sec/60°
4-3*	空载电流 Runnig current(at no load)	160 mA	190 mA
4-4	堵转扭矩 Peak stall torque	14kg.cm	15.5kg.cm
		194.8oz.in	215.6oz.in
4-5	堵转电流 Stall current	1200 mA	1500mA

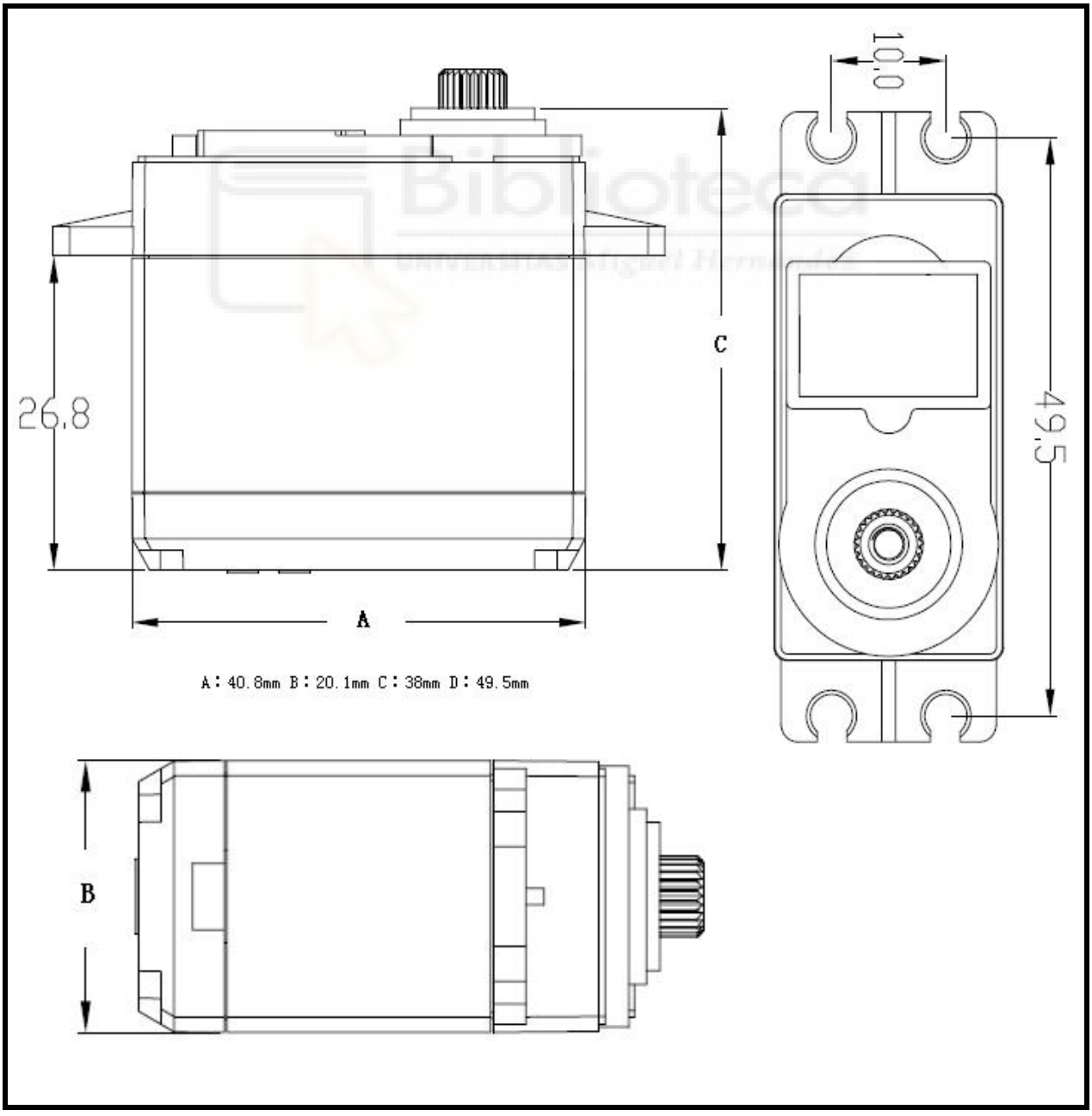
Note: "*"definition is average value when the servo runing with no load

产品名称：6V 15公斤模拟舵机 Product Name：6V 15kg.cm Analog Servo
产品型号 Model No.FS5115M

5.控制特性 Control Specification:

No.	Item	Specification
5-1	控制信号 Command signal	Pulse width modification
5-2	放大器类型 Amplifier type	Analog comparator
5-3	脉冲宽度范围 Pulse width range	500～2500μsec
5-4	中立位置 Neutral position	1500 μsec
5-5	旋转角度 Running degree	180° （±5°）(when 500～2500 μsec)
5-6	死区宽度 Dead band width	5 μsec
5-7	旋转方向 Rotating direction	Counterclockwise (when 1000～2000 μsec)

6.外形图 The Drawings



Address: 1707,Huashang Building,Xiapu district,Huizhou,Guangdong,China
E-mail:info@chd.hk Tel:86 752 2118844 Fax:86 752 2118860 WWW.CHD.HK

1. 使 用 環 境 條 件
Apply Environmental Condition：

No.	项目 item	规格 standard
1-1	保存溫度 Storage Temperature Range	-20℃～60℃
1-2	操作溫度 Operating Temperature Range	-10℃～50℃
1-3	操作電壓 Operating Voltage Range	4.8V~6.0V

2. 測 試 環 境
Standard Test Environment：

2-1	測試環境 Standard Test Environment	每一个检查必须是正常的温度和湿度进行测量，温度 25 ± 5℃，相对湿度 65 ± 10%,在按照本规范的标准测试条件下判断特征。 Every characteristic of the inspect must be normal temperature and humidity carry out the test , temperature 25±5℃ and relative humidity 65±10% of judgment made in accordance with this specification standard testing conditions.

3. 外 觀 檢 查
Appearance Inspection：

No.	项目 item	规格 standard
3-1	外觀尺寸 Outline Drawing	尺寸见附件 Dimension see the attachment
3-2	外觀 Appearance	无损坏，不允许影响功能 No damage which affects functions allowed

4. 電 氣 特 性

Electrical Specification (Function of the Performance)：

No.	項目 item	4.8V	6.0V
4-1	空載轉速 Operating speed (at no load)	0.13 sec/60°	0.11 sec/60°
4-2	空載電流 Running current (at no load)	180 mA	240 mA
4-3	停止扭力 Stall torque (at locked)	3.0 kg-cm	3.5 kg-cm
4-4	停止電流 Stall current (at locked)	1400 mA	1600 mA
4-5	待機電流 Idle current (at stopped)	5 mA	5 mA

注：项目 4-2 定义平均值时，伺服器无负荷运行

Note: Item 4-2 definition is average value when the servo running with no load

5. 機 械 特 性

Mechanical Specification：

No.	項目 item	規格 standard
5-1	外觀尺寸 Overall Dimensions	见附件 See the drawing
5-2	機構極限角度 Limit angle	210°± 10°
5-3	重量 Weight	17.5 ± 1g
5-4	導線規格 Connector wire gauge	#26 PVC
5-5	導線長度 Connector wire length	250 ± 5 mm
5-6	舵片規格 Horn gear spline	24T
5-7	舵片種類 Horn type	十字，圆盘，大十字，条型 Cross , Disk , Big Cross, Double
5-8	減速比 Reduction ratio	1/378

6. 控 制 特 性
Control Specification：

No.	项目	规格
6-1	控制系統 Control system	改变脉冲宽度 Pulse Width Modification
6-2	放大器種類 Amplifier type	模拟控制器 Analog Controller
6-3	操作角度 Operating travel	45°(在 1000→2000 μsec)
6-4	中立位置 Neutral position	1500 μsec
6-5	脈波訊號虛位 Dead band width	4 μsec
6-6	旋轉方向 Rotating direction	顺时针 (在 1500→2000 μsec) Counterclockwise (when1500→2000 μsec)
6-7	脈波寬度範圍 Pulse width range	800→2200 μsec
6-8	可作動角度範圍 Maximum travel	大约 145°(在 800→2200 μsec) Approx 145°(when800→2200 μsec)