Universidad Miguel Hernández de Elche

MÁSTER UNIVERSITARIO EN ROBÓTICA

"Brazo robótico de 6 grados de libertad de bajo coste para entornos educativos, basado en Arduino y controlado por ARTE"

Trabajo de Fin de Máster

Curso académico
2020/2021

Autor: Adrián Rueda Gómez
Tutor/es: Arturo Gil Aparicio
Luis Miguel Jimenez García

AGRADECIMIENTOS

En primer lugar, me gustaría agradecerle a mi tutor, Arturo Gil, por proponerme este trabajo y animarme a ir un paso más allá con cada avance que iba consiguiendo durante el desarrollo del mismo. Además, me ha permitido meterme más de lleno en el mundo de la robótica, un mundo que me ha apasionado desde pequeño, donde lo que parece complejo es mucho más complejo de lo que aparenta, pero a la vez es algo muy bello por lo que merece la pena esforzarse e ir un paso más allá.

También quiero agradecérselo a mis padres por el apoyo incondicional que me han brindado durante mis años de carrera, de máster y por todo el que me darán a lo largo de los años venideros. Por esos ánimos que siempre me han dado en los peores momentos y que, seguramente, sin ellos, no habría podido llegar hasta este momento.

A mi tutor del TFG, José Antonio, que confió en mí mucho antes de realizar el TFG, por ser un buen mentor y por los ánimos que me dio durante mi etapa de estudiante de grado, y por enseñarme a ser un mejor ingeniero.

Finalmente, y por ello no menos importante, a mi gran amigo, Pablo, que me ha apoyado desde el primer momento con este trabajo, por animarme en los momentos en los que el trabajo me ha superado y por su ayuda a lo largo del mismo. También a mis dos amigos, Dani y Álvaro, por aguantarme en los últimos momentos del desarrollo de este trabajo de fin de máster.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN Y OBJETIVOS

1.1. INTRODUCCIÓN Y JUSTIFICACIÓN DEL TRABAJO

El uso de robots en la industria hoy en día es algo muy extendido, desde que comenzara en los años 60 con el conocido Unimate, de la empresa Unimation [1]. Con este modelo, dio comienzo a una nueva revolución industrial en la que los procesos de fabricación se adaptaron para incluir robots en ellos. En la actualidad, el uso de robots se puede ver en diversos procesos de fabricación, siendo uno de los más conocidos las líneas de producción de vehículos.

Figura 1 - Línea de ensamblaje de vehículos [2]

Este tipo de robots cuenta con motores de gran potencia, pudiendo alcanzar consumos de 500 W por motor y, por tanto, un valor de par elevado, lo que los hace especialmente peligrosos si no se toman las precauciones adecuadas, como instalar celdas que impiden el acceso a la zona de trabajo del robot. Esto, entre otros aspectos, hace que el precio sea muy prohibitivo y se limite a aplicaciones dentro de una factoría.

No obstante, el hecho de que los robots sean peligrosos, unido a que requieran de un espacio amplio para su instalación, así como su elevado precio, hacen que las labores de formación (elemento previo de gran importancia para una correcta y segura operación del robot) se limiten, en muchos casos, a realizar simulaciones en un ordenador, por lo que no se siente la verdadera esencia de trabajar con un robot real.

Por ello, resulta interesante que existan maquetas que sean lo más similar posible a un robot real, las cuales, además de contar con un precio más atractivo, permiten que se cree un ambiente educativo y de formación totalmente seguro. Cabe decir que, aunque cada marca de robot requiera de un tipo de formación distinta para conocer en profundidad su comportamiento, en las asignaturas de robótica de los centros educativos se debería tener un primer contacto con un robot más o menos similar a uno industrial.

Aunque existen muchos tipos de simuladores, como RobotStudio, CoppeliaSim, o ARTE (*A Robotic Toolbox for Education*) [3], que son adecuados para educación, estos pueden presentar una curva de dificultad elevada en algunos casos, lo cual no es un

gran inconveniente si tenemos en cuenta que pueden llegar a complementar el uso del simulador con una maqueta de robot, lo cual sería lo más adecuado. Por ello, en este trabajo se persigue la finalidad de desarrollar un robot que pueda ser utilizado con ARTE.

Este robot, al ser pequeño y de bajo coste, permitiría que todos los estudiantes de una asignatura de robótica pudieran contar con un robot fácilmente manipulable y que así tuvieran un contacto más cercano con un robot real, similar a uno industrial.

Por tanto, tener un entorno seguro con el que aprender a utilizar un robot lo más parecido a uno industrial es una necesidad en cualquier centro de formación donde se impartan asignaturas de robótica.

1.2. OBJETIVOS

El objetivo de este trabajo es desarrollar un brazo articulado de 6 grados de libertad similar a los ya ampliamente utilizados en la industria, con el que los estudiantes de robótica puedan tener un primer contacto cercano a los robots industriales.

El robot aquí propuesto cuenta con un coste bajo y se puede replicar fácilmente, para así poder contar con más unidades o para realizar tareas de reparación en las unidades ya creadas. Por tanto, el material empleado en su fabricación es el plástico y también se emplea tecnología de impresión 3D para tal fin. Además, para agilizar en la medida de lo posible el desarrollo del trabajo, se parte de un robot ya existente de tipo serie y *open source* para así, en caso de ser necesario, modificarlo libremente para que se adecúe a las necesidades y requisitos de este trabajo.

Por otra parte, se diseña una electrónica a medida con la que controlar los motores a través de unos *drivers* de potencia y para que, además, sea conectable a un microcontrolador o microprocesador (Arduino o Raspberry Pi) que ya exista en el mercado. Para la fabricación de la electrónica, es necesario acudir a una empresa externa que pueda fabricar la placa de circuito impreso de manera profesional.

Para la controladora del robot, formada por el microcontrolador o microprocesador y la electrónica diseñada, se escribe un código capaz de gestionar y controlar las comunicaciones y los movimientos del robot.

Adicionalmente, se realiza un estudio de las necesidades dinámicas del robot para seleccionar los actuadores más adecuados que, para este trabajo, serán motores paso a paso.

Además, el robot se controla con la librería ARTE [3] a través del programa MATLAB. Por tanto, el robot de este trabajo se incorpora a la librería ARTE y se realiza el análisis cinemático necesario para su control, además de todas aquellas funciones adicionales necesarias.

En la siguiente tabla se recoge las tareas necesarias para cumplir con el objetivo del trabajo:

ELEMENTOS	OBJETIVOS
Estructura del robot	1) Estudiar distintos robots *open source.* 2) Analizar la estructura del robot seleccionado: a) Realizar modificaciones o rediseñar piezas en el caso de ser necesario. 3) Imprimir las piezas en 3D del robot. 4) Montar los distintos elementos mecánicos y electromecánicos.
Actuadores	1) Realizar un análisis dinámico de las fuerzas ejercidas en cada articulación para seleccionar el motor adecuado. 2) Realizar pruebas de posicionamiento de los actuadores seleccionados para comprobar cómo de precisos son.
Controladora	1) Seleccionar qué microprocesador o microcontrolador utilizar. 2) Diseñar una electrónica para controlar los actuadores: a) Seleccionar los *drivers* adecuados para el control de los motores. b) Diseñar el esquema electrónico y la placa de circuito impreso. b) Fabricar (de forma externa) la placa de circuito impreso. 3) Programar la controladora: a) Programar las comunicaciones entre la controladora y MATLAB. b) Programar el control del robot.
MATLAB	1) Incluir el robot en la librería ARTE: a) Realizar el estudio cinemático del robot. 2) Programar las funciones necesarias para controlar el robot con ARTE a través de MATLAB: a) Gestión de la comunicación entre MATLAB y la controladora del robot. b) Programación de las funciones para controlar el robot.
Sistema completo	1) Comprobar que el robot se comporta conforme lo esperado.

Tabla 1 - Resumen de los objetivos del trabajo

Como se puede observar en la tabla anterior, el trabajo aúna varias disciplinas, como la electrónica, la programación y la mecánica. Además, cabe destacar que el presente trabajo se puede dividir en dos partes claramente diferenciadas. Por un lado, existe un bloque centrado en lo teórico, en donde se analizan las distintas opciones que pueden existir para desarrollar este trabajo, así como realizar las tareas de programación; y, por otro lado, existe un bloque más práctico, donde se fabrican, prueban y montan los distintos componentes.

La programación de MATLAB se realiza en su lenguaje propio, mientras que el lenguaje de programación de la controladora queda definido, junto con el protocolo de comunicación, una vez seleccionado el microcontrolador o microprocesador que se emplea. No obstante, se plantea el uso de C/C++ o Python para la controladora, mientras que para las comunicaciones se plantea el uso del protocolo serie o los protocolos TCP y UDP, protocolos ampliamente utilizados en el ámbito de la robótica junto a CAN.

CAPÍTULO 2

ESTADO DEL ARTE

2.1. ESTADO DEL ARTE

En los años 60, con la creación del conocido Unimate, de la ya desaparecida empresa Unimation (un robot de coordenadas polares, de 5 grados de libertad y accionado por sistemas hidráulicos [1]), se comienza a desarrollar la industria, sobre todo la del automóvil, en donde se emplean robots para realizar tareas repetitivas y, en muchos casos, peligrosas.

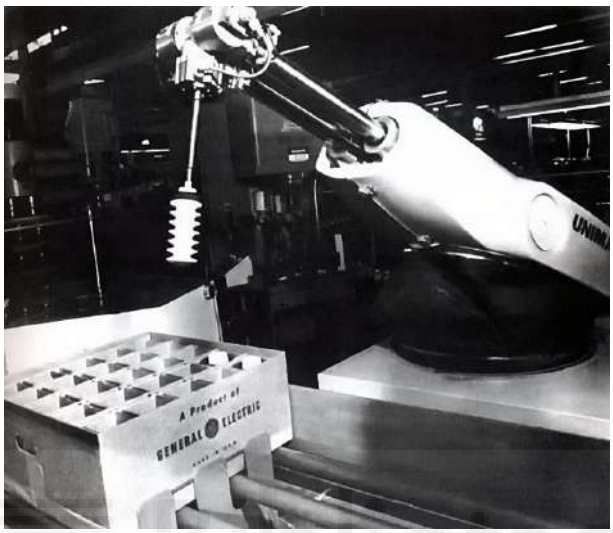


Figura 2 – Robot Unimate [4]

En aquel momento, el Unimate fue una gran revolución, puesto que fue necesario desarrollar tecnologías totalmente novedosas, como sistemas de *encoders* ópticos para cerrar el bucle de control con la controladora y asegurar un buen posicionamiento de las articulaciones. Además, también se creó un sistema de memoria no volátil que, finalmente, se bautizó como Dynastat [1], el cual permitía almacenar hasta un total de 16000 bits de información mediante un sistema de tambor giratorio [5], una cantidad nada despreciable para la época.

Aunque todos estos elementos, y muchos más, hicieron que Unimate creciese como empresa, también fue su ruina, puesto que su tecnología, poco a poco, fue quedando obsoleta frente a las necesidades del mercado. Un ejemplo de ello fue la solicitud de General Motors para que el Unimate contase con motores eléctricos frente a los actuadores hidráulicos que utilizaba por aquel entonces.

Finalmente, Unimation fue vendida a Stäubli en el año 1988, siendo este el fin de una empresa que revolucionó la industria, a la que una serie de malas decisiones llevaron a su desaparición.

Actualmente, todos los robots industriales cuentan con motores sin escobillas, o *brushless,* que permiten alcanzar potencias de más de 500 W por articulación y una precisión mucho mayor que un actuador hidráulico. No obstante, disponer de un robot de estas características no es algo sencillo, pues requieren de un entorno controlado y seguro en el que funcionar, así como una formación previa para saber cómo controlarlo. De aquí aparece la necesidad de que existan robots educativos lo más similares a los robots industriales, pero con la ventaja de que sean seguros de utilizar y, en ciertos

casos, más sencillos para que el estudiante pueda entender mejor el comportamiento de un robot.

Por ello, existen varios robots educativos de bajo coste y reducidas dimensiones con los que los estudiantes de centros educativos pueden tener una primera aproximación a lo que es un robot industrial. De ellos, se pueden destacar el THOR [6], el robot de Dr. D-Flo [7] o el Niryo One [8], siendo este último el elegido como punto de partida del presente trabajo.

Figura 3 - Robot THOR [6]

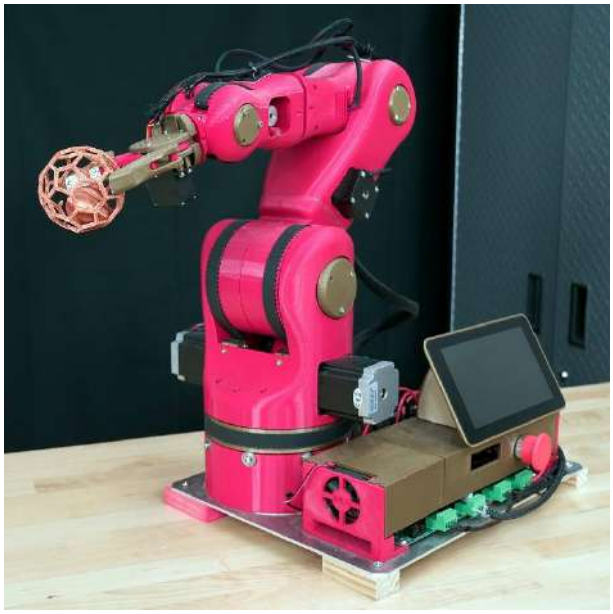


Figura 4 - Robot de Dr. D-Flo [7]

Figura 5 - Niryo ONE

Estos robots, al ser de muy reducidas dimensiones, hacen que sean los suficientemente seguros como para que los estudiantes puedan manipularlos sin correr muchos riesgos. Además, tienen la ventaja de que son sencillos de programar, lo que minimiza la frustración que puede ocasionar desarrollar un código para controlar un robot. Y, finalmente, el precio reducido los hace muy atractivos para ser empleados en centros educativos para la formación de estudiantes y posibles futuros operarios de robots industriales.

2.2. ROBOTS MANIPULADORES DEL TIPO SERIE

Para controlar los robots, es necesario una controladora, ya sea una pequeña y sencilla, como puede ser una Arduino, una Raspberry Pi, o similares, o una de mayor complejidad como la que se puede ver en la Figura 6 - Controladora de robot ABB [9].

Figura 6 - Controladora de robot ABB [9]

En el caso de las controladoras de robots industriales, estas cuentan con todos los elementos necesarios para asegurar el buen funcionamiento del robot, así como para dotar al sistema con los elementos de seguridad precisos. Además, gracias a esto, permiten la conexión de elementos auxiliares, tales como un *teach pendant* o distintas herramientas, para que se adecúen a la tarea que han de realizar.

Cabe decir que muchas empresas cuentan con sus propios actuadores, por lo que la caracterización de ellos queda recogida a la perfección en sus controladoras. Igualmente, la calibración del robot en base a su anatomía también queda recogida y, aunque sean todos iguales dentro del mismo modelo, siempre hay pequeñas diferencias que quedan reflejadas en la configuración. De este modo, se puede asegurar que el comportamiento del robot sea el esperado.

Por otra parte, cada empresa, como ABB, FANUC o KUKA, por citar algunas de ellas, cuentan con lenguajes de programación propios, siendo RAPID el de ABB, KAREL el de FANUC, y KRL el de KUKA.

Además, existen distintos tipos de simuladores, ya sean propietarios, como CoppeliaSim o RobotStudio, con los que se puede realizar un prototipado rápido de una celda de trabajo del robot, pero también la programación final del comportamiento de los movimientos del robot dentro de la celda. De igual manera existen otro tipo de simuladores, que son libres, como puede ser ARTE, que permite la simulación y un prototipado sencillo de una celda de trabajo, aunque por el momento no permite la programación de un robot que no sea simulado.

Finalmente, podemos encontrar distintos tipos de modos de control, como el control en posición y velocidad, donde el robot avanza sin tener en cuenta su entorno más allá de lo establecido en el *path planning*. Además, existen modos de control en fuerza, donde se tienen en cuenta las fuerzas ejercidas durante la trayectoria del robot. Este tipo de control ha permitido el desarrollo de robots colaborativos, un tipo de robots pensados para que pueda haber un operador humano dentro del entorno del robot. Esto es posible gracias a que el robot cuenta con una serie de sensores capaces de medir la fuerza ejercida para que así se detenga en caso de colisión con el operador, para evitar daños.

Por otra parte, también existen los sistemas de control visual o *visual servoing*, en donde una cámara (o conjunto de cámaras), junto a un algoritmo de visión por computador, es capaz de controlar el movimiento del robot en base a unas directrices establecidas, como puede ser seguir un objeto en movimiento o posicionarse en un punto, dado que puede cambiar en el espacio en determinadas ocasiones, por citar algunos ejemplos.

CAPÍTULO 3

CINEMÁTICA DE ROBOTS DE TIPO SERIE

3.1. INTRODUCCIÓN

En este apartado, aunque no se profundiza en cómo se han obtenido los parámetros que definen el robot, pues existe mucha literatura sobre ello, se plasman los resultados obtenidos, pero también se indican los parámetros que se modifican en el archivo necesario de la librería ARTE. Finalmente, también se recogen los resultados para así poder comprobar si los parámetros calculados son correctos.

3.2. CINEMÁTICA DE ROBOTS DE TIPO SERIE

Para obtener la cinemática directa o la cinemática inversa, en primer lugar, se han de determinar los parámetros de Denavit-Hartenberg (DH):

	θ	**d**	**a**	**α**
0 → 1, A_1^0	θ_1	0.183	0	$-\frac{\pi}{2}$
1 → 2, A_2^1	$\theta_2 - \frac{\pi}{2}$	0	0.21	0
2 → 3, A_3^2	θ_3	0	0.03	$-\frac{\pi}{2}$
3 → 4, A_4^3	θ_4	0.2215	0	$\frac{\pi}{2}$
4 → 5, A_5^4	θ_5	0	0	$-\frac{\pi}{2}$
5 → 6, A_6^5	$\theta_6 - \pi$	0.0745	0	0

Tabla 2 - Parámetros de Denavit-Hartenberg

Los valores de la Tabla 2 - Parámetros de Denavit-Hartenberg se han añadido al archivo "**`parameters.m`**" asociado al robot en la librería ARTE, quedando de la siguiente manera:

```
robot.DH.theta= '[q(1) q(2)-pi/2 q(3) q(4) q(5) q(6)-pi]';
robot.DH.d='[0.183 0 0              0.2215 0       0.0745]';
robot.DH.a='[0 0.21 0.03            0      0       0]';
robot.DH.alpha= '[-pi/2 0 -pi/2 pi/2 -pi/2 0]';
```

De este modo, los sistemas de referencia de cada articulación quedan dispuestos tal y como se puede ver en la Figura 7 - Disposición de los sistemas de referencia en el robot.

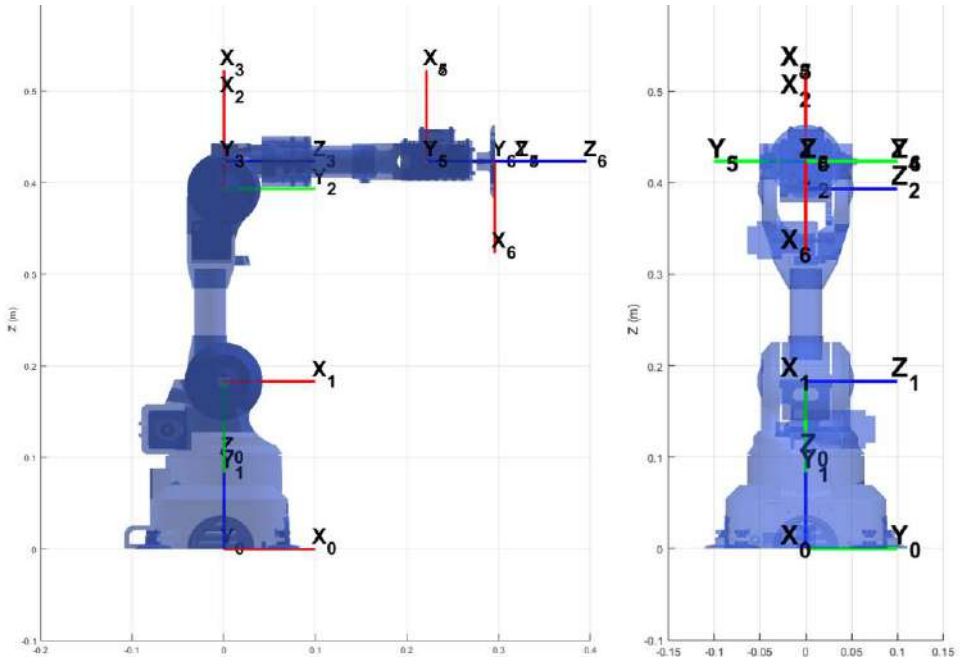


Figura 7 - Disposición de los sistemas de referencia en el robot

No obstante, el robot aún no ha quedado definido por completo, pues falta indicar el recorrido que puede realizar cada articulación. Estos valores se recogen en la Tabla 3 - Ángulos máximos y mínimos alcanzables por cada articulación.

	Ángulo mínimo [º]	**Ángulo máximo [º]**
q_1	-175	175
q_2	-36.7	90
q_3	-80	70
q_4	-175	175
q_5	-40	110
q_6	-147.5	147.5

Tabla 3 - Ángulos máximos y mínimos alcanzables por cada articulación

Al añadir los valores al archivo "**`parameters.m`**", estos quedan de la siguiente manera:

```
robot.maxangle =[deg2rad(-175) deg2rad(175); %Axis 1, minimum, maximum
                 deg2rad(-36.7) deg2rad(90); %Axis 2, minimum, maximum
                 deg2rad(-80) deg2rad(70); %Axis 3, minimum, maximum
                 deg2rad(-175) deg2rad(175); %Axis 4, minimum, maximum
                 deg2rad(-40) deg2rad(110); %Axis 5, minimum, maximum
                 deg2rad(-147.5) deg2rad(147.5)]; %Axis 6, minimum, maximum
```

Adicionalmente, se ha de indicar la velocidad y aceleración máxima a la que puede moverse cada articulación. Puesto que es un robot de dimensiones reducidas, estos valores han de ser también reducidos y, por lo tanto, se han establecido los valores de velocidad y aceleración máximos en la Tabla 4 - Velocidad y aceleración máxima de las articulaciones.

	Velocidad máxima [rad/s]	**Aceleración máxima [rad/s^2]**
q_1	0.75	7.5
q_2	0.75	7.5
q_3	1	10
q_4	1.25	12.5
q_5	1.25	12.5
q_6	1.25	12.5

Tabla 4 - Velocidad y aceleración máxima de las articulaciones

Al añadir los valores al archivo "**`parameters.m`**", estos quedan de la siguiente manera:

```
robot.velmax = [deg2rad(43); %Axis 1, rad/s
                deg2rad(43); %Axis 2, rad/s
                deg2rad(57); %Axis 3, rad/s
                deg2rad(72); %Axis 4, rad/s
                deg2rad(72); %Axis 5, rad/s
                deg2rad(72)];%Axis 6, rad/s

robot.accelmax=robot.velmax/0.1;
```

Adicionalmente, existen más parámetros que se pueden configurar, los cuales se pueden consultar en el anexo 9.2.1 - parameters.m.

No es objeto de este trabajo explicar cómo se obtienen la cinemática directa o la cinemática inversa, ni cómo son los distintos métodos que existen para calcular la cinemática inversa, pero los algoritmos para calcularlas se pueden consultar en los anexos 9.2.2 - directkinematic.m y 9.2.3 - inversekinematic_niryo.m.

Cabe mencionar que la cinemática directa es obtener la posición del extremo del robot en el espacio conocidas las posiciones articulares; mientras que la cinemática inversa es calcular las posiciones articulares del robot conocida la posición del extremo en el espacio.

Para comprobar que los parámetros DH, así como que el robot está bien añadido a la librería, se han ejecutado los siguientes comandos:

```
robot = load_robot('NIRYO')
```

La finalidad de este comando es la de cargar el robot en el *workspace* para poder trabajar con él. A continuación, se ha de crear un vector columna de 6 elementos, los cuales son aleatorios y los cuales hacen referencia a las diferentes posiciones articulares del robot. Para ello, se debe emplear el siguiente comando:

```
q = rand(6, 1)
```

```
q =

    0.2785
    0.5469
    0.9575
    0.9649
    0.1576
    0.9706
```

Ahora, se ha de obtener la matriz de transformación de la cinemática directa. Para ello, se ha de ejecutar el siguiente comando:

```
T = directkinematic(robot, q)
```

```
T =

    0.0851    0.9947   -0.0582    0.1436
    0.9900   -0.0778    0.1175    0.0510
    0.1124   -0.0676   -0.9914    0.0695
         0         0         0    1.0000
```

De esta matriz se puede obtener la posición en el espacio del efector final del robot, siendo los elementos T[1, 4], T[2, 4] y T[3, 4] las coordenadas X, Y y Z, respectivamente, del efector final.

Para calcular la cinemática inversa, se ha de ejecutar el siguiente comando:

```
qinv = inversekinematic(robot, T)
```

Este comando calcula las 8 posibles soluciones de la cinemática inversa de un robot serie. Además, si la cinemática inversa es correcta, al menos una de las soluciones ha de coincidir con el vector q obtenido anteriormente.

```
qinv =

  0.2785   0.2785   0.2785   0.2785  -2.8631  -2.8631  -2.8631  -2.8631
  0.5469   0.5469   3.0992   3.0992  -3.0992  -3.0992  -0.5469  -0.5469
  0.9575   0.9575   2.4533   2.4533   0.9575   0.9575   2.4533   2.4533
 -2.1767   0.9649  -2.9649   0.1767   0.2451  -2.8965   0.3542  -2.7874
 -0.1576   0.1576  -2.3175   2.3175  -2.5810   2.5810  -0.3811   0.3811
 -2.1710   0.9706  -1.0913   2.0503  -1.0032   2.1384  -1.5426   1.5990
```

No obstante, que una de las soluciones coincida con el vector q no es garantía de que realmente la cinemática inversa esté bien calculada. Para asegurarse de que es correcta, si se calcula la cinemática directa de cada una de las posibles soluciones, la matriz T calculada debe coincidir para cualquiera de las 8 soluciones. Por ejemplo, si se calculan para las soluciones 1, 3 y 6, se obtienen las siguientes matrices T:

- **Cinemática directa para la solución 1:**

```
T = directkinematic(robot, qinv(:, 1))

T =

    0.0851    0.9947   -0.0582    0.1436
    0.9900   -0.0778    0.1175    0.0510
    0.1124   -0.0676   -0.9914    0.0695
         0         0         0    1.0000
```

- **Cinemática directa para la solución 3:**

```
T = directkinematic(robot, qinv(:, 3))


T =

    0.0851    0.9947   -0.0582    0.1436
    0.9900   -0.0778    0.1175    0.0510
    0.1124   -0.0676   -0.9914    0.0695
         0         0         0    1.0000
```

- **Cinemática directa para la solución 6:**

```
T = directkinematic(robot, qinv(:, 6))

T =

    0.0851    0.9947   -0.0582    0.1436
    0.9900   -0.0778    0.1175    0.0510
    0.1124   -0.0676   -0.9914    0.0695
         0         0         0    1.0000
```

Como se puede observar, las matrices de transformación coinciden en todos los casos, por lo que se puede asegurar que la cinemática inversa está bien calculada. Además, si se dibuja el robot, primero, en la posición del vector q, Figura 8 - Posición q, y en la posición de las soluciones 1, Figura 9 - Posición para solución de la cinemática inversa 1, 3, Figura 10 - Posición para solución de la cinemática inversa 3, y 6, Figura 11 - Posición para la solución de la cinemática inversa 6, se obtienen las siguientes representaciones de las soluciones:

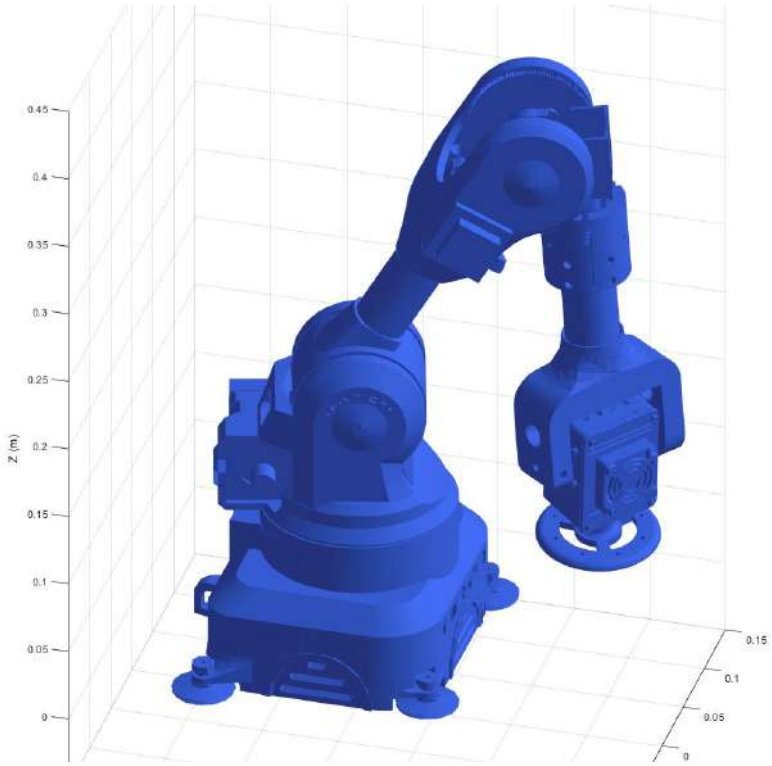


Figura 8 - Posición q

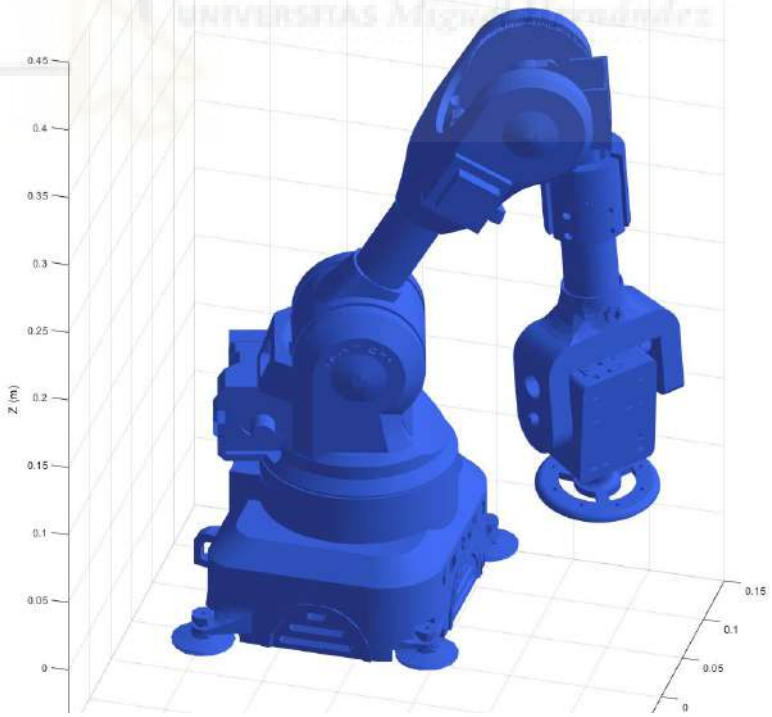


Figura 9 - Posición para solución de la cinemática inversa 1

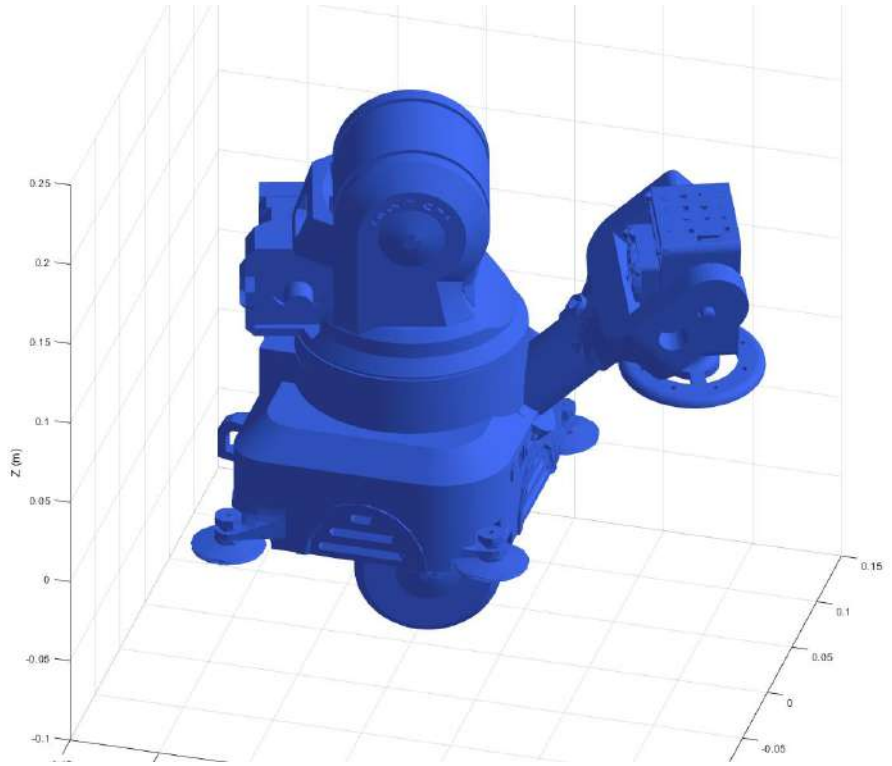


Figura 10 - Posición para solución de la cinemática inversa 3

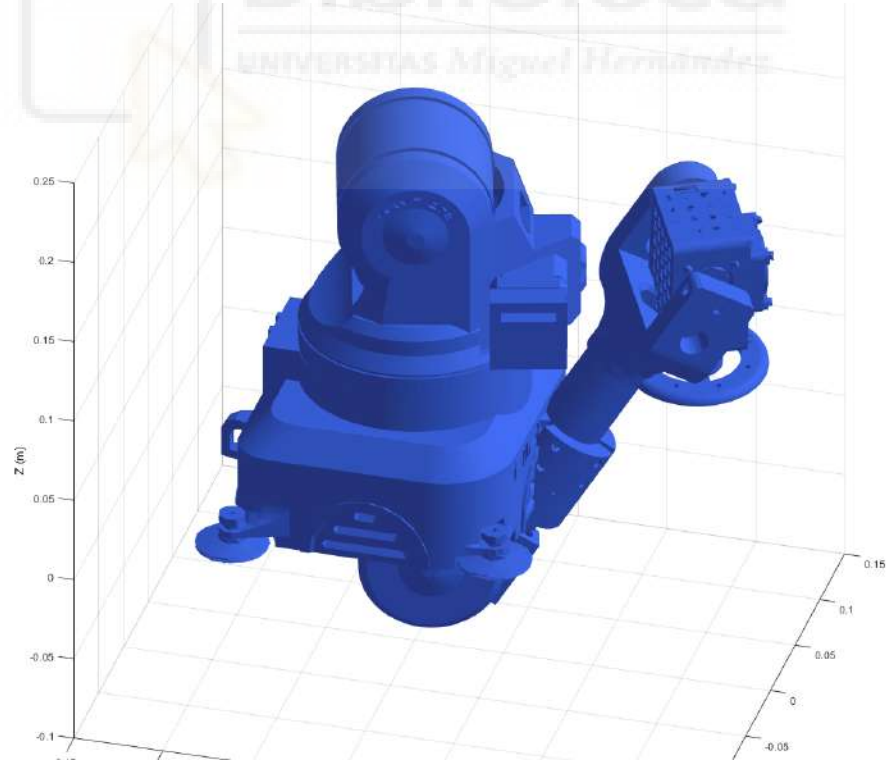


Figura 11 - Posición para la solución de la cinemática inversa 6

Como se puede apreciar, han aparecido dos soluciones que son físicamente imposibles, ya que el robot colisiona consigo mismo. Esto no significa que la cinemática inversa sea errónea, sino que ha encontrado una solución válida para que el robot esté en la posición deseada, pero es físicamente inalcanzable.

No obstante, todo lo ya expuesto acerca de la cinemática inversa solo es válido siempre y cuando la muñeca sea esférica, es decir, que los ejes q4 y q5 se corten en el espacio. Si, por lo contrario, los ejes no se cortan, sino que se cruzan, se trata de una muñeca no esférica. Un ejemplo de robot con muñeca no esférica es el UR3, de Universal Robots, Figura 12 - UR3 con muñeca no esférica.

Figura 12 - UR3 con muñeca no esférica

En el caso del robot de este trabajo, en un primer momento contaba con una muñeca no esférica, lo que requería de un algoritmo iterativo con el que, a partir de una semilla, calculase la cinemática inversa del robot. Sin embargo, debido a que se ha rediseñado la muñeca, la versión final presentada cuenta con una de tipo esférica, lo que simplifica, en gran medida, el cálculo de la cinemática inversa al ser un cálculo directo.

CAPÍTULO 4

ELEMENTOS DEL SISTEMA

4.1. DESCRIPCIÓN GENERAL DEL SISTEMA

Se trata de un robot de dimensiones reducidas y de bajo coste, el cual está pensado para emplearse en un entorno educativo junto a la librería ARTE. El sistema está formado por el robot, una controladora, una fuente de alimentación y un ordenador.

El robot está fabricado, en su gran mayoría, mediante impresión 3D de plástico PLA. Para su movimiento se han empleado motores paso a paso NEMA 17 y NEMA 14, según el espacio disponible para ello, junto a correas y poleas GT2. Además, cuenta con un motor con reductora en la articulación q2, u hombro, para poder soportar el peso del robot en las condiciones más desfavorables planteadas en el momento del diseño. También se han empleado distintos tipos de rodamientos y elementos de tornillería para su construcción, siendo todos elementos normalizados. Para la herramienta se ha utilizado un servo de radiocontrol por sus reducidas dimensiones y alto par a bajo voltaje. Igualmente, los motores son refrigerados con ventilación forzada con una serie de ventiladores sujetados a dichos motores.

La controladora del robot está formada por una Arduino MEGA junto a una placa o *shield,* sobre la Arduino MEGA que cuenta con una serie de controladores, o *drivers*, que se encargan de darle la alimentación adecuada a los motores. Para programar la Arduino MEGA se ha utilizado C++ dentro del IDE de Arduino. La comunicación entre la controladora y el ordenador se realiza mediante el protocolo de comunicaciones serie, Figura 13 - Esquema de conexiones entre el PC, la controladora y los distintos elementos del robot.

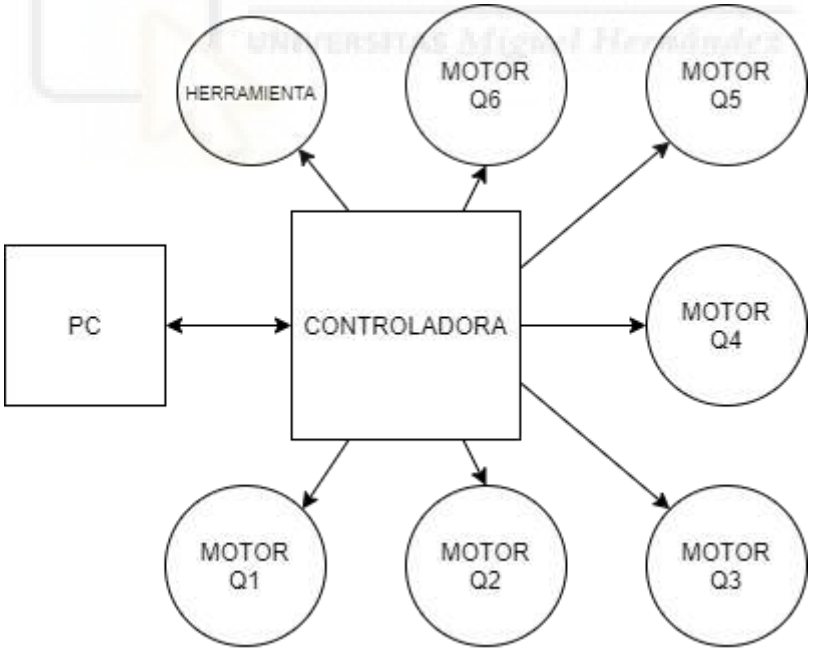


Figura 13 - Esquema de conexiones entre el PC, la controladora y los distintos elementos del robot

Para alimentar la electrónica de la controladora, así como los distintos motores y ventiladores del robot, se ha utilizado una fuente de alimentación ATX de 600 W, de la que se extraen 12 V, dejando el resto de las tensiones disponibles sin utilizar para posibles futuros usos.

El ordenador cuenta con un sistema operativo capaz de ejecutar MATLAB junto a la librería ARTE, donde se ha incluido el robot de este trabajo, y se han añadido una serie de funciones capaces de controlar el robot mediante una comunicación serie. Además, con ARTE se puede simular la posición del robot antes de llevarlo a la posición deseada.

4.2. SELECCIÓN DE ELEMENTOS DEL SISTEMA

Seleccionar los elementos adecuados es crucial para el buen funcionamiento del robot. Por ello, en los siguientes apartados se comentan los distintos elementos que se han seleccionado, así como la justificación para ello. Se habla de la controladora seleccionada, los actuadores que se han empleado en el robot y sobre los distintos elementos estructurales y de montaje empelados.

4.2.1. Selección de la controladora del robot

Para diseñar la controladora del robot se planteó utilizar un microcontrolador o un microprocesador, siendo Arduino (Figura 15 - Arduino MEGA 2560) y Raspberry, (Figura 14 - Raspberry Pi 4 Model B) las principales opciones para ello. Independientemente de la opción elegida, se fijaron unos requisitos con los que seleccionar uno u otro:

- Disponibilidad suficiente de entradas y salidas digitales.
- Potencia de cálculo suficiente para recibir información y realizar el control de los motores al mismo tiempo.
- Facilidad de programación para implementar el código de la controladora, así como para poder realizar el mantenimiento del código.
- Posibilidad de conectarse a un PC con el que comandar el robot.

Figura 14 - Raspberry Pi 4 Model B

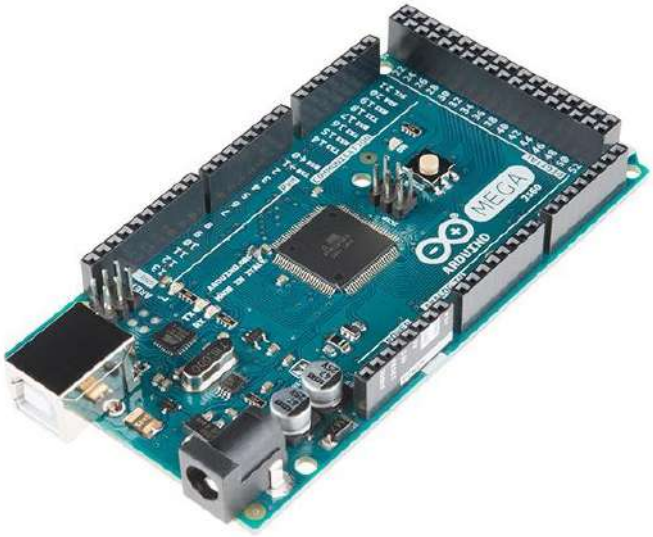


Figura 15 - Arduino MEGA 2560

Atendiendo a la manera de comunicarse con un ordenador, Raspberry ofrece conexión wifi, Bluetooth y un puerto ethernet; mientras que Arduino cuenta, únicamente con comunicación serie, ya sea mediante el puerto USB o conexión directa a la UART (*Universal Asynchronous Receiver-Transmitter*) utilizando los puertos específicos para ello.

Por una parte, para esta aplicación se descartaron conexiones inalámbricas por motivos de seguridad, por lo que quedan la conexión por ethernet de Raspberry y la comunicación serie de Arduino. Aunque en el ámbito de la robótica el uso de conexiones P2P (*Peer-to-Peer*) está muy extendido y sería una opción muy interesante para desarrollar el robot, esto complicaría el desarrollo del código al existir la necesidad de implementar un cliente y un servidor para las comunicaciones; mientras que la conexión serie de Arduino solo requiere comunicarse mediante un puerto serie, lo que simplifica, en gran medida, la programación de la controladora.

Por otra parte, Raspberry permite implementar códigos que hagan el uso de hilos, por lo que se puede tener las comunicaciones en un hilo, mientras que el control de los motores se implementa en otro hilo. No obstante, aunque es un aspecto que se debe tener en cuenta para una futura mejora, complica en exceso el desarrollo y el mantenimiento del código. Arduino, por su parte, al realizar una ejecución secuencial del código simplifica el desarrollo, aunque se puede perder parte del control del robot.

Atendiendo a la potencia de cálculo de una Arduino o de una Raspberry, no debe haber ningún problema para este trabajo. En cuanto a la disponibilidad de entradas y salidas digitales, Raspberry cuenta con un total de 40 pines repartidos entre pines de alimentación, I²C, GPIO (*General Purpose Input Output*), entre otros; mientras que, una Arduino MEGA 2560, cuenta con más de 50 pines de entradas y salidas digitales, comunicaciones y entradas y salidas de señales analógicas.

Finalmente, tras ver las ventajas y desventajas de una u otra, se ha optado por utilizar una combinación de ambas para aprovechar todas características que ofrecen una y otra, pero se descartó pues dificultaría en exceso el desarrollo de la controladora, además de incrementar el precio final del robot. Por tanto, se ha elegido una Arduino

MEGA 2560 en parte por la disponibilidad de pines, así como por la simplicidad de programación y comunicación con un ordenador.

4.2.2. Selección de los *drivers* para controlar los motores

Las salidas digitales de Arduino tienen un límite de 5 V y 40 mA, aunque se recomienda un máximo de 20 mA. Estos valores son insuficientes para alimentar la gran mayoría de motores, además de que si se excede el valor máximo de intensidad, se puede dañar el microcontrolador de Arduino. Estas limitaciones hacen necesario que haya un elemento intermedio que pueda proporcionar la alimentación adecuada a los motores.

Se compararon los distintos *drivers* que ofrece la empresa Pololu, Tabla 5 - Comparación de distintos drivers de Pololu [10].

Modelo de driver	**A4988**	**DRV8825**	**DRV8834**	**DRV8880**	**MP6500**
Voltaje de funcionamiento mínimo	8 V	8.2 V	2.5 V	6.5 V	4.5 V
Voltaje de funcionamiento máximo	35 V	45 V	10.8 V	45 V	35 V
Corriente máxima por fase	1 A	1.5 A	1.5 A	1 A	1.5 A
Corriente pico por fase	2 A	2.2 A	2 A	1.6 A	2.5 A
Microstepping máximo	1/16	1/32	1/32	1/16	1/8

Tabla 5 - Comparación de distintos drivers de Pololu

Observando las características de los distintos *drivers*, se ha seleccionado el DRV8825 y el DRV8834, ya que sus parámetros de funcionamiento son similares. Finalmente, se ha optado por el DRV8825 por los siguientes motivos:

- El voltaje de funcionamiento máximo del DRV8825 es de 45 V frente a los 10.8V del DRV8834, por lo que se puede utilizar con motores de 12 V.

- Tanto el DRV8825, como el DRV8834 tienen el mismo valor de corriente máxima por fase (1.5 A), pero el DRV8825 aguanta mayor corriente de pico.

Además, ambos *drivers* soportan un *microstepping* máximo de 1/32 de paso, lo que permite realizar movimientos más precisos y suaves. No obstante, se podría haber optado por el MP6500, que posee características eléctricas muy similares, pero solo alcanza un *microstepping* de 1/8 de paso, por lo que se ha descartado su uso.

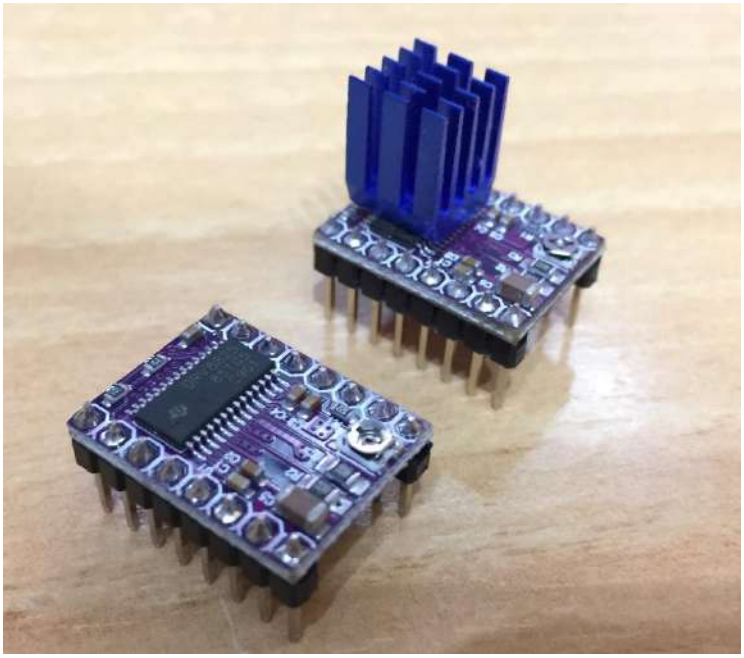


Figura 16 - Breakout board DRV8825 - Con y sin disipador de calor

Una vez seleccionados los *drivers* que se van a emplear (Figura 16 - Breakout board DRV8825 - Con y sin disipador de calor), se planteó utilizar una *breakout board* que ya lleve montado el *driver* junto a la electrónica necesaria, o diseñar esa *breakout board* sobre la PCB. Finalmente se optó por utilizar una *breakout board* por simplicidad en el diseño y por motivos de mantenimiento (sustitución del *driver* en caso de rotura).

Por otra parte, el fabricante recomienda el uso de condensadores externos para filtrar el posible ruido de la señal de alimentación del motor, por lo que se ha añadido un condensador electrolítico y un condensador cerámico en paralelo para tal fin. Además, se ha previsto el caso en el que un motor necesite una tensión de alimentación distinta de 12 V, por lo que se han añadido unos terminales para alimentarlo de forma externa, así como un selector de alimentación mediante el cortocircuito de dos terminales por medio de un *jumper*.

4.2.3. Selección de actuadores

La selección de los motores es uno de los aspectos más críticos a la hora de diseñar un robot, pues éstos son los músculos del mismo y son los encargados de realizar los movimientos deseados. Por ello, se ha prestado especial atención a su selección.

Normalmente, en un brazo industrial se emplean motores sin escobillas o *brushless* de corriente continua, esencialmente son motores síncronos de imán permanente. Para este tipo de motores se utilizan unos *drivers* capaces de realizar el control de posición y velocidad o del par ejercido. Además, este tipo de motores cuentan con un *encoder* incremental con el que realizar el control de posición.

Esta solución, aunque muy extendida en la industria y altamente interesante, no tiene un precio asequible para un trabajo de estas características, por lo que se ha

decidido descartarla. Por tanto, para este diseño, se ha optado por utilizar motores paso a paso.

Asimismo, el diseño original del robot contaba con una combinación de motores paso a paso en las articulaciones q1, q2 y q3; pero también empleaba servomotores en las articulaciones q4, q5 y q6. No obstante, para este trabajo se han sustituido los servomotores por motores paso a paso. De este modo, se ha buscado conseguir un diseño más homogéneo a la hora de realizar modificaciones en el diseño original del robot, pero también a la hora de programar el comportamiento de los motores.

4.2.3.1. Diferencias entre motores paso a paso y servomotores

Aunque un motor paso a paso y un servo puedan ser parecidos, existen una serie de diferencias entre ellos. No obstante, hay que diferenciar entre los servos (que son similares a los motores paso a paso) y los servos RC utilizados en el ámbito del radio control y, en muchas ocasiones, en brazos robóticos similares, pero a niveles técnicos mucho más reducidos que los que se presentan en este trabajo. Por lo tanto, estos últimos quedan descartados de la comparativa.

De esta manera, se puede decir que un servo (Figura 17 - Servo industrial Omron) está compuesto por un elemento electromecánico (el motor), una serie de sensores de posicionamiento y un *driver* o controlador. Por su parte, un motor paso a paso está compuesto, del mismo modo, por un elemento electromecánico y un *driver* o controlador.

Figura 17 - Servo industrial Omron

Aunque puede parecer que son dos elementos casi idénticos, el motor paso a paso tiene la ventaja de que todo el rango de velocidades está libre de vibraciones. Además, los motores paso a paso permiten mantener una posición fija y constante, y necesitan una menor tensión de alimentación en las bobinas.

Por su parte, los servos, pueden alcanzar mayores velocidades mientras mantienen el par respecto a la velocidad. No obstante, el precio de un servo, por lo general, suele ser mucho más elevado que el de un motor paso a paso, por lo que los hacen inviables para este trabajo. [11]

Finalmente, aunque los servos parezcan una solución más adecuada, se ha optado por utilizar motores paso a paso, ya que sus propiedades son adecuadas para este trabajo. Además, el precio es mucho menor, por lo que se ajusta a uno de los objetivos finales del trabajo, y es que el robot sea de bajo coste.

4.2.3.2. Diferencia entre los distintos motores paso a paso existentes

Una vez decidido que se van a usar motores paso a paso (PaP), aún era necesario determinar si se utilizarían motores bipolares, unipolares o de reluctancia variable. Los últimos se descartaron por no ser tan habituales como los bipolares y unipolares, por lo que se estudió con mayor detenimiento estos.

4.2.3.2.1. Motores unipolares

Este tipo de motores cuenta con 2 bobinados del que parte un cable común que divide a cada bobinado en 2, creando un total de 4 bobinas parciales, tal y como se puede ver en la Figura 18 - Esquema de funcionamiento de un motor PaP unipolar. Este tipo de motores se caracteriza por tener de 5 a 6 cables, 4 correspondientes a los extremos de las bobinas; y 1 o 2 para los cables comunes.

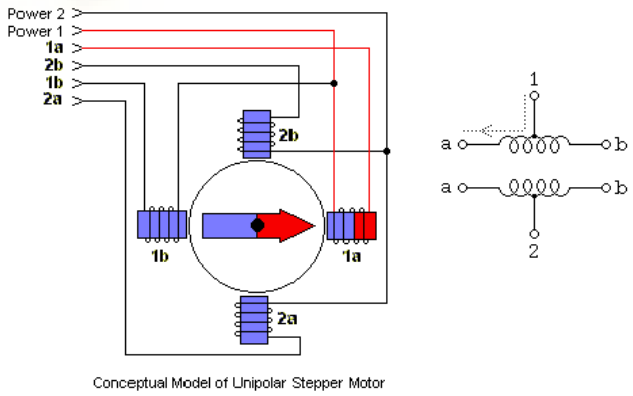


Figura 18 - Esquema de funcionamiento de un motor PaP unipolar

Para esta configuración de los embobinados, el cable común se alimenta, mientras que los extremos de las bobinas se conectan a tierra de manera secuencial. De este modo, se alimenta media bobina (o una bobina parcial) generando un único polo que atrae al polo opuesto del rotor [12].

4.2.3.2.2. Motores bipolares

Este tipo de motores, a diferencia de los unipolares, cuenta también con dos bobinados, pero sin un punto intermedio. Así, se caracteriza por tener tan solo 4 cables correspondientes a los extremos de cada una de las bobinas, tal y como se puede observar en la Figura 19 - Esquema de funcionamiento de un motor PaP bipolar.

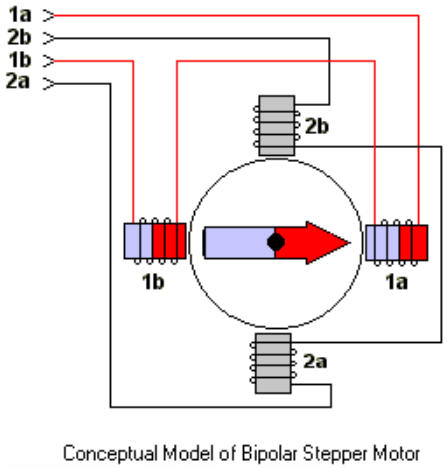


Figura 19 - Esquema de funcionamiento de un motor PaP bipolar

En los motores bipolares, el embobinado se alimenta al completo y genera un par de polos norte-sur que atraen los polos opuestos del rotor. A diferencia de los motores unipolares, la dirección de giro de estos motores radica en que los motores bipolares dependen de la dirección del flujo eléctrico con el que se alimentan las bobinas, mientras que, en los motores unipolares, la dirección del rotor depende de la secuencia con la que se vayan excitando las distintas bobinas parciales.

Una vez visto el funcionamiento básico de cada uno de los motores, se presentan una serie de ventajas e inconvenientes que se muestran, a continuación, en la Tabla 6 - Comparativa entre motores unipolares y bipolares.

Motores unipolares	Motores bipolares
Menor par	Mayor par
Menor par de retención (*holding torque*)	Mayor par de retención (*holding torque*)
Control de la excitación de los bobinados simple	Control de la excitación de los bobinados complejo
Mayores dimensiones y peso	Menores dimensiones y peso

Tabla 6 - Comparativa entre motores unipolares y bipolares

El aspecto más importante a la hora de seleccionar un tipo u otro de motor es el par que pueden ejercer, siendo necesario tener un valor de par elevado en algunas articulaciones, como puede ser q2 y q3. Del mismo modo, es muy importante que el

holding torque (par resistente que evita que el rotor gire estando el motor alimentado) sea elevado para que el robot no se mueva por su propio peso mientras esté esperando una orden. Por ello, se han elegido, además, los motores bipolares, ya que van a añadir menor peso a la estructura del brazo.

Finalmente, el control de los motores no era un aspecto determinante en cuanto a la selección, ya que las propiedades mecánicas eran más importantes. Los aspectos del control se pueden ver en el apartado 5.5.2.2 - Control de los motores.

4.2.4. Elementos auxiliares

A continuación, se comentan los distintos tipos de elementos auxiliares que se han empleado en el desarrollo de este trabajo.

4.2.4.1. Tornillería, rodamientos y elementos transmisores del movimiento

Para este trabajo, donde se busca que replicar el robot sea una tarea sencilla, se emplean elementos estándar. Por un lado, la tornillería utilizada es de métricas estándar, con sus tuercas y arandelas correspondientes. Por otro lado, los rodamientos empleados son fáciles de encontrar en cualquier establecimiento especializado.

Del mismo modo, las poleas y correas utilizadas son fáciles de encontrar pues son ampliamente utilizadas en el ámbito de la impresión 3D.

4.2.5. Impresora 3D y plástico PLA

El avance en la tecnología de impresión 3D supone una gran ventaja a la hora de fabricar piezas y prototipos. Esto es debido a su flexibilidad y los cortos tiempos de fabricación que se requieren para crear cualquier pieza. Además, el bajo coste de los materiales hace que sea un método de fabricación y producción altamente interesante.

Todas las piezas plásticas del robot se han fabricado utilizando tecnología FDM (*Fused Deposition Modeling*) mediante una impresora 3D modelo Prusa i3, Figura 20 - Impresora 3D FDM modelo Prusa i3. Además, el plástico utilizado ha sido el PLA (Polylactic Acid). El PLA tiene ciertas ventajas frente a otros plásticos muy utilizados en los procesos de impresión 3D, como puede ser el ABS, que presenta propiedades físicas y mecánicas similares que el PLA, pero para el caso de este trabajo, donde se han impreso piezas voluminosas, o con gran cantidad de detalles, y que requieran de un tiempo de impresión elevado, el PLA es el material más adecuado. [13]

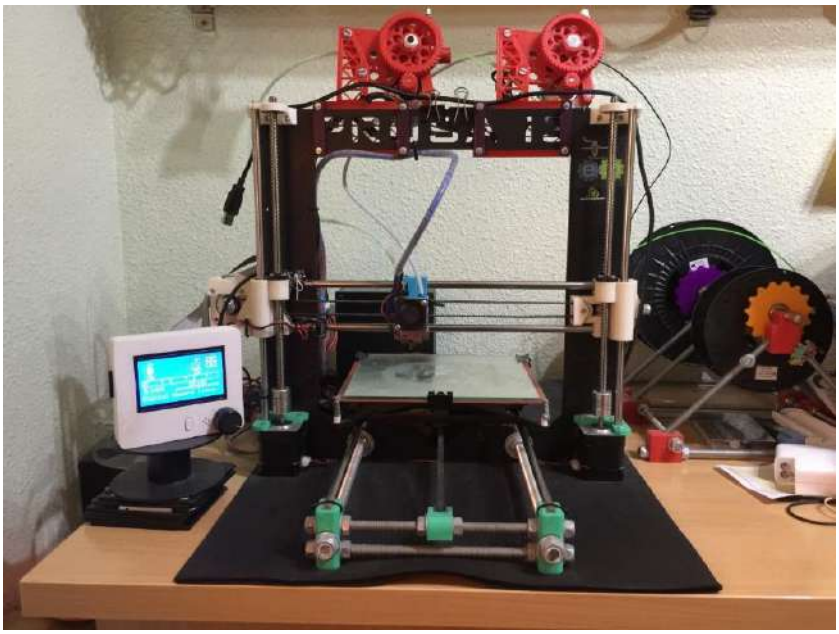


Figura 20 - Impresora 3D FDM modelo Prusa i3

Por una parte, la temperatura de impresión necesaria para el PLA es de alrededor de 190-220 °C, frente a los 210-250 °C que requiere el ABS. Por otra parte, el ABS es muy sensible a las variaciones de temperatura debidas a la diferencia de temperatura entre la cama caliente y la boquilla o *hot-end*, o a las corrientes de aire. Esto puede provocar que la pieza se despegue de la cama caliente (la base sobre la que se imprimen las piezas) y acabe la pieza deformada en la base, lo que se conoce como *warping*. Además, esta sensibilidad a los cambios de temperatura puede provocar que las capas de plástico no se fundan bien unas con otras y acaben por separarse.

El PLA es más estable en cuanto a sus propiedades térmicas, pero no significa que estos problemas sean inexistentes, sino que el PLA no es tan propenso a sufrir los problemas del ABS.

Por otra parte, el ABS exige el uso de adhesivos en la cama caliente, generalmente, laca para el pelo, y que, además, la cama caliente se ajuste a una temperatura de entre 100-110 °C. El PLA, por el contrario, no requiere de adhesivos ni que se caliente la cama, aunque sí que es muy recomendable para evitar posibles impresiones fallidas debido a una mala adherencia a la cama caliente el uso de adhesivos (laca para el pelo), así como calentar la cama a una temperatura de entre 40-60 °C.

CAPÍTULO 5

DISEÑO, IMPRESIÓN, PROGRAMACIÓN Y CONTROL

5.1. REDISEÑO DE LAS PIEZAS

Figura 21 - Niryo ONE original

Este brazo articulado parte del diseño original del NIRYO ONE [8], Figura 21 - Niryo ONE original, el cual cuenta con motores paso a paso en las articulaciones q1, q2 y q3; y servomotores en las articulaciones q4, q5 y q6. El uso de servomotores en las articulaciones q4, q5 y q6 se debería a un fin puramente funcional donde se buscaría aligerar el peso del extremo del robot.

No obstante, para este trabajo, se busca homogeneizar el diseño del robot, por lo que se ha decidido utilizar únicamente motores paso a paso. Esto lleva consigo una serie de modificaciones, no solo en cuanto a los motores se refiere, sino que es necesario rediseñar múltiples piezas para ajustarse a los nuevos motores, así como reforzarlas para soportar el mayor peso de los motores.

5.1.1. Rediseño de la articulación q4 (codo y antebrazo)

La primera pieza cuyo rediseño fue necesario es el codo (Figura 22 - Modelo CAD del diseño original de la articulación q4). Ésta estaba pensada para utilizar un servo XL430 de la marca DYNAMIXEL, junto a un rodamiento de tipo axial en la parte delantera para facilitar el movimiento entre las articulaciones q3 y q4.

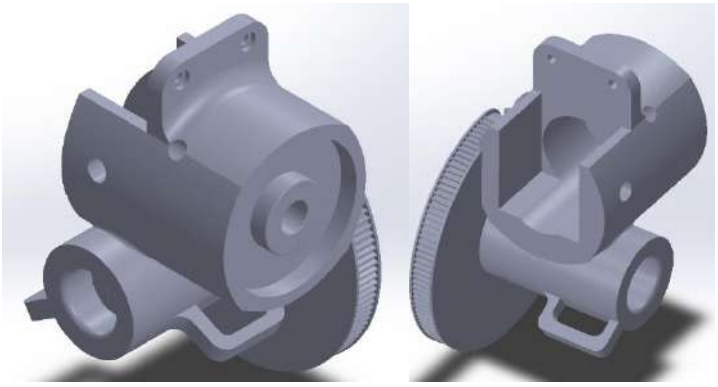


Figura 22 - Modelo CAD del diseño original de la articulación q4

Para el nuevo diseño, se eliminaron por completo todos los elementos excepto el disco dentado para la correa. A partir de ahí, partiendo de las medidas originales, se creó un nuevo bastidor para albergar un motor NEMA 14. Además, se dispusieron varios canales para permitir el paso de los cables de los motores de las articulaciones q5, q6 y para los cables del servo de la herramienta (Figura 23 - Modelo CAD del diseño final de la articulación q4).

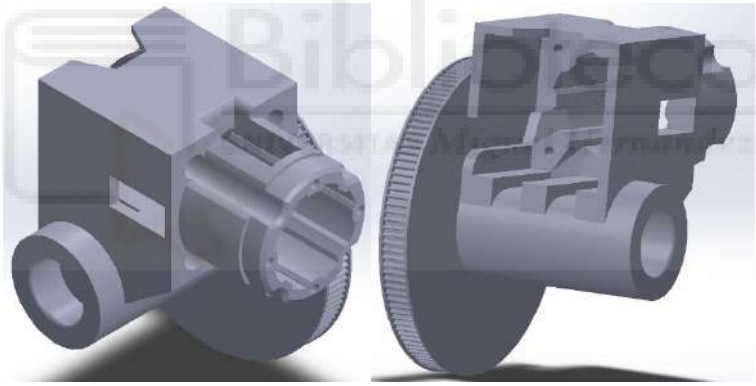


Figura 23 - Modelo CAD del diseño final de la articulación q4

Adicionalmente, para facilitar el montaje del siguiente eslabón, se añadieron unas aberturas con las que acceder fácilmente al eje del motor de la articulación q4, tal y como se puede ver en Figura 23 - Modelo CAD del diseño final de la articulación q4.

Igualmente, este robot, así como otros robots educativos que se barajaron para este trabajo, contaban con un diseño similar en esta articulación, donde el punto de unión es únicamente el eje del motor. Por tanto, para evitar un futuro fallo en la articulación, como se puede ver en la Figura 24 - Robot de Dr. D-Flo [7], en donde se aprecia una grave desalineación entre los eslabones, se ideó un diseño para la articulación que fuera más robusto.

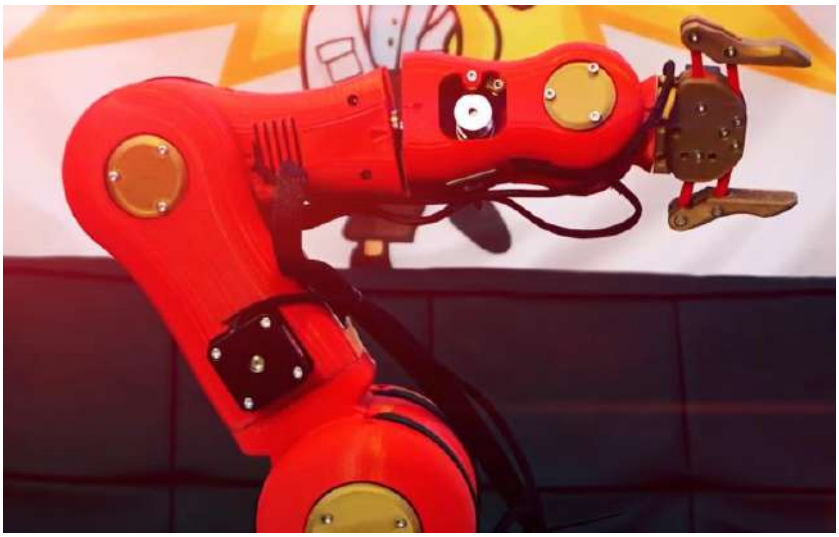


Figura 24 - Robot de Dr. D-Flo

Uno de los diseños que se planteó, y que se ha empleado para la pieza final, cuenta con un rodamiento (61807-2RS) sobre el que apoyan una serie de “pétalos” sujetos al eje de la articulación q4. Asimismo, cuenta con un acople flexible de 5 mm a 8 mm para unir el eje del motor con un vástago sujeto al antebrazo. Este acople, además de transmitir el movimiento de giro del motor, compensa las posibles desalineaciones entre el eje del motor y el vástago del antebrazo. El prototipo de la articulación se puede ver en detalle en la Figura 25 - Vista seccionada en CAD del prototipo de la muñeca y la primera prueba del diseño en la Figura 26 - Prototipo de la articulación q4, en donde se pueden apreciar varios elementos internos del diseño, como el rodamiento y el acople para el motor y el eje.

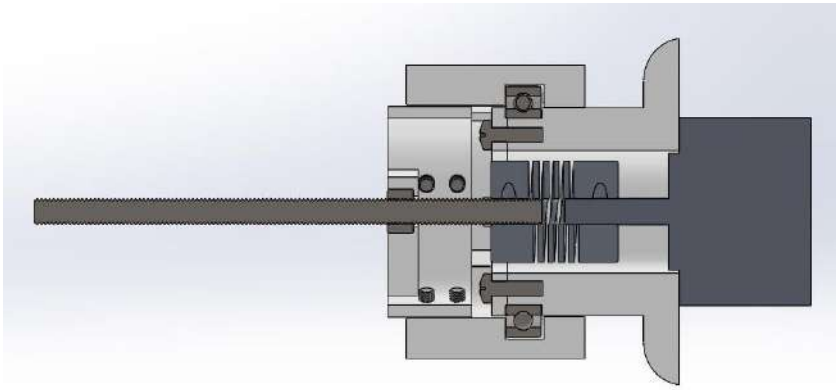


Figura 25 - Vista seccionada en CAD del prototipo de la muñeca

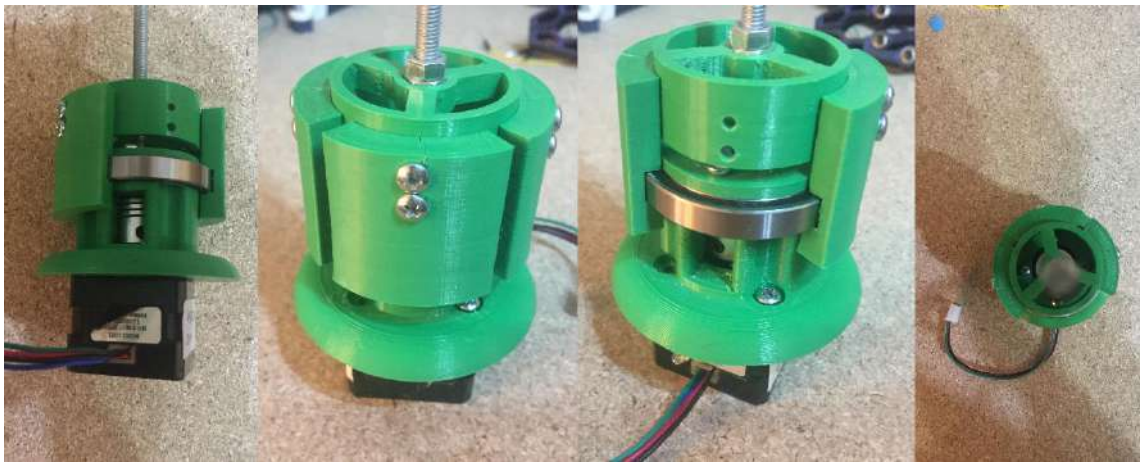


Figura 26 - Prototipo de la articulación q4

Una vez comprobado que el diseño era viable, se trasladó el prototipo a la pieza de la articulación del codo. Como se puede ver en la Figura 27 - Diseño implementado del antebrazo en el robot, se ha seguido la misma premisa que en el prototipo. Además, durante las pruebas para verificar que gira correctamente se ha podido comprobar que su funcionamiento era el esperado.

Figura 27 - Diseño implementado del antebrazo en el robot

5.1.2. Rediseño de la muñeca

El diseño original de la muñeca (Figura 28 - Diseño original de la muñeca) contaba con servomotores de la marca DYNAMIXEL, pero, teniendo en cuenta los objetivos de este trabajo, donde se buscaba utilizar únicamente motores paso a paso, se decidió finalmente rediseñar la muñeca al completo.

Un aspecto importante del diseño original, además de utilizar servos, es que es una muñeca no esférica (los ejes q5 y q6 se cruzan en el espacio, sin llegar a cortarse). Con el objetivo de conseguir un diseño compacto, se planteó hacer un diseño de muñeca esférica (los ejes q5 y q6 se cortan en el espacio). Además, puesto que los dos motores van a estar encerrados en la muñeca se ha provisto de canales para la circulación del aire para evitar que los motores se sobrecalienten. Igualmente, se han añadido los soportes necesarios para poder colocar un ventilador que fuerce el flujo del aire dentro de la muñeca.

Otro aspecto importante a la hora de plantear el nuevo diseño es utilizar motores NEMA 14 para las articulaciones q5 y q6. Con ellos se ha conseguido reducir las dimensiones de la muñeca, además de reducir el peso del conjunto.

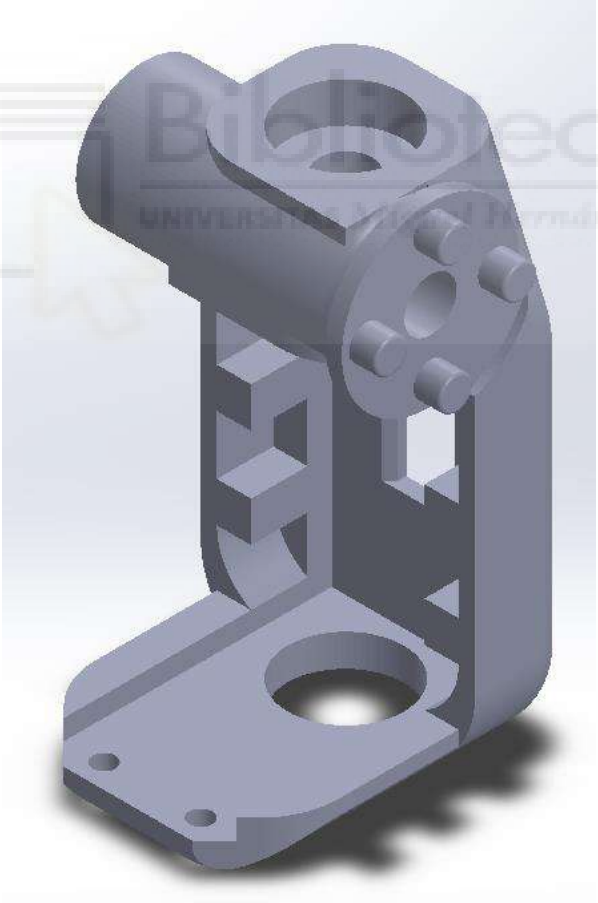


Figura 28 - Diseño original de la muñeca

El diseño de la muñeca se hizo a modo de caja (Figura 29 - Modelo CAD de la nueva muñeca) para así simplificar el proceso de diseño. Además, se planteó la posibilidad de que, al estar encerrados, los motores se calentasen, por lo que se dispusieron orificios de ventilación (Figura 30 - Detalle de los orificios de ventilación de la muñeca) por la estructura de la muñeca, y también se le dotó de una la tapa con un orificio para instalar un ventilador.

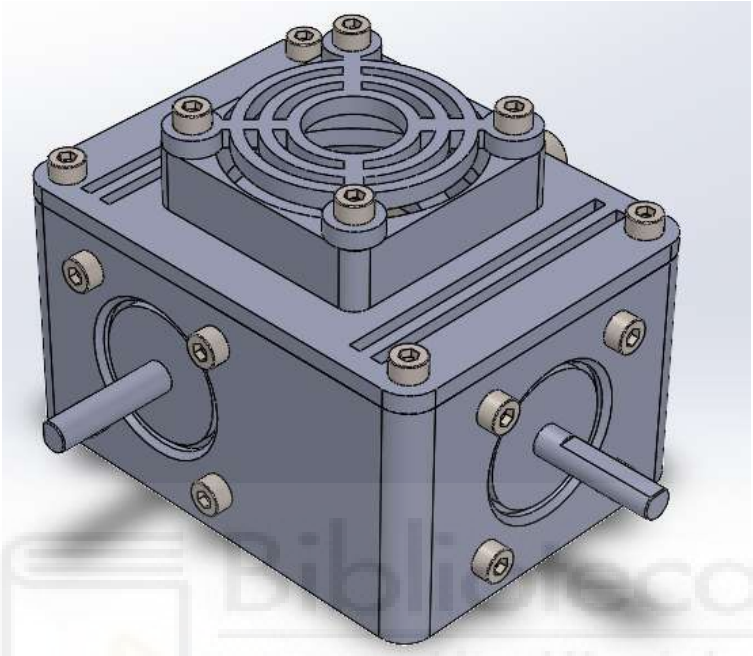


Figura 29 - Modelo CAD de la nueva muñeca

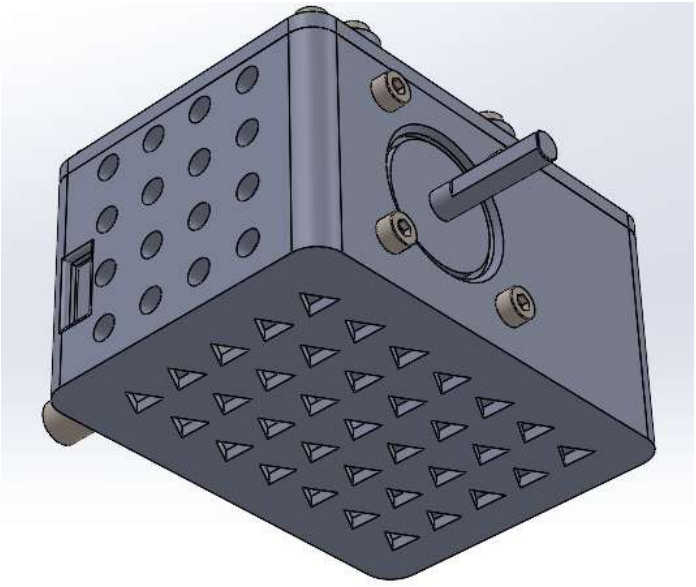


Figura 30 - Detalle de los orificios de ventilación de la muñeca

Una ventaja de este diseño con tapa es que, en el caso de querer añadir una cámara, tan solo es necesario diseñar una tapa nueva con la que poder sujetarla. De esta manera, no es necesario crear una nueva muñeca al completo para poder añadir una cámara o cualquier otro accesorio.

Finalmente, el resultado obtenido, tras imprimir las distintas piezas y realizar el montaje que se detalla en el anexo 9.4.5 - Muñeca, se puede ver en la Figura 31 - Muñeca montada en el robot.

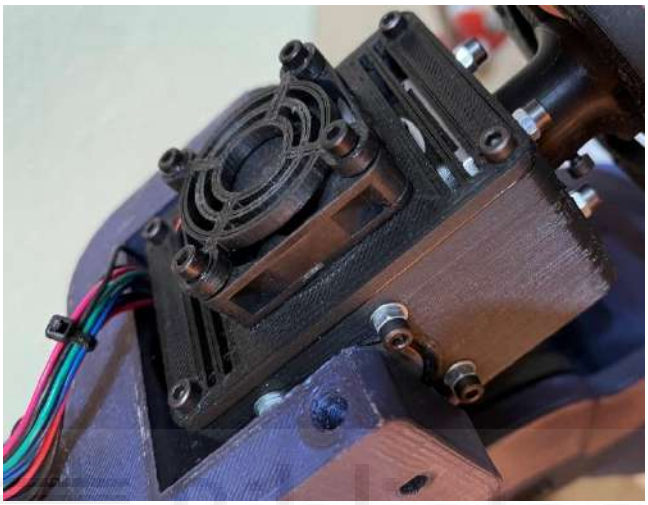


Figura 31 - Muñeca montada en el robot

5.1.3. Rediseño de la herramienta y del *end-tool*

Para ajustarse a las necesidades del trabajo, tanto en el diseño de las nuevas piezas como en el peso que han de soportar los motores, se rediseñó una de las herramientas existentes del robot, Figura 32 - Pinza original.

Figura 32 - Pinza original

Para ello, se modificó la sujeción del servo (de DYNAMIXEL a SG90 o similar). Con ello se ha conseguido reducir considerablemente el peso del conjunto de la herramienta, así como la compatibilidad con servos más genéricos, Figura 33 - Nuevo diseño de la herramienta.

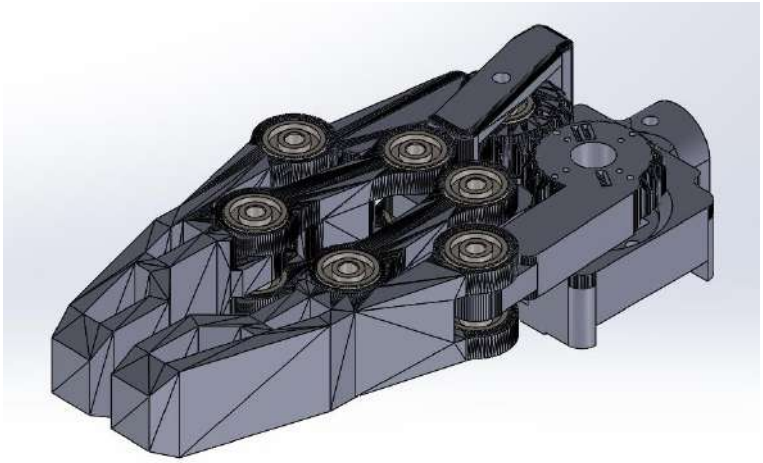


Figura 33 - Nuevo diseño de la herramienta

El resultado final del rediseño de la pinza se puede ver en la Figura 34 - Pinza montada. Tal y como se puede apreciar, el tamaño del servo es muy reducido, haciendo que, en comparación, su peso sea despreciable frente al cuerpo de la pinza.

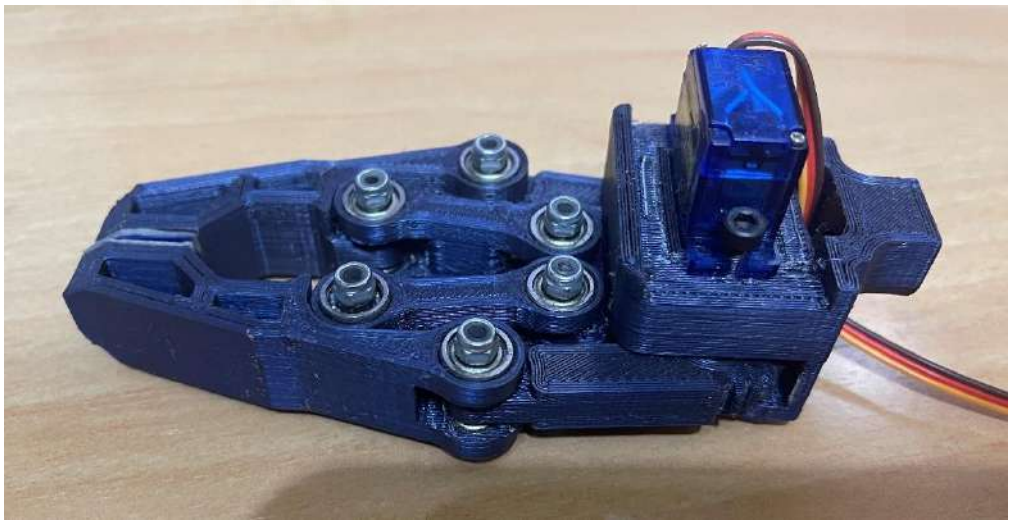


Figura 34 - Pinza montada

El diseño ha mantenido la capacidad de apertura de la pinza, siendo esta de 60 mm. Para el movimiento se ha utilizado un servo SG90, aunque también podría emplearse uno compatible, como el TS90M. Ambos ejercen un par de 1.2 kg/cm a 4.7 V, aunque admiten un voltaje máximo de 7.2 V.

Para la unión mecánica entre el eje del servo y el brazo motor de la pinza, se ha utilizado uno de los accesorios que vienen con el servo. Se ha utilizado un acople en cruz y se han recortado sus brazos para adaptarse a la geometría de la pinza.

Otro elemento modificado respecto al diseño original es el modo con el que se sujeta al eje de la articulación q6, o *end-tool*. El diseño original solo podía sujetarse a un *end-tool* muy específico, Figura 35 - End-tool original. Además, este diseño se imprimió para estudiarlo con detenimiento y se pudo observar que, debido a su morfología, se rompía con demasiada facilidad.

El nuevo diseño del *end-tool*, Figura 36 - Nuevo end-tool, es mucho más sencillo que el original, siendo un acople con agujeros dispuestos de manera circular donde enganchar una herramienta.

Figura 35 - End-tool original

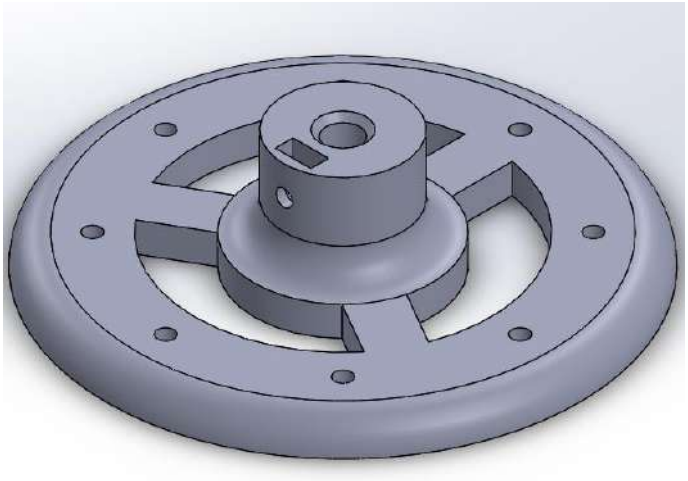


Figura 36 - Nuevo end-tool

Comparativamente, el nuevo diseño es mucho más sencillo, puesto que permite sujetar cualquier herramienta de manera directa o con un acople, como ha sido el caso de este trabajo, Figura 37 - Montaje de la herramienta en la muñeca del robot, frente al diseño original que requiere que todas las herramientas lleven un diseño más elaborado que se adapte al *end-tool* original.

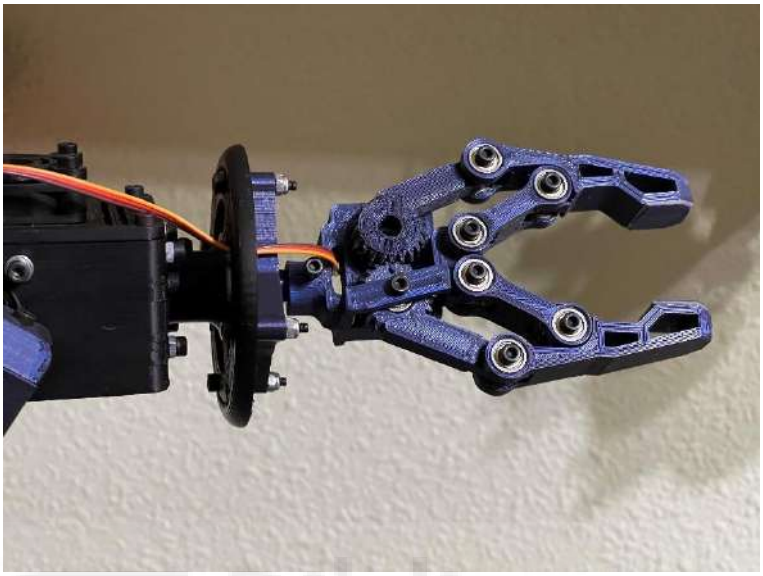


Figura 37 - Montaje de la herramienta en la muñeca del robot

Como se puede apreciar, entre la herramienta y el *end-tool* hay intercalado un acople para sujetar la herramienta al extremo del robot.

5.2. SELECCIÓN DE MOTORES PASO A PASO

Para seleccionar los motores, en primer lugar, se tuvo que caracterizar, de manera aproximada, el peso de cada eslabón, así como su momento de inercia y sus centros de gravedad. Una vez se dispuso de estos datos, para determinar las necesidades de cada motor, el robot se colocó en varias posiciones en las que los motores estén solicitados a altos pares. Por ello, las características de par de los motores se calcularon con el robot en la posición de reposo, Figura 38 - NIRYO ONE diseño definitivo – Reposo, completamente estirado en horizontal, Figura 39 – NIRYO ONE diseño definitivo – Extendido horizontal, y completamente estirado en vertical, Figura 40 – NIRYO ONE diseño definitivo – Extendido vertical.

Para calcular las necesidades de los motores se utilizó el *script* "`motor_selection.m`" incluido en la librería ARTE. Para poder utilizarlo, se introdujeron los datos de la posición inicial del robot, la velocidad y aceleración máximas expresadas en radianes por segundo y las relaciones de transmisión de las articulaciones.

La velocidad articular se fijó en [0.75 0.75 1 1.25 1.25 1.25] rad/s; la aceleración articular se fijó en [1.25 1.25 1.5 1.75 2 2.25] rad/s^2; y las relaciones de transmisión se fijaron en [97/16 133/16 126/16 1 1 1], siendo las tres últimas articulaciones las que utilizan una transmisión directa.

Por otra parte, para las articulaciones q1, q2 y q3, se eligieron motores NEMA 17, puesto que ofrecen mayor par; mientras que, para las articulaciones q4, q5 y q6, se eligieron motores NEMA 14 por su reducido peso, además de que ofrecían valores de par suficiente para las articulaciones.

5.2.1. Robot en posición de reposo

Esta es la posición en la que el robot se encuentra en reposo, Figura 38 - NIRYO ONE diseño definitivo – Reposo. Las articulaciones que más solicitación tienen son las articulaciones q2 y q3, hombro y codo respectivamente.

Figura 38 - NIRYO ONE diseño definitivo – Reposo

Una vez ejecutado el script se obtuvieron los siguientes resultados:

	q1	**q2**	**q3**	**q4**	**q5**	**q6**
Par pico [N*cm]	3.3	70.8	41.7	21.6	2.9	0.1
Par nominal [N*cm]	0.1	62.0	37.5	18.7	2.2	0.0
Velocidad máxima [rpm]	43.4	59.5	72.2	11.9	11.9	11.9

Tabla 7 - Valores de par obtenidos para la posición de reposo

Tal y como se puede ver en la Tabla 7 - Valores de par obtenidos para la posición de reposo, las articulaciones que mayor par presentan son las articulaciones q2 y q3, debido a que q3 ha de soportar el peso de los eslabones de las articulaciones comprendidas entre la q4 y la q6, y q2 ha de soportar el peso de todo el robot. Finalmente, las articulaciones q5 y q6, como era de esperar, son las que menor par necesitan.

5.2.2. Robot completamente extendido en horizontal

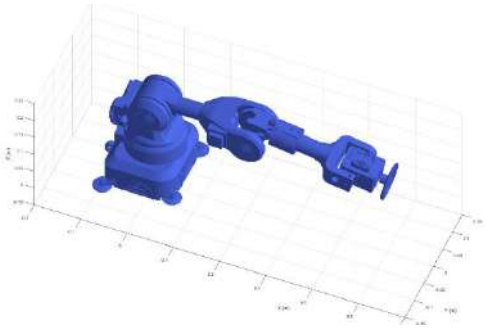


Figura 39 – NIRYO ONE diseño definitivo – Extendido horizontal

	q1	q2	q3	q4	q5	q6
Par pico [N*cm]	7.1	101.5	37.6	21.8	7.0	0.1
Par nominal [N*cm]	0.7	89.9	33.0	18.5	6.2	0.0
Velocidad máxima [rpm]	43.4	59.5	72.2	11.9	11.9	11.9

Tabla 8 – Valores de par obtenidos para la posición extendido horizontal

En este caso, las articulaciones con más solicitación a par son las articulaciones q2, q3 y q5, tal y como se puede ver en la Tabla 8 – Valores de par obtenidos para la posición extendido horizontal. Estos valores corresponden con lo esperado si se observa la Figura 39 – NIRYO ONE diseño definitivo – Extendido horizontal.

5.2.3. Robot completamente extendido en vertical

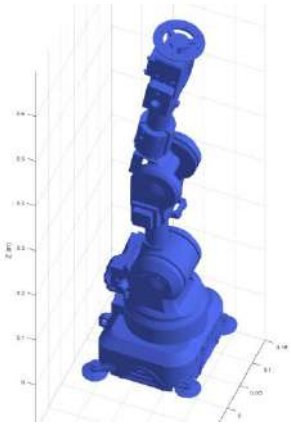


Figura 40 – NIRYO ONE diseño definitivo – Extendido vertical

	q1	q2	q3	q4	q5	q6
Par pico [N*cm]	1.3	64.6	35.4	7.9	23.1	0.1
Par nominal [N*cm]	0.2	52.9	30.9	3.0	22.3	0.1
Velocidad máxima [rpm]	43.4	59.5	72.2	11.9	11.9	11.9

Tabla 9 – Valores de par obtenidos para la posición extendida vertical

En este caso, de nuevo se puede ver que la articulación q2 y q3 son las que están más solicitadas a par, además de la articulación q5 que, al no poder alcanzar el robot una posición puramente vertical, tiene que ejercer un par para soportar el peso de los motores.

Finalmente, una vez analizados los casos anteriores, se eligieron motores con las siguientes características:

	q1	q2	q3	q4	q5	q6
Par pico [N*cm]	59	70	45	23	14	14

Tabla 10 - Valores de par para seleccionar los motores que se montarán

Como se puede observar, en todas las articulaciones se han elegido motores con valor de par pico superior al máximo. No obstante, para la articulación q2 se ha partido de un valor pico que coincidiese con las especificaciones de algún fabricante, pero, además, para poder alcanzar el valor de par necesario, se ha elegido un motor que contase con una reductora de velocidad con la que aumentar el par ejercido por el motor. Esto se trata con más detalle en el apartado 6.1.3.2 - Uso de un motor con reductora planetaria.

Por otra parte, para las articulaciones q5 y q6 se eligieron el mismo motor por simplicidad a la hora de diseñar la muñeca del robot.

5.3. DISEÑO DE LA ELECTRÓNICA Y DE LA PCB

En el mercado existen diversos modelos de placas de circuito impreso (PCB) especialmente diseñadas para controlar distintos tipos de motores, como por ejemplo la RAMPS [14], Figura 41 - RAMPS 1.6 (ampliamente utilizada en el mundo de la impresión 3D), o la *Arduino Motor Shield*.

En una etapa temprana del diseño, se planteó utilizar alguna de las placas ya existentes en el mercado, pero, puesto que su uso está pensado para impresoras 3D, es decir, robots cartesianos, solo pueden manejar un motor por cada eje, además de uno o dos motores adicionales para el o los extrusores del plástico, por lo que seguiría quedando uno o dos grados de libertad del robot sin poder controlar con la misma placa.

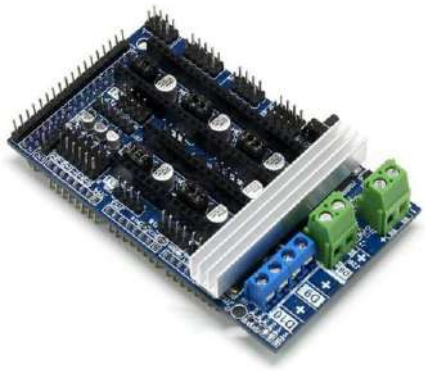


Figura 41 - RAMPS 1.6

De este modo, el diseño de la PCB que se ha realizado en este trabajo, si se compara con una RAMPS o cualquier otra placa similar, cuenta con espacio para conectar 6 *drivers* para controlar los motores paso a paso, frente a los 4 o 5 disponibles en una RAMPS, y prescinde de los elementos necesarios para controlar y alimentar los elementos calefactores de la impresora (fusor y cama caliente).

No obstante, tal y como se verá más adelante, el diseño de la PCB de este trabajo cuenta con una serie de características que hacen que haya una marcada diferencia entre la placa de circuito impreso diseñada específicamente para el robot y las placas de circuito impreso genéricas como la RAMPS.

Para ello, se establecen una serie de criterios de diseño básicos: ha de ser una placa que se pueda conectar sobre una Arduino MEGA a modo de *shield* (igual que las dos PCB anteriormente mencionadas), ha de ser un diseño compacto, y solo puede montar componentes en una de las caras de la PCB, aunque las pistas sí que pueden discurrir por ambas caras de la placa.

La PCB integra todos los componentes necesarios en una superficie de 101.6 x 53.34 mm para el control de los motores paso a paso de cada una de las articulaciones, así como una pequeña fuente de alimentación basada en un regulador de voltaje LM7806 para alimentar el servomotor de la herramienta. Para el diseño de la PCB se ha utilizado tecnología SMT (*Surface-Mount Technology*) y THT (*Through-Hole Technology*). El esquema eléctrico, así como el enrutado de las pistas por ambas caras, se puede consultar en el anexo 9.3 - ESQUEMA ELECTRÓNICO Y DIAGRAMA DE LA PCB.

Finalmente, el diseño de la PCB se mandó a fabricar con la empresa JLCPCB, y también se solicitó que se montaran todos los componentes SMD (*Surface-Mount Device*). El resultado se puede ver en la Figura 42 - Caras frontal (izquierda) y trasera (derecha) de la PCB.

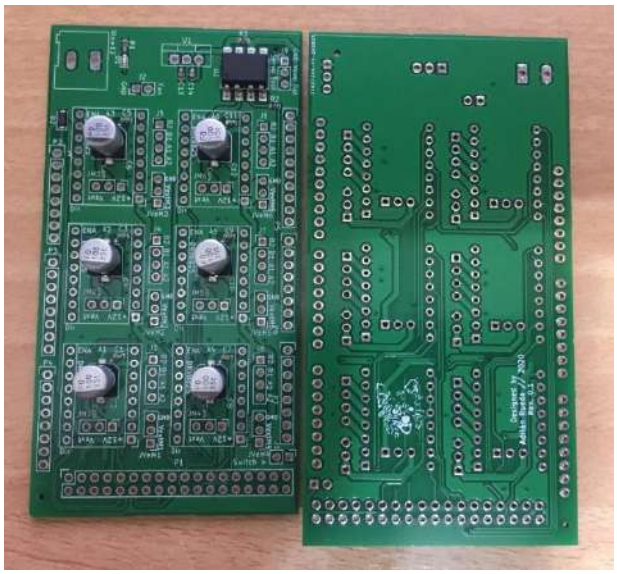


Figura 42 - Caras frontal (izquierda) y trasera (derecha) de la PCB

No obstante, en la Figura 42 - Caras frontal (izquierda) y trasera (derecha) de la PCB, se puede apreciar que en la PCB, aunque ya contiene algunos de los componentes, aún quedan *pads* de soldadura para otros componentes THT por soldar. Los componentes que restaban por soldar eran tiras de pines macho y hembra. El resultado tras añadir el resto de componentes se puede ver en la Figura 43 - PCB con todos los elementos soldados.

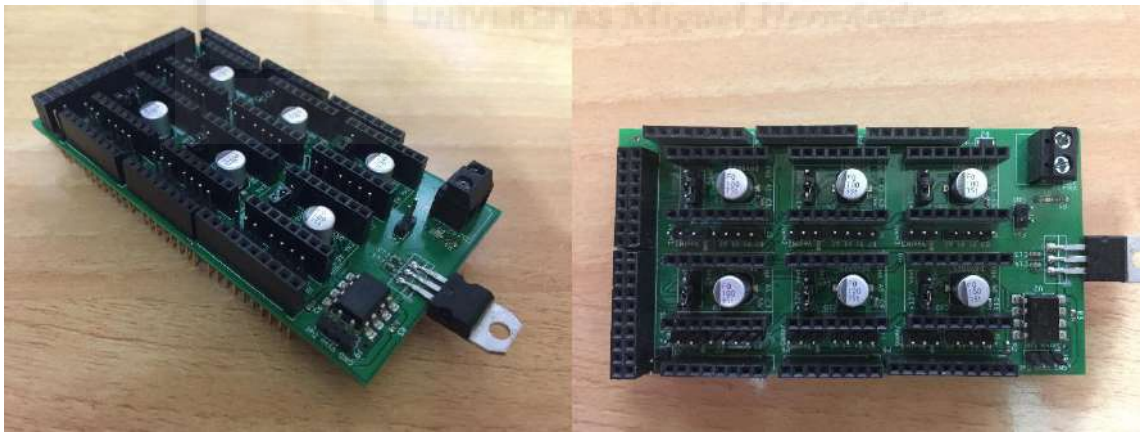


Figura 43 - PCB con todos los elementos soldados

Si se observa con detalle, en la Figura 44 - Detalle del zócalo para conectar el driver DRV8825, se pueden ver una serie de conectores etiquetados como: **VextM6**, **+12V Vext** y **B2 B1 A1 A2**. El conector **VextM6** sirve para alimentar cada motor de manera independiente, es decir, si no se puede alimentar a 12 V generales con ese conector se puede alimentar el *driver* y el motor de manera externa, una diferencia realmente importante entre este diseño y el de la RAMPS, o similares. Por ello, también se añadió el selector "**+12V Vext**", con el que seleccionar si se alimenta con los 12 V generales o con una alimentación externa. Para seleccionar uno u otro se utiliza un

jumper que permite abrir o cerrar un circuito u otro. En este caso, todos los motores se alimentan de los 12 V generales.

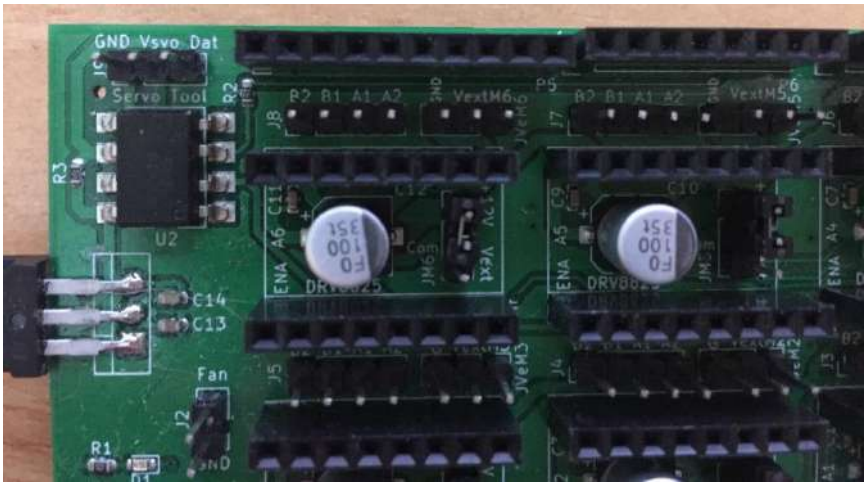


Figura 44 - Detalle del zócalo para conectar el driver DRV8825

El conector etiquetado como "**B2 B1 A1 A2**" se utiliza para conectar el motor, siendo **B1** y **B2** una bobina, y **A1** y **A2** la otra bobina del motor.

Adicionalmente, se pueden ver otros conectores en la parte superior izquierda de la PCB y otro en la parte central. El primero es para la conexión del servo de la herramienta, mientras que el segundo se utiliza para conectar los distintos ventiladores que utiliza el robot.

Finalmente, tras configurar la PCB para el tipo de alimentación que se va a utilizar, colocar los *drivers* y conectar con la Arduino MEGA, el resultado se puede ver en la Figura 45 - Montaje final de la PCB.

Una diferencia importante entre la placa diseñada y una RAMPS, o similares, es que en esta se ha prescindido de los pines necesarios para configurar el *microstepping* mediante *hardware*. Mientras que en la RAMPS, si se observa la Figura 41 - RAMPS 1.6, hay una serie de pines cortocircuitados con *jumpers* que se utilizan para configurar dicho *microstepping*, en el diseño creado esta selección se hace mediante *software* con la activación o desactivación de una salida digital, y se limita a *full-step* o el máximo *microstepping* permitido por los *drivers*, en el caso de los DRV8825 de 1/32 de paso.

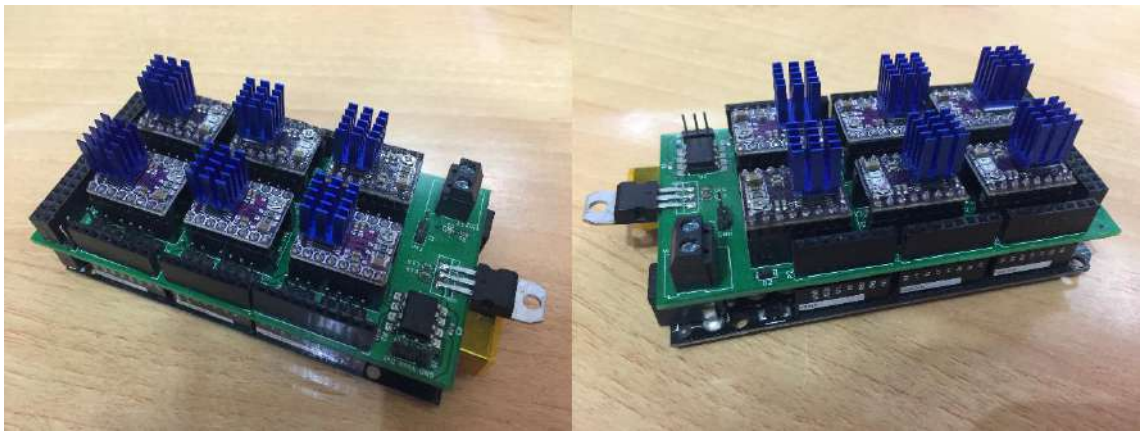


Figura 45 - Montaje final de la PCB

Además, este robot está pensado para ser modificado en el futuro, por lo que todas las conexiones a los pines de Arduino se han dejado accesibles. Esto permite, por una parte, añadir elementos al robot, como pueden ser sensores u otros elementos que aporten alguna utilidad al robot; pero, por otra parte, también son útiles en el caso de querer realizar mediciones para comprobar el funcionamiento del robot. En la Tabla 11 - Tabla de conexiones de la PCB con Arduino se pueden ver todas las conexiones actuales a los pines de la Arduino MEGA.

PIN	CON.	PIN	CON.	PIN	CON.	PIN	CON.	PIN	CON.
A0	NC	0 (RX0)	NC	16 (TX2)	NC	32 (I/O)	DIR_A6	48 (I/O)	EN_A2
A1	NC	1 (TX0)	NC	17 (RX2)	NC	33 (I/O)	DIR_A5	49 (I/O)	EN_A1
A2	NC	2 (PWM)	PWMservo	18 (TX1)	NC	34 (I/O)	STP_A6	50 (I/O)	NC
A3	NC	3 (PWM)	NC	19 (RX1)	NC	35 (I/O)	STP_A5	51 (I/O)	NC
A4	NC	4 (PWM)	NC	20 (SDA)	NC	36 (I/O)	EN_A6	52 (I/O)	NC
A5	NC	5 (PWM)	NC	21 (SCL)	NC	37 (I/O)	EN_A5	53 (I/O)	NC
A6	NC	6 (PWM)	NC	22 (I/O)	NC	38 (I/O)	DIR_A4		
A7	NC	7 (PWM)	NC	23 (I/O)	NC	39 (I/O)	DIR_A3		
A8	NC	8 (PWM)	NC	24 (I/O)	NC	40 (I/O)	STP_A4		
A9	NC	9 (PWM)	NC	25 (I/O)	NC	41 (I/O)	STP_A3		
A10	NC	10 (PWM)	NC	26 (I/O)	NC	42 (I/O)	EN_A4		
A11	NC	11 (PWM)	NC	27 (I/O)	NC	43 (I/O)	EN_A3		
A12	NC	12 (PWM)	NC	28 (I/O)	NC	44 (I/O)	DIR_A2		
A13	NC	13 (PWM)	NC	29 (I/O)	NC	45 (I/O)	DIR_A1		
A14	NC	14 (TX3)	NC	30 (I/O)	MSTP	46 (I/O)	STP_A2		
A15	NC	15 (RX3)	NC	31 (I/O)	Cswitch	47 (I/O)	STP_A1		

Tabla 11 - Tabla de conexiones de la PCB con Arduino

Donde:

- **NC** es un pin no conectado o no utilizado.
- **MSTP** es el selector de *microstepping* (*full-step* o 1/32 *step*).
- **Cswitch** es el conector del botón.
- **DIR_AX** son los selectores de dirección del motor de las articulaciones.
- **STP_AX** son los pines donde se aplican los pulsos para controlar los motores.
- **EN_AX** son los pines de habilitación de los motores.

5.4. IMPRESIÓN

El proceso de impresión de las distintas partes del robot, Figura 46 - Proceso de impresión del hombro del robot, ha sido una de las tareas más críticas de este trabajo, pues una correcta impresión significa que, aún debido a los esfuerzos a los que se someten las distintas articulaciones y partes del robot, podrá aguantar sin romperse ninguna pieza debido a la separación entre las capas de impresión.

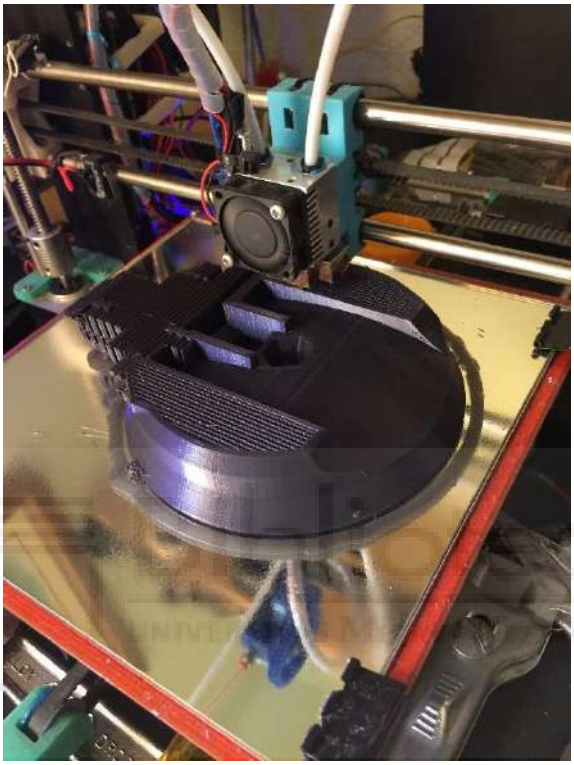


Figura 46 - Proceso de impresión del hombro del robot

Además, el poder fabricar el robot con una impresora 3D ha permitido poder realizar una pequeña parte de prototipado en algunas piezas, como la articulación q4, 5.1.1 - Rediseño de la articulación q4 (codo y antebrazo), o probar distintas herramientas, así como hacer distintas pruebas de impresión a la hora de modificarla, 5.1.3 - Rediseño de la herramienta y del *end-tool*.

Esto ha sido muy crítico, pues algunos archivos de las piezas de la herramienta no se podían medir correctamente desde el ordenador, por lo que se han tenido que tomar medidas de manera manual, lo que introduce cierto error que, en algunos casos, ha provocado desviaciones importantes en el diseño final y ha hecho que fuera necesario modificar el archivo en numerosas ocasiones y reimprimirlo.

Por otra parte, debido a las limitaciones de la impresora utilizada, la pieza de la base del robot, Figura 47 - Proceso de impresión de media base del robot, era inviable imprimirla de una sola pieza, por lo que se optó por dividirla en dos partes y, entonces, proceder a imprimirla.

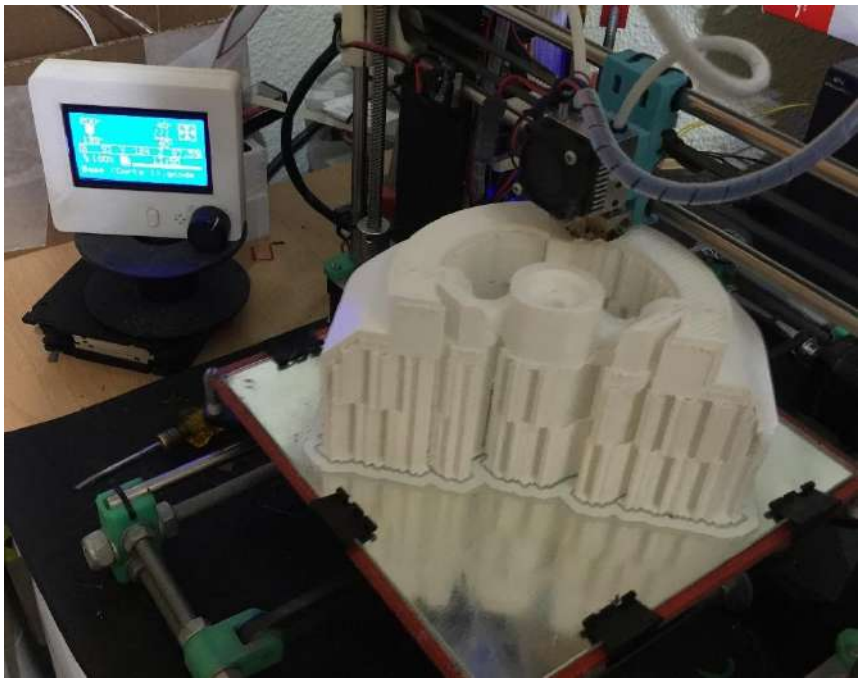


Figura 47 - Proceso de impresión de media base del robot

5.5. PROGRAMACIÓN

La programación se ha dividido en dos partes: por un lado, la programación de la controladora en la Arduino y, por otro lado, la programación en MATLAB para comunicarse con la controladora desde la librería ARTE, además de las distintas funciones para controlar el robot desde ARTE.

5.5.1. Inclusión del robot en la librería arte

Una de las primeras tareas de este trabajo ha sido añadir el robot a la librería ARTE. No es tarea de este trabajo explicar cómo se ha realizado esta labor, aunque se puede consultar cómo se ha realizado en [3].

5.5.2. Programación de la controladora

La programación de la controladora se ha realizado con el lenguaje C++ de Arduino. Para el desarrollo del código de la controladora se plantearon varias opciones, pero tras varias pruebas, la implementación de una máquina de estados fue la más acertada por su simplicidad y funcionamiento adecuado frente a las necesidades de este trabajo.

El funcionamiento de la máquina de estados de la controladora se basa en 5 estados descritos a continuación:

- ***idle***: Estado de reposo a la espera de cualquier orden.
- ***home***: Estado en el que se coloca el robot en el origen de cada articulación de manera automática.

- ***calibrate***: Estado en el que se coloca el robot en el origen de cada articulación de manera manual.
- ***move***: Movimiento del robot.
- ***tool*****:** Activación o desactivación de la herramienta

El diagrama de la máquina de estados puede observarse en la Figura 48 - Esquema de la máquina de estados de la controladora del robot.

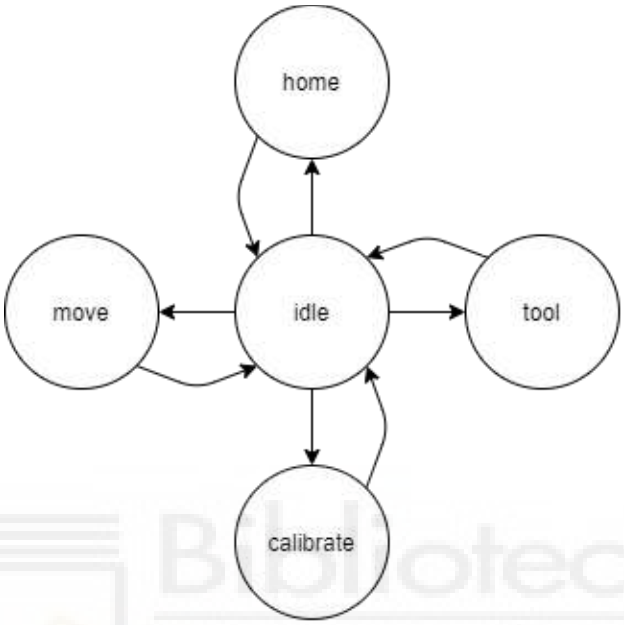


Figura 48 - Esquema de la máquina de estados de la controladora del robot

Tras alimentarse la controladora y comenzar una comunicación serie nueva, la rutina de ejecución pasa del **estado *idle*** al **estado *calibrate*** de manera automática mediante un *flag*. En este momento, el usuario ha de colocar cada articulación en el origen, este procedimiento se explica en detalle en el apartado 5.6 - Puesta en marcha del robot.

Una vez realizado este primer paso, el robot permanecerá de manera constante en el **estado *idle*** a la espera de recibir un nuevo comando. Del **estado *idle*** puede pasar al **estado *move*** para realizar un movimiento; puede pasar del **estado *idle*** al **estado *tool*** para actuar sobre la herramienta; puede pasar del **estado *idle*** al **estado home** para volver al origen de cada articulación. El comportamiento del **estado *home*** se puede realizar en el **estado *move***, pero se ha considerado oportuno añadir un estado específico para este comportamiento. La transición entre los distintos estados se puede observar en el apartado 5.6.1 - Control del robot con la librería ARTE

Aunque el código de la controladora del robot se puede consultar en el anexo 9.1 - CÓDIGO DE LA CONTROLADORA, a continuación, se comentan las funciones principales creadas para el funcionamiento de la controladora.

`void state_machine()` - Esta función es la encargada de realizar las transiciones entre un estado u otro dependiendo del mensaje recibido. Se comprueba periódicamente

dentro de la función **`void loop()`** para comprobar si hay que realizar una transición entre estados.

`void read_serial()` – Esta función es la encargada de comprobar si hay datos disponibles para leer en el buffer de la comunicación serie. Del mismo modo que **`void state_machine()`**, esta función se comprueba en cada iteración de la función **`void loop()`**.

`void calibrate_joints()` – Esta función es la que se ejecuta cuando se ha de calibrar el robot. Se ha de tener cuidado pues el ejecutar esta función hace que los *drivers* de los motores se deshabiliten, haciendo que el robot caiga por su propio peso.

`void move_tool(char mov_tl)` – Esta función es la que se ejecuta cuando se ha de abrir o cerrar la herramienta. Si el valor **`mov_tl`** toma valor 1, entonces la herramienta se cierra. Por el contrario, si el valor que toma es 0, la herramienta se abre. En el caso de que no fuera ninguno de los dos, el valor tomado se ignora.

5.5.2.1. Comunicaciones serie

Arduino cuenta con distintos elementos para comunicarse con otros dispositivos, pero el más utilizado para comunicarse con un ordenador es mediante la comunicación serie, ya sea directamente por la UART (*Universal Asynchronous Receiver-Transmitter*), o de una manera mucho más cómoda con el puerto USB que lleva integrado.

Por tanto, las comunicaciones entre el robot y el ordenador con MATLAB y ARTE se realizan mediante comunicaciones serie por USB. Esto permite que el diseño de los mensajes sea flexible, ya que el protocolo serie no cuenta con un estándar para delimitar los mensajes, por lo que brinda una gran libertad a la hora de crear mensajes propios que facilitarán enormemente el *parsing* o análisis de la trama de datos transmitidos.

Por ello, para que el robot y el ordenador se puedan comunicar, se han creado los siguientes mensajes:

`MOVQ_q1_q2_q3_q4_q5_q6` – Este mensaje le indica al robot a la posición articular a la que ha de moverse. Donde q_i son las distintas posiciones articulares.

`HOME` – Le indica al robot que ha de volver a la posición de inicio.

`TOOL_X` – Este mensaje le indica al robot si ha de abrir o cerrar la pinza en función del valor de X. Si X toma el valor de 1, cierra la pinza; por el contrario, si X toma el valor 0, abre la pinza.

Los efectos de estos mensajes se pueden ver en detalle en el apartado 5.6.1 - Control del robot con la librería ARTE, así como su envío desde ARTE en el anexo 9.2.4 - *Scripts* y funciones para el control del robot.

5.5.2.2. Control de los motores

Controlar un motor paso a paso no es una tarea compleja de implementar con un microcontrolador, puesto que solo es necesario que exista un tren de pulsos que active la entrada “step” del *driver* que controla al motor. Sin embargo, realizarlo de una

manera eficiente y que permita controlar más de un motor a la vez es algo bastante complejo, por lo que se optó por utilizar las librerías MultiStepper y AccelStepper [15].

El uso de la librería se puede consultar en [15], así como ver la implementación de esta en el anexo 9.1 - CÓDIGO DE LA CONTROLADORA.

5.5.2.2.1. Microstepping

Los motores paso a paso, o *steppers*, por lo general, para dar un giro de 360º con un incremento de paso de 1.8º, requieren de un total de 200 pasos/vuelta. Esto significa que el motor, junto al *driver*, funcionan a *full-step*. No obstante, a velocidades lentas funcionando a *full-step*, los motores paso a paso, por lo general, vibran, lo que es un comportamiento nada deseado en un sistema en el que se busca alcanzar un mínimo de precisión en el movimiento.

Para solucionar este problema de vibraciones, existe el *microstepping*, que es utilizar una señal PWM (*Pulse-Width Modulated*) del voltaje para controlar la cantidad de corriente que les llega a los motores. [16]

El *driver* envía dos señales senoidales de voltaje desfasadas 90 grados a los bobinados del motor. Mientras que la corriente aumenta en un devanado, desciende en el otro. Esta transferencia gradual de corriente produce un movimiento más suave y un par más constante que en el control en *full-step* o *half-step.* [16]

El efecto del *microstepping* se puede ver en la Figura 49 - Efecto del microstepping , donde en la parte superior se ve la forma de la señal que alimenta cada una de las bobinas del motor, donde se comienza con una señal puramente cuadrada (*full-step*), para continuar con una señal cuadrada más compleja con distintos niveles (*half-step*), y finalizar con una señal cuadrada que se puede aproximar a una onda senoidal (*microsteps*). Esto es que, con el *microstepping,* se intenta alcanzar que la corriente que alimenta cada una de las fases del motor sea lo más aproximado a una señal senoidal.

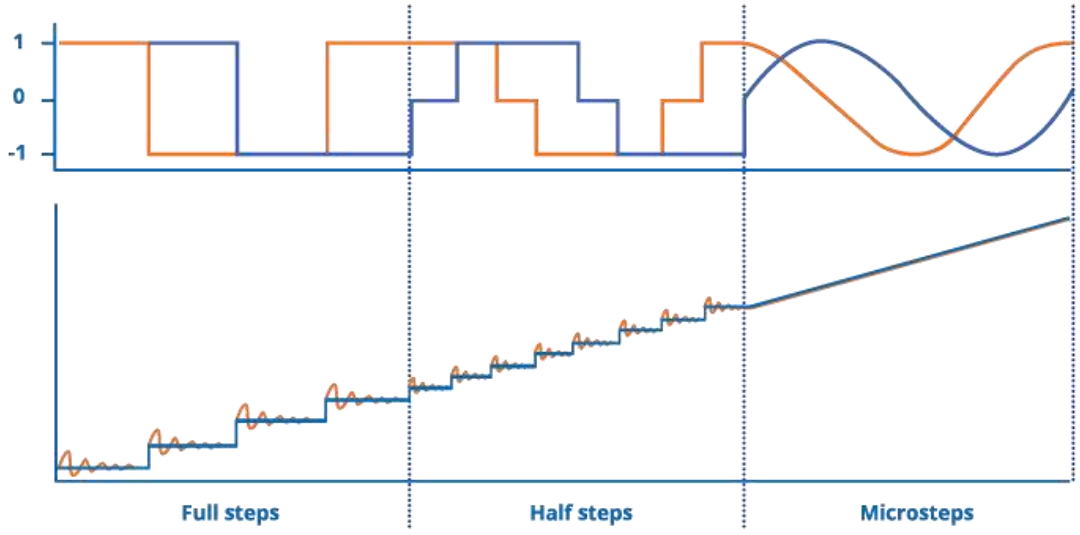


Figura 49 - Efecto del microstepping [17]

En la parte inferior de la Figura 49 - Efecto del microstepping se puede observar el efecto de cómo se comporta la corriente que alimenta cada bobina, existiendo en *full-step* una gran sobreoscilación (naranja) respecto a la señal esperada (azul), que se va atenuando conforme se aumenta al *microstepping*, donde no se aprecia ningún tipo de sobreoscilación.

Por tanto, se puede extraer que el *microstepping* aumenta la exactitud de la posición del motor, pero que también disminuye el ruido y las vibraciones, a la vez que aumenta la suavidad del movimiento y el par ejercido por el eje del motor [17].

5.6. Puesta en marcha del robot

El proceso para poner en marcha el robot se resumen en 4 pasos que se describen, uno a uno, a continuación.

1- Inicialización de la librería ARTE

El primer paso es inicializar la librería ARTE. Para ello, se ha de configurar el *path* con la librería y, una vez hecho, se inicializa con el comando "**`init_lib`**", Figura 50 - Mensajes tras inicializar la librería. El *path* depende de dónde se haya guardado la librería en el ordenador.

```
>> init_lib
%     ARTE (A Robotics Toolbox for Education)
%     Copyright (C) 2012  Arturo Gil Aparicio, arturo.gil@umh.es
%     http://arvc.umh.es/arte
%
%     This program is free software: you can redistribute it and/or modify
%     it under the terms of the GNU Lesser General Public License as published by
%     the Free Software Foundation, either version 3 of the License, or
%     any later version.
%
%     This program is distributed in the hope that it will be useful,
%     but WITHOUT ANY WARRANTY; without even the implied warranty of
%     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%     GNU  Lesser General Public License for more details.
%
%     You should have received a copy of the GNU Lesser General Public License
%     along with this program.  If not, see  <http://www.gnu.org/licenses/>.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     To begin with, try loading a robot:
%     >> robot = load_robot('ABB','IRB140');
%
%     Next, draw it on its zero position:
%     >> drawrobot3d(robot,[0 0 0 0 0 0])
%
%     Next, try a different pose:
%     >> drawrobot3d(robot,[0 pi/2 -pi/2 0 0 0])
%
%     Finally, use the teach pendant to move the robot:
%     >> teach
%
%   Please, type:
%   >> configuration.delta_time=0.1
%   if you find that the simulation is running too slow.
%
%   Please, type:
%   >> configuration.time_delay = 0.01;
%   if the simulation plot does not show the robot correctly.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
```

Figura 50 - Mensajes tras inicializar la librería

2- Conectar el robot al ordenador

La conexión del robot al ordenador se realiza mediante conexión USB y se puede tener conectado antes de ejecutar MATLAB e inicializar la librería ARTE, o se puede conectar tras inicializar la librería.

Si se conecta el robot una vez abierto MATLAB, el programa detectará automáticamente que se le ha conectado una Arduino, una Arduino MEGA 2560 en este caso, Figura 51 - Detección de Arduino por MATLAB.

```
Arduino Mega 2560 detected.
To use this device with MATLAB, install MATLAB Support Package for Arduino Hardware.
To use this device with Simulink, install Simulink Support Package for Arduino Hardware.
```

Figura 51 - Detección de Arduino por MATLAB

3- Cargar el robot en el *Workspace* de MATLAB

Para poder controlar el robot desde MATLAB es necesario cargarlo en el programa. Para ello, con el comando **`load_robot`** se cargará en el *workspace* de MATLAB.

```
>> robot = load_robot('NIRYO')

ans =

    'E:\UMH\Master\TFM\ARTE\arte\robots\NIRYO'

Reading link 0
 E:\UMH\Master\TFM\ARTE\arte\robots\NIRYO/link0.stl
EndOfFile found...
Reading link 1
 E:\UMH\Master\TFM\ARTE\arte\robots\NIRYO/link1.stl
EndOfFile found...
Reading link 2
 E:\UMH\Master\TFM\ARTE\arte\robots\NIRYO/link2.stl
EndOfFile found...
Reading link 3
 E:\UMH\Master\TFM\ARTE\arte\robots\NIRYO/link3.stl
EndOfFile found...
Reading link 4
 E:\UMH\Master\TFM\ARTE\arte\robots\NIRYO/link4.stl
EndOfFile found...
Reading link 5
 E:\UMH\Master\TFM\ARTE\arte\robots\NIRYO/link5.stl
EndOfFile found...
Reading link 6
 E:\UMH\Master\TFM\ARTE\arte\robots\NIRYO/link6.stl
EndOfFile found...
robot =

  struct with fields:

                  name: 'NIRYO'
                    DH: [1×1 struct]
                     J: []
    inversekinematic_fn: 'inversekinematic_niryo(robot, T)'
     directkinematic_fn: 'directkinematic(robot, q)'
                  port: 'COM3'
              baudrate: 9600
                serial: [1×1 serial]
                   DOF: 6
                  kind: 'RRRRRR'
              maxangle: [6×2 double]
                velmax: [6×1 double]
              accelmax: [6×1 double]
         linear_velmax: 2.5000
                    T0: [4×4 double]
                 debug: 0
                     q: [6×1 double]
                    qd: [6×1 double]
                   qdd: [6×1 double]
                  time: []
              q_vector: []
             qd_vector: []
            qdd_vector: []
           last_target: [4×4 double]
```

```
last_zone_data: 'fine'
         tool0: [1 0 0 0 1 0 0 0 0.1000 0 0 0 1 0 0 0 0 0 0]
         wobj0: []
tool_activated: 0
          path: 'E:\UMH\Master\TFM\ARTE\arte\robots\NIRYO'
     graphical: [1×1 struct]
          axis: [-0.5000 0.7500 -0.7500 0.7500 0 1.1000]
    parameters: [1×1 struct]
  has_dynamics: 1
      dynamics: [1×1 struct]
        motors: [1×1 struct]
```

Como se puede ver en la respuesta de MATLAB, al ejecutar el comando **`load_robot`**, el robot cuenta con 3 parámetros importantes:

- ***port*** – Es el puerto serie para las comunicaciones. Este valor se puede cambiar desde el archivo "**`parameters.m`**" si fuera necesario.
- ***baudrate*** – Es la velocidad de transferencia de datos. Este valor no se puede modificar, pues va fijado en la controladora del robot.
- ***serial*** – Es el puerto por el que se comunican el robot y el ordenador.

4- Inicialización del robot

Una vez conectado el robot, se ha de inicializar y colocar cada articulación en su posición inicial. Para ello, se debe ejecutar el script "**`robot_start.m`**". Una vez ejecutado, MATLAB indicará qué se ha de hacer, tal y como se puede ver en el siguiente extracto de las respuestas:

```
>> robot_start
robot_online
Starting calibration process
Put each joint on start position
Press robot button once each joint is on the starting position
Joint 1 (base) enabled and calibrated
Joint 2 (shoulder) enabled and calibrated
Joint 3 (elbow) enabled and calibrated
Joint 4 (forearm) enabled and calibrated
Joint 5 (wirst 1) enabled and calibrated
Joint 6 (wirst 2) enabled and calibrated
Robot ready!
```

Para que cada una de las articulaciones quede fija en la posición de inicio (se habilita el motor de la articulación), se ha de presionar el botón que se encuentra en la base del robot, Figura 52 - Botón de habilitación de las articulaciones.

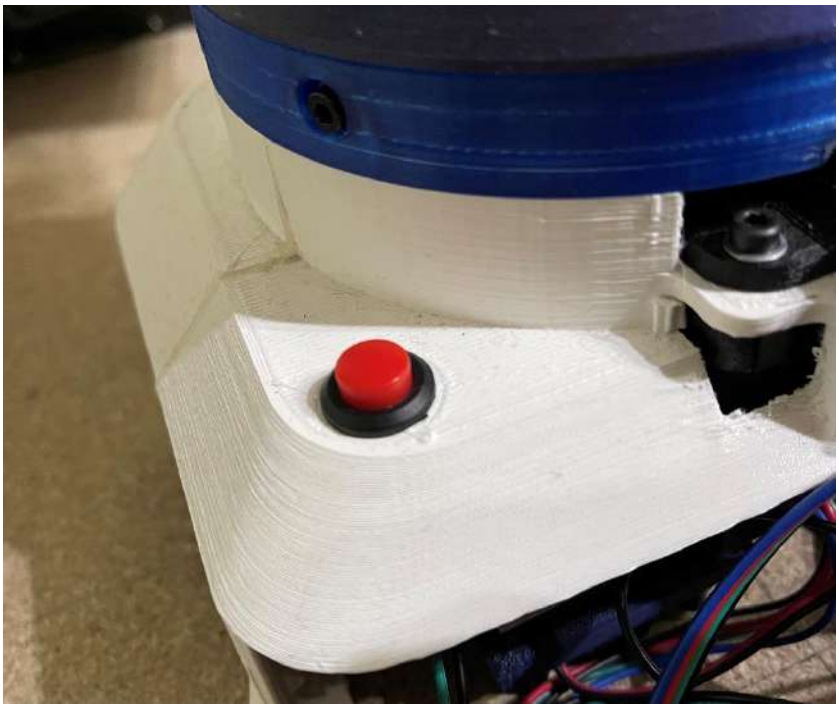


Figura 52 - Botón de habilitación de las articulaciones

Si MATLAB devuelve el mensaje "`Robot ready!`", el robot está listo y a la espera de recibir órdenes. Los distintos comandos, así como las funciones necesarias para controlar el robot se encuentran en el apartado 5.6.1 - Control del robot con la librería ARTE.

5.6.1. Control del robot con la librería ARTE

Para controlar el robot se han de tener en cuenta, antes de nada, una serie de precauciones, como que no haya nada que interfiera con la trayectoria del robot. A continuación, se detallan los distintos comandos o funciones ejecutables y sus diferentes consecuencias:

- **`robot_start`** – Este comando inicializa las comunicaciones con la controladora. No obstante, independientemente de la posición en la que el robot se encuentre, todos los motores de las articulaciones se deshabilitan y, por tanto, el robot cae por su propio peso. Se han de tomar las precauciones oportunas, como sujetar el robot antes de ejecutar este comando. Adicionalmente, si ya existe una conexión establecida, no se puede ejecutar el comando y MATLAB devuelve el siguiente mensaje:

```
Error using serial/fopen (line 72)
Open failed: OBJ has already been opened.
```

- **`fclose(robot.serial)`** – Este comando finaliza las comunicaciones con la controladora. Puede darse el caso de que la controladora se reinicie y deshabilite la alimentación de los motores. En dicho caso, el robot caería por

su propio peso. Se han de tomar las precauciones necesarias, como sujetar el robot tras ejecutar este comando.

- **`move_robot(robot, param)`** – Con este comando se indica la posición a la cual el robot debe moverse. En todo momento se debe pasar el parámetro "**`robot`**" a la función "**`move_robot`**", así como un vector fila de posiciones articulares, la palabra "**`HOME`**" / "**`home`**", o la palabra "**`TEACH_X`**" / "**`teach_X`**". En el caso de pasarle un vector fila de posiciones articulares, el robot se mueve hasta alcanzarlas. En el caso de indicarle "**`HOME`**", el robot realiza un *homing* o vuelta al inicio, donde vuelve a la posición que se le hubiera calibrado inicialmente (**`robot_start`**), siempre y cuando no haya perdido pasos alguno de sus motores. En caso de que esto último ocurriera, se recomienda reiniciar el robot y recalibrarlo. Si se le indica "**`TEACH_X`**", se actúa sobre la herramienta del robot. El valor de X puede ser 1 para cerrar la pinza, o 0 para abrirla.

Si se quiere controlar el robot con el ***teach pendant*** de la librería ARTE, se han de seguir los siguientes pasos:

1- Lanzar el teach pendant

Para lanzar el *teach pendant*, en primer lugar, se ha de tener cargada la librería con **`init_lib`**. A continuación, se llama con el comando **`teach`**, y aparece el *teach pendant*, Figura 53 - Teach pendant de ARTE.

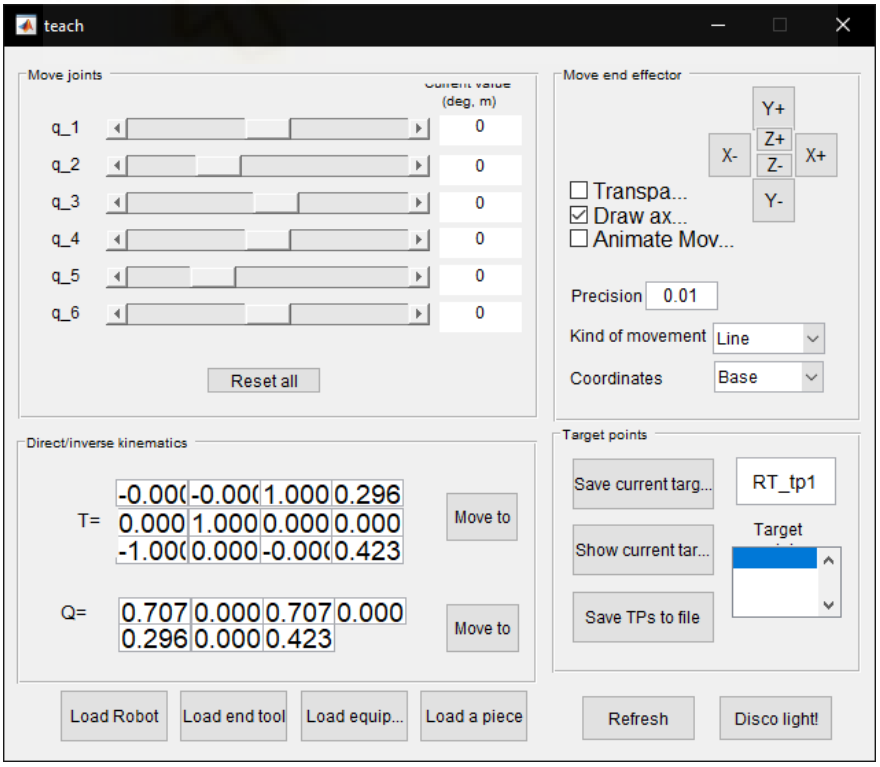


Figura 53 - Teach pendant de ARTE

2- Colocar el robot en la posición deseada

Una vez abierto el *teach pendant*, con los controles deslizantes, se coloca el robot en una posición deseada, Figura 54 - Robot posicionado desde el *teach pendant*.

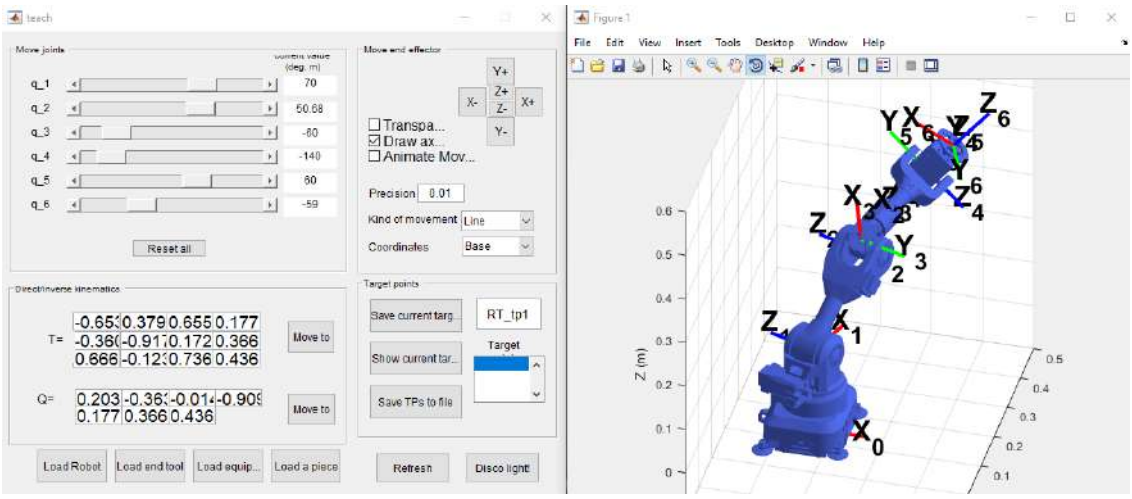


Figura 54 - Robot posicionado desde el teach pendant

Esta posición queda almacenada en uno de los parámetros del robot (ver anexo 9.2.1 - parameters.m) que será llamado en el siguiente paso para indicarle al robot real dónde tiene que colocarse.

3- Mover el robot real

Para mover el robot, se ha de utilizar la función **`move_robot(robot, param)`**, pero para utilizar la posición seleccionada con el *teach pendant* se ha de ejecutar de la siguiente manera:

```
move_robot(robot, robot.q)
```

El parámetro **`robot.q`** es en el que se almacena la información de la posición del robot. El control de la herramienta no está disponible desde el *teach pendant*.

CAPÍTULO 6

PRUEBAS Y RESULTADOS EXPERIMENTALES

6.1. PRUEBAS Y RESULTADOS EXPERIMENTALES

A continuación, se exponen las distintas pruebas y resultados que se han ido realizando y obteniendo a lo largo del desarrollo de este trabajo, así como los problemas que se hayan podido ir presentando.

6.1.1. Pruebas con los *drivers* y los motores

Para comprobar la precisión de los motores, así como para hacer las primeras pruebas con el código con la finalidad de obtener la relación entre el ángulo girado y el número de pasos necesario, se ha creado un pequeño dispositivo que se puede acoplar a un motor paso a paso NEMA 17, Figura 55 - Sistema de pruebas para la medición de ángulos.

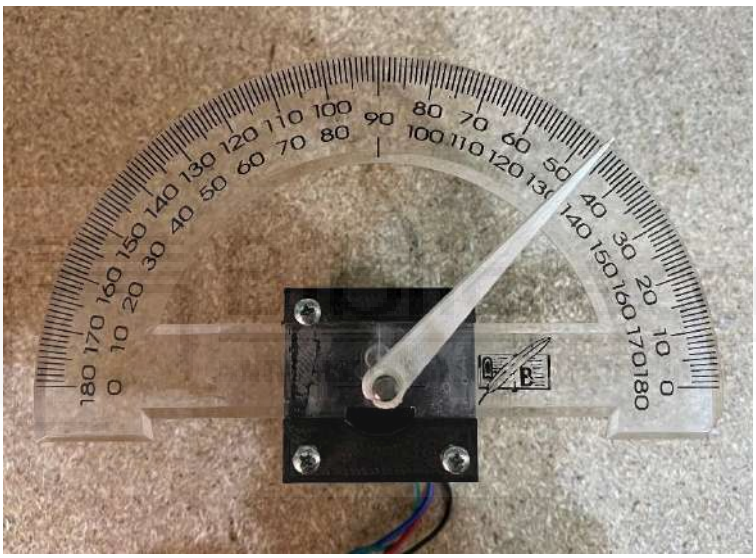


Figura 55 - Sistema de pruebas para la medición de ángulos

Este dispositivo se compone de un medidor de ángulos pegado a un bastidor y un dial impresos en 3D con el que visualizar de una manera cómoda cómo el ángulo de giro del eje del motor. Con este montaje se llegó a la siguiente expresión:

$$steps = \frac{S * ángulo}{360.0} * microstep * rt$$

Donde:

- La ***constante S*** viene dada por la relación entre el ángulo de giro máximo del motor (360.0°) y el ángulo recorrido por cada paso. Para los motores utilizados, el ángulo recorrido por cada paso es de 1.8°. En este caso, S = 200, que significa que son necesarios 200 pasos para que el motor gire una vuelta completa.

- La ***variable microstep*** depende de la configuración de cada *driver*. En este trabajo solo puede tomar los valores 1 y 32, según si el *driver* está configurado en *full-step* o 1/32 *step*.
- La ***variable rt*** es la relación de transmisión y dependerá de cada articulación.

Tras comprobar que la expresión era correcta, se pudo observar que el posicionamiento del motor era muy preciso y, sobre todo, la repetitividad de la posición alcanzada era también muy buena. Además, durante las pruebas no se observó que la aparición de desfases de su posición respecto a la posición deseada.

6.1.2. Sobrecalentamiento de los motores y ruido eléctrico

Durante las pruebas iniciales con el robot, se observó que todos los motores se encontraban a una temperatura elevada, siendo imposible tocar algunos de ellos durante más de un segundo. Además, también se pudo escuchar ruido de alta frecuencia proveniente de los motores.

En un primer momento se barajó la posibilidad de que los drivers (DRV8825 en este caso) que controlan a cada uno de los motores no estuvieran debidamente ajustados. Por tanto, el primer paso para encontrar una posible solución al problema fue comprobar que los drivers estuvieran configurados correctamente. El fabricante indica que, midiendo con un voltímetro sobre el potenciómetro que incorporan (cerca de una esquina) para ajustarlos, Figura 16 - Breakout board DRV8825 - Con y sin disipador de calor, el voltaje obtenido viene dado por la siguiente expresión:

$$Current\ Limit = 2 * VREF$$

Donde:

- ***Current Limit*** es la intensidad máxima por fase que puede soportar el motor.
- ***VREF*** es el voltaje medido en el potenciómetro.

Teniendo en cuenta esta expresión, se calcularon los valores de VREF para cada uno de los motores mostrados en la Tabla 12 - Tabla de motores y propiedades de configuración.

MOTOR	ARTICULACIÓN	I/fase [A]	70% I/fase [A]	I/fase real [A]	VREF [V]
17HD48002H-22B	q1	1.7	1.19	0.95	0.85
17HS6401S	q2	1.7	1.19	1.06	0.85
17HS15-1504S-X1	q3	1.5	1.05	0.96	0.75
14HS17-0504S	q4	0.5	0.35	0.26	0.25
14HS10-0404S	q5	0.4	0.28	0.22	0.2
14HS10-0404S	q6	0.4	0.28	0.22	0.2

Tabla 12 - Tabla de motores y propiedades de configuración

El fabricante de los *drivers* recomienda que el ajuste de la intensidad que atraviesa los motores, si se mide directamente la intensidad de fase con un amperímetro, ha de ser un 70% del valor máximo; mientras que, si se ajusta con la expresión anterior, se ha de calcular con el valor máximo de la intensidad por fase. Esto se debe a que la expresión anterior es una aproximación del valor real de la intensidad que recorre cada uno de los bobinados.

En un primer momento, el ajuste de los *drivers* se hizo con la expresión anterior, y posteriormente se comprobó que, efectivamente, el valor es una aproximación del valor real de la intensidad, tal y como se puede ver en la columna “I/fase real [A]” de la Tabla 12 - Tabla de motores y propiedades de configuración, donde se midió el valor real de la intensidad en las bobinas.

No obstante, aun teniendo la intensidad valores inferiores al máximo y al 70%, los motores, estos seguían a una temperatura demasiado elevada, siendo peligroso si se alcanzan 80 °C, puesto que puede provocar que los imanes permanentes de los motores se desmagneticen y quede el motor completamente inservible.

Para solucionar el problema del ruido eléctrico, se colocaron las articulaciones en posiciones desfavorables y se ajustaron cada uno de los *drivers* a un valor tal que los motores pudieran aguantar el peso del robot, y que el calentamiento no fuera excesivo. Sin embargo, el calor de los motores no parecía desaparecer por completo o bajar a unos niveles aceptables para el caso de un robot educativo, por lo que se optó por añadir un sistema de refrigeración.

Cabe mencionar que los motores paso a paso en bucle abierto (sin realimentación de la corriente que los atraviesa) están diseñados de manera que su funcionamiento normal sea a una temperatura elevada, pero, puesto que la gran mayoría de motores están en zonas accesibles y su uso es educativo, fue necesario tener en consideración este calentamiento.

6.1.2.1. Enfriamiento de los motores

Para evitar que los motores, así como las distintas piezas de plástico, sufran daño debido al calentamiento propio del funcionamiento de los motores, se han dispuesto una serie de elementos para favorecer su refrigeración. Para ello, se han combinado elementos de refrigeración pasiva y activa.

Se han utilizado ventiladores y disipadores de calor para proporcionar una buena refrigeración en los motores. Con ello se ha podido aumentar la corriente de los motores hasta el límite del fabricante sin correr riesgo de rotura por desmagnetización de los imanes permanentes debido a la temperatura.

Los ventiladores y disipadores, según el caso, se han colocado en distintas zonas teniendo en cuenta el espacio disponible para ello (Figura 56 - Motor junto a disipador de calor en la articulación q2). No en todos los motores se ha utilizado una combinación de ventilador y disipador de calor, siendo el conjunto de ambos utilizado en las articulaciones q1, q2, q4; únicamente ventilador en q5 y q6; y sin disipador ni ventilador en q3.

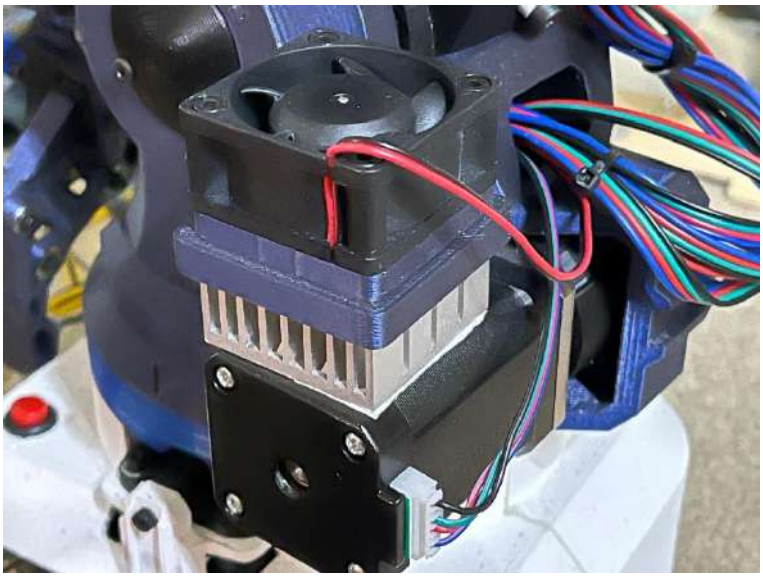


Figura 56 - Motor junto a disipador de calor en la articulación q2

En este punto, en el que se dispone de un sistema de refrigeración, se elevó la potencia suministrada a todos los motores (excepto q3) para que pudieran soportar mejor las cargas solicitadas en cada articulación.

6.1.3. Incorporación de caja reductora en la articulación del hombro (q2)

En un primer momento, el movimiento de la articulación q2 no era el esperado. Tras descartar un mal ajuste en el *driver* correspondiente, se planteó que el motor comprado no tuviera las especificaciones por las que se compró. Por tanto, se barajaron varias opciones: comprar un motor nuevo, comprar una reductora para obtener más par en el eje de salida, comprar un motor con reductora incorporada, o diseñar y fabricar una reductora propia. Finalmente, se optó por diseñar una caja reductora simple con la que conseguir mayor par para mover la articulación. Teniendo en cuenta la siguiente expresión:

$$P = T * w$$

Donde:

- **P** es la potencia del eje.
- **T** es el par del eje.
- **w** es la velocidad angular del eje.

Considerando que la potencia del eje de entrada es igual a la potencia del eje de salida, se tiene que:

$$T_1 * w_1 = T_2 * w_2 \rightarrow \frac{T_1}{T_2} = \frac{w_2}{w_1}$$

Además, si se tiene en cuenta la expresión de la relación de transmisión:

$$rt = \frac{w_2}{w_1} = \frac{Z_1}{Z_2}$$

Donde **Z** es el número de dientes del engranaje.

Por tanto, se obtiene la expresión siguiente:

$$\frac{T_1}{T_2} = \frac{Z_1}{Z_2} \rightarrow \frac{T_2}{T_1} = \frac{Z_2}{Z_1}$$

Teniendo en cuenta que el par de entrada (T_1) es de, aproximadamente, 70 Ncm (según especificaciones del motor), y que para mover la articulación de manera correcta se necesitan, al menos, 101.5 Ncm de par pico en el motor, según lo obtenido en 5.2 - SELECCIÓN DE MOTORES PASO A PASO, se estableció que eran necesarios 120 Ncm para mover con seguridad la articulación en T_2.

Esto hace que se llegue a la siguiente expresión:

$$Z_2 = 1.7 * Z_1$$

En este caso, se seleccionó un valor arbitrario de $Z_1 = 10$, por lo que el valor de $Z_2 = 17$. De los valores del número de dientes de ambas ruedas dentadas, se obtiene que la relación de transmisión de la reductora es $rt \cong 0.6$, valor que se ha de tener en cuenta a la hora de programar el movimiento de la articulación en la controladora.

6.1.3.1. Fabricación y montaje de la caja reductora en el robot

Una vez establecidos los parámetros de diseño, se realizaron los distintos elementos en CAD para su posterior impresión 3D y montaje.

Por una parte, para la impresión, se decidió imprimir los engranajes con un relleno del 70% para dotarlos de mayor resistencia mecánica. Además, los engranajes tienen un espesor de 20 mm para aumentar la superficie de contacto entre dientes y, así, repartir la fuerza ejercida en el diente y aumentar la durabilidad de los engranajes.

Por otra parte, se han utilizado varillas roscadas de 5 mm de diámetro de acero cincado, y cada eje va apoyado sobre un rodamiento MR105. Los engranajes, debido a las limitaciones a la hora de fabricar la caja reductora, se han unido a los ejes mediante resina epoxi, lo que limita enormemente su reparabilidad.

No obstante, tras realizar una serie de pruebas, la reductora falló dejándola inservible, por lo que se descartó la idea de fabricar una propia con piezas impresas, y se optó por buscar un nuevo motor con reductora ya incluida. En cualquier caso, el proceso de montaje de esta reductora se puede encontrar en el anexo 9.4.2.1 - Caja reductora.

6.1.3.2. Uso de un motor con reductora planetaria

Tras el intento de fabricar una reductora a medida, se optó por comprar un motor que ya la incorpore de fábrica. Para ello, se buscó un motor con reductora que superase holgadamente el par necesario para mover la articulación. Finalmente, se eligió un conjunto que ofreciese 120 Ncm (según el fabricante), del mismo modo que se planteó en el apartado 6.1.3 - Incorporación de caja reductora.

Figura 57 - Comparación del motor con y sin reductora

En la Figura 57 - Comparación del motor con y sin reductora, aunque no se puede apreciar a simple vista, en la etiqueta se indica que ambos motores son idénticos, por lo que, si el motor de la derecha puede ejercer un par de 70 Ncm, con la reductora de 3.71:1 se obtienen, aproximadamente, 260 Ncm, lo que hace estar muy por encima del valor requerido por la simulación, y el valor planteado. El montaje de este motor con reductora se encuentra en el anexo 9.4.2.2 - Motor con caja reductora.

Tras montar el motor con la reductora y hacer las pruebas pertinentes, se observó que el motor cumplía con creces las necesidades de par que solicita el robot, tal y como se puede ver en la Figura 58 - Robot completamente estirado, donde se puede ver el robot en una pose bastante desfavorable y el motor puede aguantar el peso del robot sin problema.

Figura 58 - Robot completamente estirado

6.1.4. Modificación de la PCB

Durante las pruebas de funcionamiento del robot, se comprobó que el control de la herramienta, en el caso de este trabajo, una herramienta controlada por un servo RC, no funcionaba como se esperaba, por lo que se tuvo que retirar el optoacoplador 6N136S, el regulador de tensión LM7806, y varios componentes como resistencias o condensadores que eran necesarios para el funcionamiento de dichos componentes. En la Figura 59 - Diferencia entre el diseño original (izquierda) y la modificación (derecha) se puede ver la modificación realizada a la PCB.

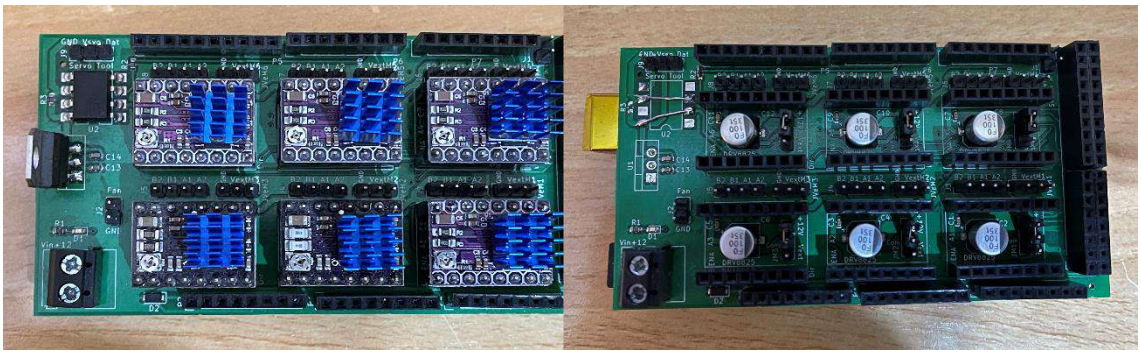


Figura 59 - Diferencia entre el diseño original (izquierda) y la modificación (derecha)

Como se puede observar, se han retirado los componentes, pero también se han añadido una serie de puentes para realizar conexiones de la señal que controla el servo RC y para alimentarlo a 5 V tomados directamente de la Arduino.

Finalmente, tras realizar la modificación, el servo RC se comportó como se esperaba abriendo y cerrando la pinza.

6.1.5. Problemas detectados en el comportamiento de los motores

Durante las pruebas de movimiento, se comprobó que el motor de la articulación q5 no conseguía soportar el peso de los elementos de la muñeca, es decir, se estaba superando el par de retención del motor. Esto se debe a que el par de retención no se correspondía con el par ejercido durante el movimiento del eje del motor, siendo este último inferior al par de retención.

Para seleccionar un motor adecuado, se debería disponer de las curvas de par características de cada motor, información de la cual no se ha dispuesto de ningún motor durante el desarrollo de este trabajo.

CAPÍTULO 7

CONCLUSIONES Y POSIBLES TRABAJOS FUTUROS

7.1. CONCLUSIONES Y POSIBLES TRABAJOS FUTUROS

Se puede concluir que, aunque se han encontrado numerosas dificultades y diversos obstáculos a la hora de desarrollar este trabajo, se han cumplido los objetivos marcados en el apartado 1.2 - OBJETIVOS. Este trabajo ofrece una base sólida sobre la que poder seguir desarrollando nuevas ideas, por lo que está sujeto a mejoras y modificaciones, ya sean de diseño electrónico, mecánico o programación. A continuación, se exponen una serie de posibles mejoras y modificaciones.

En cuanto al diseño de la electrónica y la programación, se puede plantear el uso de una Raspberry Pi junto a una Arduino. De este modo, la controladora puede ser la Raspberry Pi y que sea ésta la que comanda a la Arduino para controlar los motores a través de los drivers. También se puede plantear el uso de motores con *encoders* con lo que se mejoraría el posicionamiento del robot. En cualquier caso, cualquier modificación de este tipo requiere que se modifique el diseño de la PCB.

En el caso de contar con una Raspberry Pi, se puede plantear la creación de una paletera o *teach pendant* conectado a la Raspberry Pi con una pantalla táctil y, de este modo, poder controlar el robot desde ahí, además desde ARTE. Adicionalmente, se puede plantear el uso del puerto Ethernet de la Raspberry Pi para realizar todas las comunicaciones mediante los protocolos TCP o UDP, por ejemplo.

Un posible trabajo también puede ser la inclusión de una cámara para poder realizar un control basado en visión por computador, ya sea conectando la cámara al PC donde se ejecute MATLAB con ARTE, o bien, si se dispone de una Raspberry Pi, en la Raspberry Pi.

Si se trata de algún trabajo relacionado con los aspectos más mecánicos del robot, se puede plantear la inclusión de un sistema de topes con finales de carrera. Con ellos se detectaría si alguna articulación ha alcanzado el final del recorrido. Esto es importante en el caso de que algún motor pierda pasos y el control no sepa realmente dónde se encuentra el robot. Adicionalmente, se propone plantear la inclusión de un sistema de marcas de referencia para hacer una correcta calibración del robot.

CAPÍTULO 8

BIBLIOGRAFÍA

8.1. BIBLIOGRAFÍA

[1] R. Staff, «ROBOT,» Botmag, 02 12 2010. [En línea]. Available: http://www.botmag.com/the-rise-and-fall-of-unimation-inc-story-of-robotics-innovation-triumph-that-changed-the-world/. [Último acceso: 13 12 2020].

[2] J. Co, «APIfriends,» APIfriends, 23 05 2019. [En línea]. Available: https://apifriends.com/digital-strategy/devops-pipeline/. [Último acceso: 14 01 2021].

[3] A. G. Aparicio, «ARTE: A ROBOTICS TOOLBOX FOR EDUCATION,» 2012. [En línea]. Available: http://arvc.umh.es/arte/practicals/pr7/add_your_robot.pdf. [Último acceso: 12 12 2020].

[4] Á. M. Benito, «Robots in Action,» [En línea]. Available: https://robotsinaction.com/tag/unimate/. [Último acceso: 13 12 2020].

[5] «Dynastat memory aids Unimate robot,» *Electrical Engineering,* vol. 80, nº 4, pp. 319-320, 1961.

[6] Á. L. M., «HACKADAY.IO,» [En línea]. Available: https://hackaday.io/project/12989-thor. [Último acceso: 15 01 2021].

[7] D. Florian, «Dr. D-Flo,» [En línea]. Available: https://www.drdflo.com/pages/Projects/6Axis-Robotic-Arm.html. [Último acceso: 13 09 2020].

[8] NIRYO, «NIRYO,» [En línea]. Available: https://niryo.com/niryo-one/. [Último acceso: 13 09 2020].

[9] Direct Industry, «Direct Industry,» Direct Industry, [En línea]. Available: https://www.directindustry.es/prod/abb-robotics/product-30265-169114.html. [Último acceso: 15 01 2021].

[10] Pololu, «Pololu,» [En línea]. Available: https://www.pololu.com/category/120/stepper-motor-drivers. [Último acceso: 13 09 2020].

[11] microPaP, «microPaP,» [En línea]. Available: http://www.micropap.com/index.php/blog/motores/item/19-motores-paso-a-paso-vs-servos. [Último acceso: 02 11 2020].

[12] «330ohms,» [En línea]. Available: https://blog.330ohms.com/2016/02/09/motores-a-pasos-unipolares-o-bipolares/. [Último acceso: 13 09 2020].

[13] A. R. Gómez, *SISTEMA INALÁMBRICO BASADO EN RASPBERRY PI PARA EL CONTROL DE PERÍMETROS AD HOC,* Elche, 2020.

[14] RepRap, «RepRap,» 07 03 2019. [En línea]. Available: https://reprap.org/wiki/RAMPS_1.6. [Último acceso: 13 09 2020].

[15] AirSpayce Pty Ltd., «https://www.airspayce.com/,» https://www.airspayce.com/, 20 04 2020. [En línea]. Available: https://www.airspayce.com/mikem/arduino/AccelStepper/index.html. [Último acceso: 12 12 2020].

[16] D. Collins, «Linear Motion Tips - A Design World Resource,» 11 21 2017. [En línea]. Available: https://www.linearmotiontips.com/microstepping-basics/. [Último acceso: 27 09 2020].

[17] Trinamic, «TRINAMIC - Now part of Maxim Integrated,» TRINAMIC, [En línea]. Available: https://www.trinamic.com/technology/motor-control-technology/microstepping/. [Último acceso: 27 09 2020].

[18] NIRYO ONE - Base, «YouTube,» NIRYO ONE, 09 09 2018. [En línea]. Available: https://www.youtube.com/watch?v=3f-Ecz0IYLY&feature=emb_logo. [Último acceso: 22 09 2020].

[19] NIRYO ONE - Hombro, «YouTube,» NIRYO ONE, 09 09 2018. [En línea]. Available: https://www.youtube.com/watch?v=f9BH28SffIw&feature=emb_logo. [Último acceso: 23 09 2020].

ANEXOS

9.1. CÓDIGO DE LA CONTROLADORA

```
//    LIBRARIES    //
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <AccelStepper.h>
#include <MultiStepper.h>
#include <Servo.h>

////////////////////////////////////////////////////////////////////////////////
//////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////

//    FSM STATE ENUMERATION    //

enum State {
  idle_st,
  home_st,
  calibrate_st,
  move_st,
  tool_st,
};

State currentState; // Variable for states assignment

////////////////////////////////////////////////////////////////////////////////
//////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////

//    DIGITAL PIN ASSIGNMENT    //

#define ena_q1 49 // Arduino pin for q1 driver enabling
#define ena_q2 48 // Arduino pin for q2 driver enabling
#define ena_q3 43 // Arduino pin for q3 driver enabling
#define ena_q4 42 // Arduino pin for q4 driver enabling
#define ena_q5 37 // Arduino pin for q5 driver enabling
#define ena_q6 36 // Arduino pin for q6 driver enabling

#define pin_tool 2  // Arduino pin for servo tool

#define Switch 31 // Arduino pin for robot base button

#define MSTP 30 // Arduino pin for microstepping mode change

////////////////////////////////////////////////////////////////////////////////
//////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////

//    VARIABLE - CONSTANTS - OBJECTS - DEFINE STATEMENTS    //

#define max_size 50       // Max serial messages buffer size
char cmd[max_size];       // Variable for serial inputs
char cmd_aux[5];          // Auxiliar variable for general parsing
size_t serial_index = 0;  // Index variable for storing serial data on cmd
variable

Servo tool; // Object for tool
unsigned int servo_pos = 0;
bool tool_mv = false; // Tool state

// Stepper objects
AccelStepper q1(1, 47, 45);
AccelStepper q2(1, 46, 44);
```

```
AccelStepper q3(1, 41, 39);
AccelStepper q4(1, 40, 38);
AccelStepper q5(1, 35, 33);
AccelStepper q6(1, 34, 32);

//MultiStepper motors;  // MultiStepper object for handling all motors

#define mstp 32 // Microstepping value. Fixed value due to PCB design

bool calibrate = false; // Flag for first calibration
bool move_enable = false; // Flag for movement enabling

const int microstepping [2] = {1, 32}; // Microstepping selection: full-step
or 1/32 of step
uint8_t ena_status [6] = {HIGH, HIGH, HIGH, HIGH, HIGH, HIGH};  // Array with
driver state. Inverse logic


const float Zb = 97.0;    // Robot base teeth
const float Zs = 133.0;   // Robot shoulder teeth
const float Ze = 126.0;   // Robot elbow teeth
const float Zps = 20.0;   // Robot shoulder pinion theeth
const float Zp = 16.0;    // Robot base and elbow pinion teeth

const float rtb = Zp / Zb; // Gear ratio for robot base
const float rts = Zps / Zs; // Gear ratio for robot shoulder
const float rte = Zp / Ze; // Gear ratio for robot elbow

long q[] = {0, 0, 0, 0, 0, 0};   // Long int array for joint positions
////////////////////////////////////////////////////////////////////////////
//////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////

//   FUNCTION PROTOTYPES    //

void state_machine();
void read_serial();
void calibrate_joints();
void move_tool(char mov_tl);

////////////////////////////////////////////////////////////////////////////
//////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////

void setup() {
  tool.attach(pin_tool);  // Attach for servo tool pin


  q1.setMaxSpeed(765);  // Max speed configuration for each motor. This speed
is steps/s
  q2.setMaxSpeed(765 * 3); // 43°/s
  q3.setMaxSpeed(1013); // 57°/s
  q4.setMaxSpeed(1280); // 72°/s
  q5.setMaxSpeed(1280);
  q6.setMaxSpeed(1280);

  q1.setAcceleration(1000); // Max acceleration configuration for each motor.
This acceleration is steps/s^2
  q2.setAcceleration(1000 * 3); // 72°/s^2
  q3.setAcceleration(1000); // 86°/s^2
  q4.setAcceleration(1778); // 100°/s^2
  q5.setAcceleration(1000); // 115°/s^2
  q6.setAcceleration(1000); // 129°/s^2


  pinMode(Switch, INPUT_PULLUP);  // Input mode with internal pull-up resistor
```

```
  pinMode(MSTP, OUTPUT);
  digitalWrite(MSTP, HIGH);

  pinMode(ena_q1, OUTPUT);
  pinMode(ena_q2, OUTPUT);
  pinMode(ena_q3, OUTPUT);
  pinMode(ena_q4, OUTPUT);
  pinMode(ena_q5, OUTPUT);
  pinMode(ena_q6, OUTPUT);


  // Next lines force driver disabling for security
  digitalWrite(ena_q1, ena_status[0]);
  digitalWrite(ena_q2, ena_status[1]);
  digitalWrite(ena_q3, ena_status[2]);
  digitalWrite(ena_q4, ena_status[3]);
  digitalWrite(ena_q5, ena_status[4]);
  digitalWrite(ena_q6, ena_status[5]);

  // Serial comunication setup
  Serial.begin(9600);
  Serial.println("robot_online");   // Message for ARTE
}


void loop() {

  state_machine();
  delayMicroseconds(10);
  read_serial();

  q1.run();
  q2.run();
  q3.run();
  q4.run();
  q5.run();
  q6.run();

}


void state_machine() {
  if (currentState == idle_st) {
    if (strcmp(cmd_aux, "HOME") == 0) {
      currentState = home_st;
    } else if (strcmp(cmd_aux, "MOVQ") == 0) {
      currentState = move_st;
    } else if (strcmp(cmd_aux, "TOOL") == 0) {
      currentState = tool_st;
    } else if (calibrate == false) {
      currentState = calibrate_st;
      calibrate = true;  //
    }

  } else if (currentState == home_st) {
    Serial.println("Home");
    q1.moveTo(0);
    q2.moveTo(0);
    q3.moveTo(0);
    q4.moveTo(0);
    q5.moveTo(0);
    q6.moveTo(0);
    currentState = idle_st;
    cmd[0] = '\0';
    cmd_aux[0] = '\0';

  } else if (currentState == calibrate_st) {
```

```
    move_tool(1);
    calibrate_joints();
    currentState = idle_st;

  } else if (currentState == tool_st) {
    move_tool(cmd[5]);
    currentState = idle_st;
    cmd[0] = '\0';
    cmd_aux[0] = '\0';

  } else if (currentState == move_st) {
    char *pos;  // String pointer for message parsing
    char token = '_';
    uint8_t counter = 0;  // Counter for storing data into q arry
    // Next lines split cmd message into smaller parts using tokens
    pos = strtok(cmd, "_");
    while (pos != NULL) {
      pos = strtok(NULL, "_");
      if (pos == NULL) {
        break;
      }
      q[counter] = atol(pos); // Converts string data into float

      counter++;
    }

    // Next lines convert received (angle) data into steps for each motor
    // This expressions are simplified from: (q[x] * 360.0 * rt) / ((360.0 *
microstepping[1]) / 1.8)
    // Motors 4 - 5 - 6 are direct drive
    q[0] = long(ceil(q[0] * 6400.0 * rtb * 32.0 / 360.0));
    q1.moveTo(q[0]);

    q[1] = long(ceil(q[1] * 6400.0 * rts * 32.0 * 3.0 / 360.0));
    q2.moveTo(q[1]);

    q[2] = long(ceil(q[2] * 6400.0 * rte * 32.0 / 360.0));
    q3.moveTo(q[2]);

    q[3] = long(ceil(q[3] * 6400.0 / 360.0));
    q4.moveTo(q[3]);

    q[4] = long(ceil(q[4] * 6400.0 / 360.0));
    q5.moveTo(q[4]);

    q[5] = long(ceil(q[5] * 6400.0 / 360.0));
    q6.moveTo(q[5]);

    move_enable = true;
    currentState = idle_st;

    cmd[0] = '\0';
    cmd_aux[0] = '\0';

  } else {  // Default state
    currentState = idle_st;
  }
}


void read_serial() {
  char serial_in; // Variable for incoming serial data

  // Serial data readings
  while (Serial.available() > 0) {
    delay(2); // Adding this little delay is highly recommended
    serial_in = Serial.read();
    if (serial_in != '\n') {
```

```
      cmd[serial_index] = serial_in;
      serial_index++;
    } else {
      serial_index = 0;
      strncpy(cmd_aux, cmd, 4);
    }
  }
}

void calibrate_joints() {

  unsigned long lastDebounceTime = 0;  // Last time button was pressed
  unsigned long debounceDelay = 100;   // Time for debouncing
  uint8_t buttonState;
  uint8_t lastButtonState = HIGH;  // Previous button state HIGH => Pull-Up //
LOW => Pull-Down

  Serial.println("Starting calibration process");
  Serial.println("Put each joint on start position");
  Serial.println("Press robot button once each joint is on the starting
position");

  for (uint8_t n_driver = 0; n_driver < 6;) {
    uint8_t readBtn = digitalRead(Switch);

    if (readBtn != lastButtonState) {
      lastDebounceTime = millis();
    }

    if ((millis() - lastDebounceTime) > debounceDelay) {
      if (readBtn != buttonState) {
        buttonState = readBtn;
        if (buttonState == LOW) {

          lastButtonState = !lastButtonState; // Change button state
          ena_status[n_driver] = !ena_status[n_driver]; // Change driver state

          if (n_driver == 0) { // Driver habilitation block
            digitalWrite(ena_q1, ena_status[0]);
            q1.setCurrentPosition(0); // Sets motor start position
            Serial.println("Joint 1 (base) enabled and calibrated");
            n_driver++;
          } else if (n_driver == 1) {
            digitalWrite(ena_q2, ena_status[1]);
            q2.setCurrentPosition(0);
            Serial.println("Joint 2 (shoulder) enabled and calibrated");
            n_driver++;
          } else if (n_driver == 2) {
            digitalWrite(ena_q3, ena_status[2]);
            q3.setCurrentPosition(0);
            Serial.println("Joint 3 (elbow) enabled and calibrated");
            n_driver++;
          } else if (n_driver == 3) {
            digitalWrite(ena_q4, ena_status[3]);
            q4.setCurrentPosition(0);
            Serial.println("Joint 4 (forearm) enabled and calibrated");
            n_driver++;
          } else if (n_driver == 4) {
            digitalWrite(ena_q5, ena_status[4]);
            q5.setCurrentPosition(0);
            Serial.println("Joint 5 (wirst 1) enabled and calibrated");
            n_driver++;
          } else if (n_driver == 5) {
            digitalWrite(ena_q6, ena_status[5]);
            q6.setCurrentPosition(0);
            Serial.println("Joint 6 (wirst 2) enabled and calibrated");
            break;
          }
```

```
        }
      }
    }
    lastButtonState = readBtn;
  }
  Serial.println("robot_ready!");
}

void move_tool(char mov_tl) {
  int closed = 10;
  int opened = 100;

  if (mov_tl == '1') { // Closed as true
    tool.write(closed);
    delay(1000);
  } else if (mov_tl == '0') { // Opened as false
    tool.write(opened);
    delay(1000);
  }
}
```

9.2. CÓDIGOS DE MATLAB

9.2.1. parameters.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   PARAMETERS Returns a data structure containing the parameters of the
%   ABB IRB140.
%
%   Author: Arturo Gil. Universidad Miguel Hernandez de Elche.
%   email: arturo.gil@umh.es date:   09/01/2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Copyright (C) 2012, by Arturo Gil Aparicio
%
% This file is part of ARTE (A Robotics Toolbox for Education).
%
% ARTE is free software: you can redistribute it and/or modify
% it under the terms of the GNU Lesser General Public License as published by
% the Free Software Foundation, either version 3 of the License, or
% (at your option) any later version.
%
% ARTE is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU Lesser General Public License for more details.
%
% You should have received a copy of the GNU Leser General Public License
% along with ARTE.  If not, see <http://www.gnu.org/licenses/>.
function robot = parameters()

robot.name= 'NIRYO';

robot.DH.theta= '[q(1) q(2)-pi/2 q(3) q(4) q(5) q(6)-pi]';
robot.DH.d='[0.183 0 0              0.2215 0        0.0745]';
robot.DH.a='[0 0.21 0.03            0      0        0]';
robot.DH.alpha= '[-pi/2 0 -pi/2 pi/2 -pi/2 0]';
robot.J=[];

robot.inversekinematic_fn = 'inversekinematic_niryo(robot, T)';
robot.directkinematic_fn = 'directkinematic(robot, q)';

% Serial parameters
robot.port = 'COM3';        % Change this value as needed
robot.baudrate = 9600;        % Don't change this value
robot.serial = serial(robot.port, 'Baudrate', robot.baudrate);

%number of degrees of freedom
robot.DOF = 6;

%rotational: 0, translational: 1
robot.kind=['R' 'R' 'R' 'R' 'R' 'R'];

%minimum and maximum rotation angle in rad
robot.maxangle =[deg2rad(-175) deg2rad(175); %Axis 1, minimum, maximum
                deg2rad(-36.7) deg2rad(90); %Axis 2, minimum, maximum
                deg2rad(-80) deg2rad(70); %Axis 3, minimum, maximum
                deg2rad(-175) deg2rad(175); %Axis 4, minimum, maximum
                deg2rad(-40) deg2rad(110); %Axis 5, minimum, maximum
                deg2rad(-147.5) deg2rad(147.5)]; %Axis 6, minimum, maximum

%maximum absolute speed of each joint rad/s or m/s
robot.velmax = [deg2rad(43); %Axis 1, rad/s
                deg2rad(43); %Axis 2, rad/s
                deg2rad(57); %Axis 3, rad/s
                deg2rad(72); %Axis 4, rad/s
                deg2rad(72); %Axis 5, rad/s
                deg2rad(72)];%Axis 6, rad/s
```

```
robot.accelmax=robot.velmax/0.1; % 0.1 is here an acceleration time

% end effectors maximum velocity
robot.linear_velmax = 2.5; %m/s

%base reference system
robot.T0 = eye(4);

%INITIALIZATION OF VARIABLES REQUIRED FOR THE SIMULATION
%position, velocity and acceleration
robot=init_sim_variables(robot);
robot.path = pwd;

% GRAPHICS
robot.graphical.has_graphics=1;
robot.graphical.color = [82 108 240]./255;
%for transparency
robot.graphical.draw_transparent=0;
%draw DH systems
robot.graphical.draw_axes=1;
%DH system length and Font size, standard is 1/10. Select 2/20, 3/30 for
%bigger robots
robot.graphical.axes_scale=1;
%adjust for a default view of the robot
robot.axis=[-0.5 0.75 -0.75 0.75 0 1.1];
%read graphics files
robot = read_graphics(robot);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% iterative solution   ===   Esto es para las inversas especiales
robot.parameters.stop_iterations = 500;
robot.parameters.step_time = 0.01;
%Error in XYZ to stop inverse kinematics
robot.parameters.epsilonXYZ=0.0000001;
robot.parameters.epsilonOrientation=0.0000001; %
%Error in Quaternion to stop inverse kinematics.
robot.parameters.epsilonQ=0.000001;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DYNAMIC PARAMETERS
%   WARNING! These parameters do not correspond to the actual IRB 140
%   robot. They have been introduced to demonstrate the necessity of
%   simulating the robot and should be used only for educational purposes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
robot.has_dynamics=1;

%consider friction in the computations
robot.dynamics.friction=0;

%link masses (kg)
% robot.dynamics.masses=[25 27 15 10 2.5 1.5];
robot.dynamics.masses = [0.7 1 0.35 0.6 0.5 0.15];

%COM of each link with respect to own reference system
robot.dynamics.r_com=[-0.055    0.05    0; %(rx, ry, rz) link 1
                      -0.05     -0.01   0; %(rx, ry, rz) link 2
                      0         0       0;  %(rx, ry, rz) link 3
                      0         0       -0.03;%(rx, ry, rz) link 4
                      0         0       0;%(rx, ry, rz) link 5
                      0         0       0];%(rx, ry, rz) link 6

%Inertia matrices of each link with respect to its D-H reference system.
%                       Ixx     Iyy     Izz     Ixy Iyz Ixz, for each row
robot.dynamics.Inertia=[0       2e-3    0       0   0   0;  % Hombro
                        0.01    0       0.25    0   0   0;  % Brazo
                        1e-4    3e-3    0       0   0   0;  % Codo
```

```
                        0.03    0.01    0.03    0   0   0;  % Antebrazo
                        4.1e-4  4.1e-4  4.1e-4  0   0   0;  % Muñeca
                        5e-4    5e-4    5e-5    0   0   0]; % End tool

% https://es.wikipedia.org/wiki/Teorema_de_Steiner
% https://en.wikipedia.org/wiki/List_of_moments_of_inertia

% robot.dynamics.Inertia=[0      0.35     0     0   0   0;
%                         .13     .524  .539    0   0   0;
%                         .066  .086    .0125   0   0   0;
%                         1.8e-3    1.3e-3  1.8e-3  0   0   0;
%                         .3e-3 .4e-3   .3e-3   0   0   0;
%                         .15e-3    .15e-3  .04e-3  0   0   0];


%robot.motors=load_motors([5 5 5 4 4 4]);
%Speed reductor at each joint
robot.motors.G=[0.17 0.12 0.13 1 1 1];
```

9.2.2. directkinematic.m

```
%   DIRECTKINEMATIC     Direct Kinematic for serial robots.
%
%   T = DIRECTKINEMATIC(robot, Q) returns the transformation matrix T
%   of the end effector according to the vector q of joint coordinates.
%
%   See also DH
%
%   Author: Arturo Gil. Universidad Miguel Hernandez de Elche.
%   email: arturo.gil@umh.es date:   01/04/2012

% Copyright (C) 2012, by Arturo Gil Aparicio
%
% This file is part of ARTE (A Robotics Toolbox for Education).
%
% ARTE is free software: you can redistribute it and/or modify
% it under the terms of the GNU Lesser General Public License as published by
% the Free Software Foundation, either version 3 of the License, or
% (at your option) any later version.
%
% ARTE is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU Lesser General Public License for more details.
%
% You should have received a copy of the GNU Leser General Public License
% along with ARTE.  If not, see <http://www.gnu.org/licenses/>.
function T = directkinematic(robot, q)

%In the case of a parallel robot, switch to its particular direct kinematic
%function
if isfield(robot, 'parallel')
    %Call specific direct kinematic function for parallel robots
    T = eval(robot.directkinematic_fn);
    return;
end

%compute direct kinematics for serial robotics
theta = eval(robot.DH.theta);
d = eval(robot.DH.d);
a = eval(robot.DH.a);
alfa = eval(robot.DH.alpha);

n=length(theta); %number of DOFs

if robot.debug
    fprintf('\nComputing direct kinematics for the %s robot with %d
DOFs\n',robot.name, n);
end
%load the position/orientation of the robot's base
T = robot.T0;

for i=1:n,
    T=T*dh(theta(i), d(i), a(i), alfa(i));
end

%if there is a tool attached to it, consider it in the computation of
% direct kinematics
% if isfield(robot, 'tool')
%     T=T*robot.tool.TCP; %dh(theta(i), d(i), a(i), alfa(i));
% end
```

9.2.3. inversekinematic_niryo.m

```
% Copyright (C) 2012, by Arturo Gil Aparicio
%
% This file is part of ARTE (A Robotics Toolbox for Education).
%
% ARTE is free software: you can redistribute it and/or modify
% it under the terms of the GNU Lesser General Public License as published by
% the Free Software Foundation, either version 3 of the License, or
% (at your option) any later version.
%
% ARTE is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU Lesser General Public License for more details.
%
% You should have received a copy of the GNU Leser General Public License
% along with ARTE.  If not, see <http://www.gnu.org/licenses/>.
function q = inversekinematic_niryo(robot, T)

%initialize q,
%eight possible solutions are generally feasible
q=zeros(6,8);

%Evaluate the parameters
d = eval(robot.DH.d);

%See geometry at the reference for this robot
L6=d(6);

%A1 = a(1);

%T= [ nx ox ax Px;
%     ny oy ay Py;
%     nz oz az Pz];
Px=T(1,4);
Py=T(2,4);
Pz=T(3,4);

%Compute the position of the wrist, being W the Z component of the end
effector's system
W = T(1:3,3);

% Pm: wrist position
Pm = [Px Py Pz]' - L6*W;

%first joint, two possible solutions admited:
% if q(1) is a solution, then q(1) + pi is also a solution
q1=atan2(Pm(2), Pm(1));


%solve for q2
q2_1=solve_for_theta2(robot, [q1 0 0 0 0 0 0], Pm);

q2_2=solve_for_theta2(robot, [q1+pi 0 0 0 0 0 0], Pm);

%solve for q3
q3_1=solve_for_theta3(robot, [q1 0 0 0 0 0 0], Pm);

q3_2=solve_for_theta3(robot, [q1+pi 0 0 0 0 0 0], Pm);


%Arrange solutions, there are 8 possible solutions so far.
% if q1 is a solution, q1* = q1 + pi is also a solution.
% For each (q1, q1*) there are two possible solutions
% for q2 and q3 (namely, elbow up and elbow up solutions)
```

```
% So far, we have 4 possible solutions. Howefer, for each triplet (theta1,
theta2, theta3),
% there exist two more possible solutions for the last three joints, generally
% called wrist up and wrist down solutions. For this reason,
%the next matrix doubles each column. For each two columns, two different
%configurations for theta4, theta5 and theta6 will be computed. These
%configurations are generally referred as wrist up and wrist down solution
q = [q1         q1         q1         q1       q1+pi    q1+pi    q1+pi    q1+pi;
     q2_1(1)    q2_1(1)    q2_1(2)    q2_1(2)  q2_2(1)  q2_2(1)  q2_2(2)  q2_2(2);
     q3_1(1)    q3_1(1)    q3_1(2)    q3_1(2)  q3_2(1)  q3_2(1)  q3_2(2)  q3_2(2);
     0          0          0          0         0       0        0        0;
     0          0          0          0         0       0        0        0;
     0          0          0          0         0       0        0        0];

%leave only the real part of the solutions
q=real(q);

%Note that in this robot, the joint q3 has a non-simmetrical range. In this
%case, the joint ranges from 60 deg to -219 deg, thus, the typical normalizing
%step is avoided in this angle (the next line is commented). When solving
%for the orientation, the solutions are normalized to the [-pi, pi] range
%only for the theta4, theta5 and theta6 joints.

%normalize q to [-pi, pi]
q(1,:) = normalize(q(1,:));
q(2,:) = normalize(q(2,:));

% solve for the last three joints
% for any of the possible combinations (theta1, theta2, theta3)
for i=1:2:size(q,2),
    % use solve_spherical_wrist2 for the particular orientation
    % of the systems in this ABB robot
    % use either the geometric or algebraic method.
    % the function solve_spherical_wrist2 is used due to the relative
    % orientation of the last three DH reference systems.

    %use either one algebraic method or the geometric
    %qtemp = solve_spherical_wrist2(robot, q(:,i), T, 1, 'geometric'); %wrist
up
    qtemp = solve_spherical_wrist2(robot, q(:,i), T, 1,'algebraic'); %wrist up
    qtemp(4:6)=normalize(qtemp(4:6));
    q(:,i)=qtemp;

    %qtemp = solve_spherical_wrist2(robot, q(:,i), T, -1, 'geometric'); %wrist
down
    qtemp = solve_spherical_wrist2(robot, q(:,i), T, -1, 'algebraic'); %wrist
down
    qtemp(4:6)=normalize(qtemp(4:6));
    q(:,i+1)=qtemp;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% solve for second joint theta2, two different
% solutions are returned, corresponding
% to elbow up and down solution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function q2 = solve_for_theta2(robot, q, Pm)

%Evaluate the parameters
d = eval(robot.DH.d);
a = eval(robot.DH.a);

%See geometry
L2=a(2);
L3=d(4);
A2 = a(3);
```

```
L4 = sqrt(A2^2 + L3^2);

%given q1 is known, compute first DH transformation
T01=dh(robot, q, 1);

%Express Pm in the reference system 1, for convenience
p1 = inv(T01)*[Pm; 1];

r = sqrt(p1(1)^2 + p1(2)^2);

beta = atan2(-p1(2), p1(1));
% gamma = (acos((L2^2+r^2-L3^2)/(2*r*L2)));
gamma = (acos((L2^2+r^2-L4^2)/(2*r*L2)));

if ~isreal(gamma)
    disp('WARNING:inversekinematic_irb140: the point is not reachable for this
configuration, imaginary solutions');
    %gamma = real(gamma);
end

%return two possible solutions
%elbow up and elbow down
%the order here is important and is coordinated with the function
%solve_for_theta3
q2(1) = pi/2 - beta - gamma; %elbow up
q2(2) = pi/2 - beta + gamma; %elbow down


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% solve for third joint theta3, two different
% solutions are returned, corresponding
% to elbow up and down solution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function q3 = solve_for_theta3(robot, q, Pm)

%Evaluate the parameters
d = eval(robot.DH.d);
a = eval(robot.DH.a);

%See geometry
L2=a(2);
L3=d(4);

A2 = a(3);

L4 = sqrt(A2^2 + L3^2);

phi = acos((A2^2+L4^2-L3^2)/(2*A2*L4));

%given q1 is known, compute first DH transformation
T01=dh(robot, q, 1);

%Express Pm in the reference system 1, for convenience
p1 = inv(T01)*[Pm; 1];

r = sqrt(p1(1)^2 + p1(2)^2);

% eta = (acos((L2^2 + L3^2 - r^2)/(2*L2*L3)));

eta = real(acos((L2^2 + L4^2 - r^2)/(2*L2*L4)));

if ~isreal(eta)
   disp('WARNING:inversekinematic_irb140: the point is not reachable for this
configuration, imaginary solutions');
   %eta = real(eta);
end

%return two possible solutions
```

```
%elbow up and elbow down solutions
%the order here is important
% q3(1) = pi/2 - eta;
% q3(2) = eta - 3*pi/2;

q3(1) = pi - phi - eta;
q3(2) = pi - phi + eta;
```

9.2.4. *Scripts* y funciones para el control del robot

9.2.4.1. robot_start.m

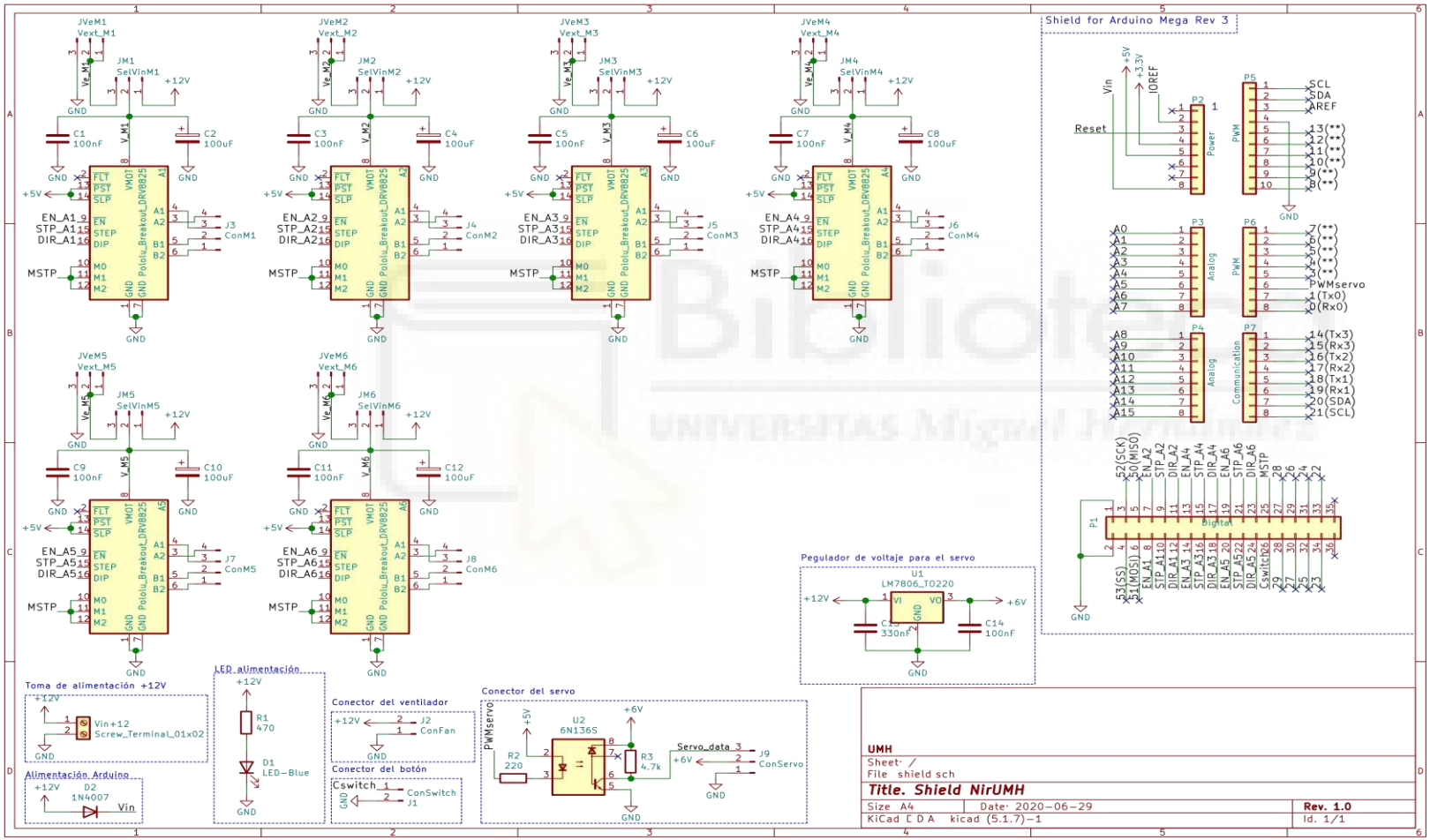
```
fopen(robot.serial);
msg = '';
while isequal(msg, 'robot_ready!') == false
    msg = fgets(robot.serial);
    msg = msg(1:end-2); %% This erases two last elements \r and \n
    if isequal(msg, 'robot_ready!') == true
        continue
    else
        disp(msg)
    end
end

disp("Robot ready!")
```

9.2.4.2. move_robot.m

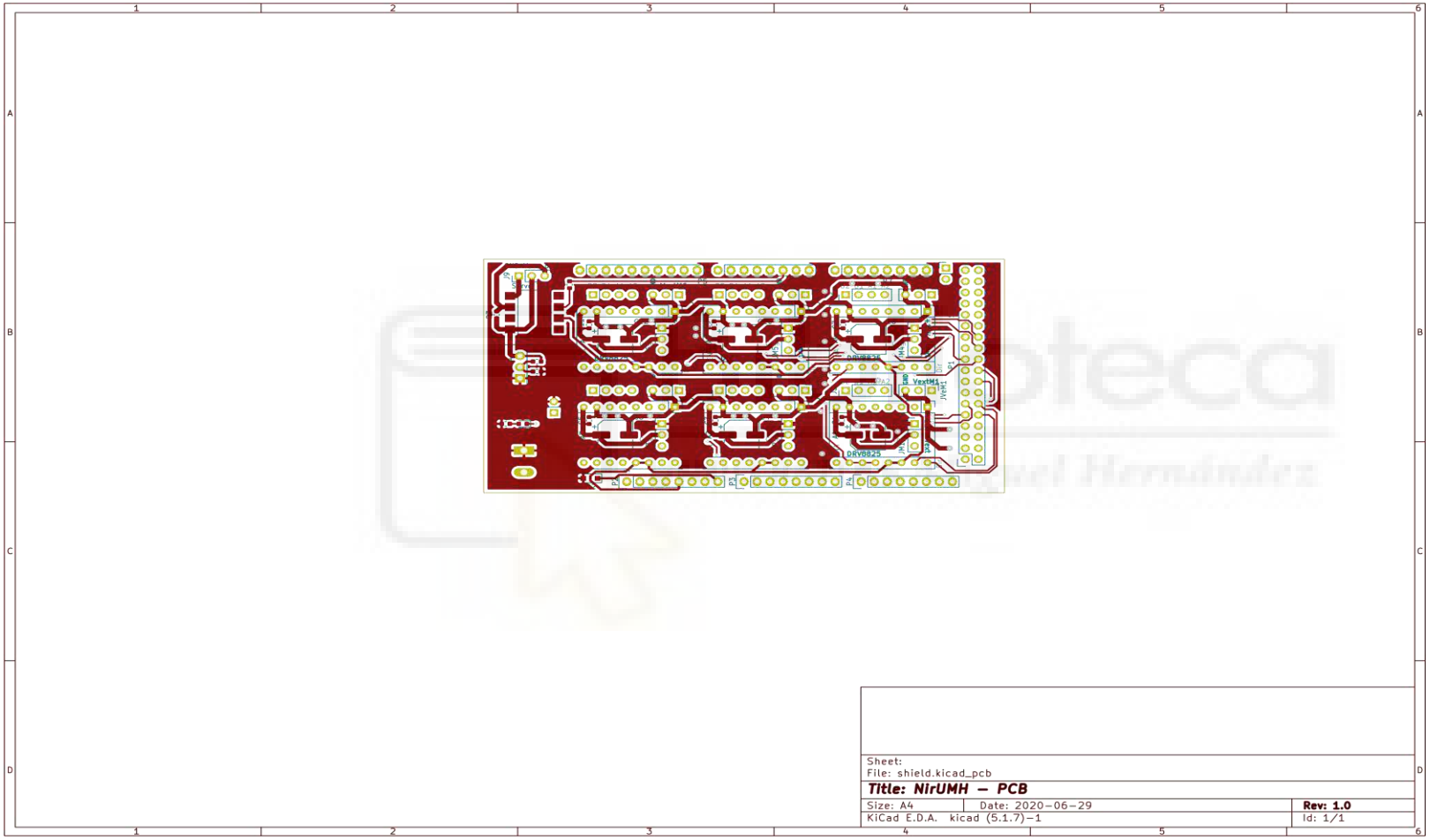
```
function move_robot(robot, q)
    if strcmp(q, 'HOME') == 1 || strcmp(q, 'home') == 1
        fprintf(robot.serial, 'HOME');
        drawrobot3d(robot, zeros(1, 6))
    elseif strcmp(q(1:4), 'TOOL') == 1 || strcmp(q(1:4), 'tool') == 1
        q = upper(q);
        fprintf(robot.serial, q);
    else
        q = rad2deg(q);
        msg = 'MOVQ ';
        for i=1:6
            msg = strcat(msg, num2str(q(i)));
            if i ~= 6
                msg = strcat(msg, '_');
            end
        end
        drawrobot3d(robot, deg2rad(q))
        fprintf(robot.serial, msg);
        msg = fgets(robot.serial);
        disp(msg)
    end
end
```

9.3. ESQUEMA ELECTRÓNICO Y DIAGRAMA DE LA PCB

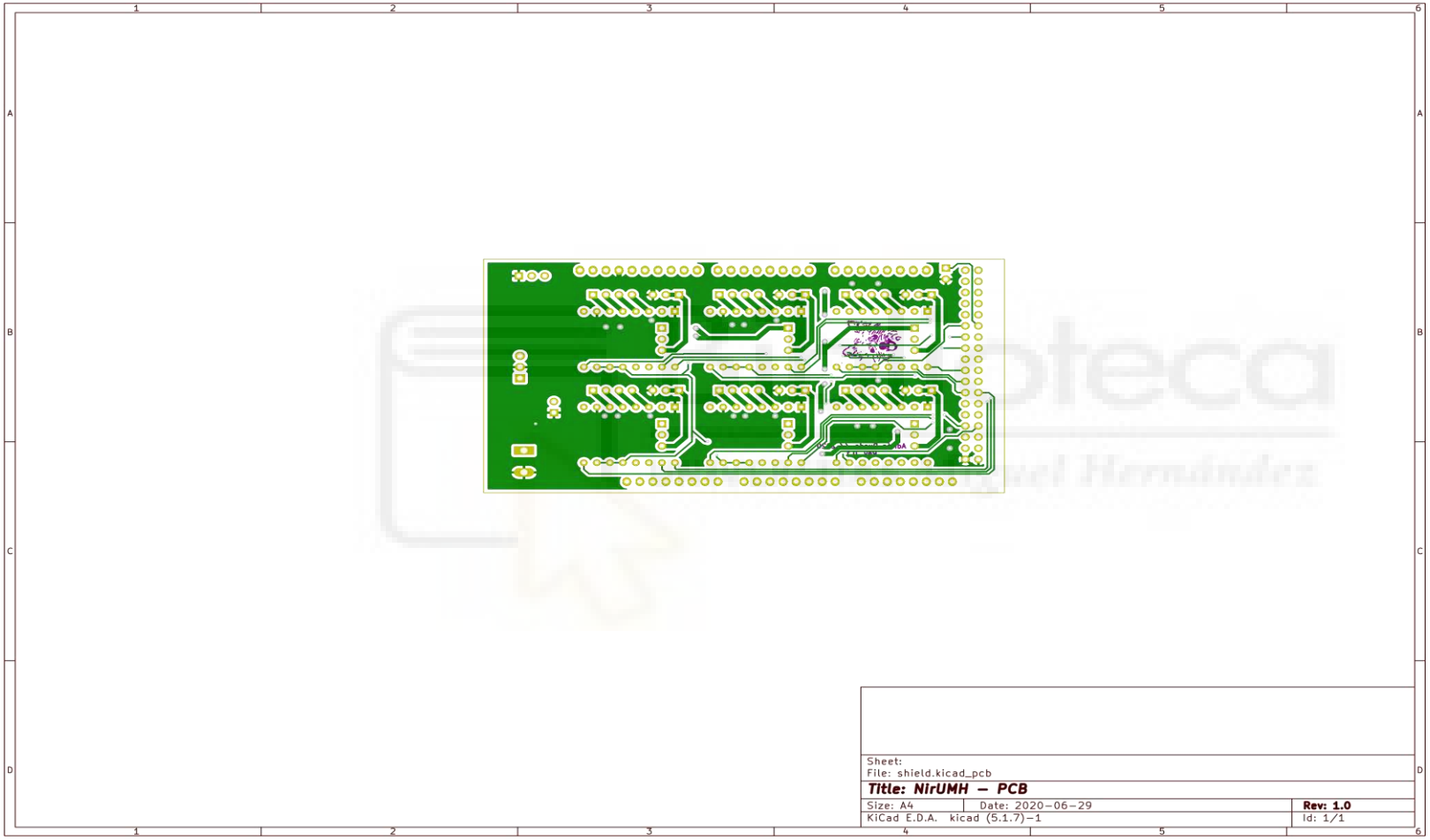
9.3.1. Esquema electrónico



9.3.2. Diagrama de pistas cara superior



9.3.3. Diagrama de pistas cara inferior



9.4. ENSAMBLAJE DEL ROBOT

Aunque el proceso de ensamblaje del robot, por lo general, no es complejo, sí que hay algunas piezas que requieren de una pequeña explicación para facilitar el ensamblaje.

Un primer paso común para el montaje de cualquier pieza del robot es haber roscado previamente los agujeros donde se colocan los tornillos con un macho de roscar adecuado (M3x0.5 – M4x0.7). Además, siempre que sea posible, se recomienda el uso de tornillos pavonados de cabeza Allen, en el caso de no utilizar un tornillo de estas características se indicará debidamente. También se recomienda, en general, la lubricación de todas las piezas móviles, excepto en las correas dentadas.

9.4.1. Base

El proceso de montaje de la base del robot se va a explicar de manera resumida, pues el fabricante del robot proporciona un vídeo para realizar el montaje de la base [18]. Para montar la base se necesitan los elementos de la Tabla 13 - Lista de elementos de la base.

ELEMENTO	OBSERVACIONES	CANTIDAD
Motor articulación q1	17HD48002H-22B	1
Rodamiento F8-19M	Axial	1
Rodamiento 608ZZ	Cualquier tipo de tapas - Radial	1
Rodamiento 623ZZ	Cualquier tipo de tapas - Radial	3
Botón	NA – Ø12.7 mm	1
Tornillo M8x60	Allen pavonado	1
Polea	16 dientes – GT2	1
Correa dentada cerrada	Ancho 6 mm – GT2 – 220 mm	1
Tornillo M3x10	Allen pavonado	10
Tornillo M3x15	Allen pavonado	3
Arandela M3	Cincada	10
Tornillo M3x20	Allen pavonado	2
Tuerca M3	Autoblocante	2
Tapa de ventilación		3
Piezas de la base		2
Soporte de rodamientos		1
Tensor de correa		1

Tabla 13 - Lista de elementos de la base

El montaje de la base del robot requiere de la unión de las dos piezas de la base con un adhesivo apto para plásticos, y una buena elección es la resina epoxi. La unión se puede apreciar en la Figura 60 - Vista inferior de la base del robot.

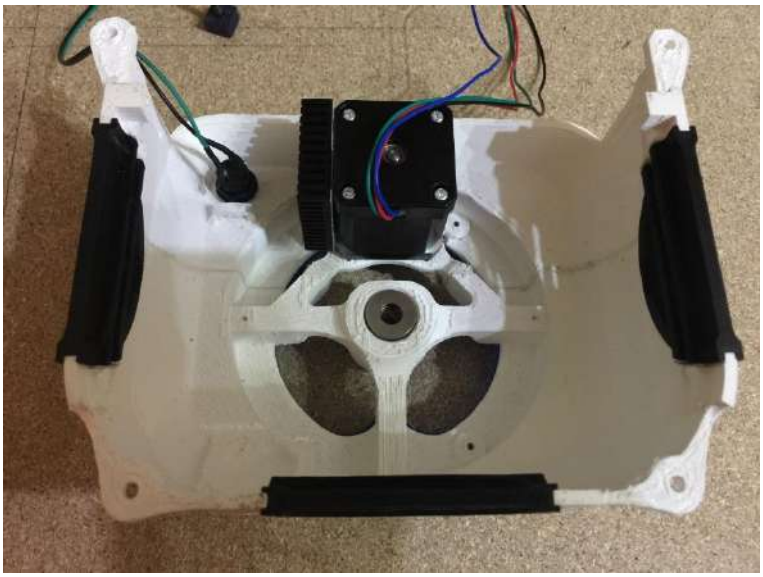


Figura 60 - Vista inferior de la base del robot

Además, se han de pegar las tapas de ventilación en los laterales de la base del robot, así como introducir el rodamiento F8-19M en el hueco central. En este paso se puede colocar, también, el botón, tal y como se puede ver en la esquina superior izquierda, tal y como se puede ver en la Figura 60 - Vista inferior de la base del robot.

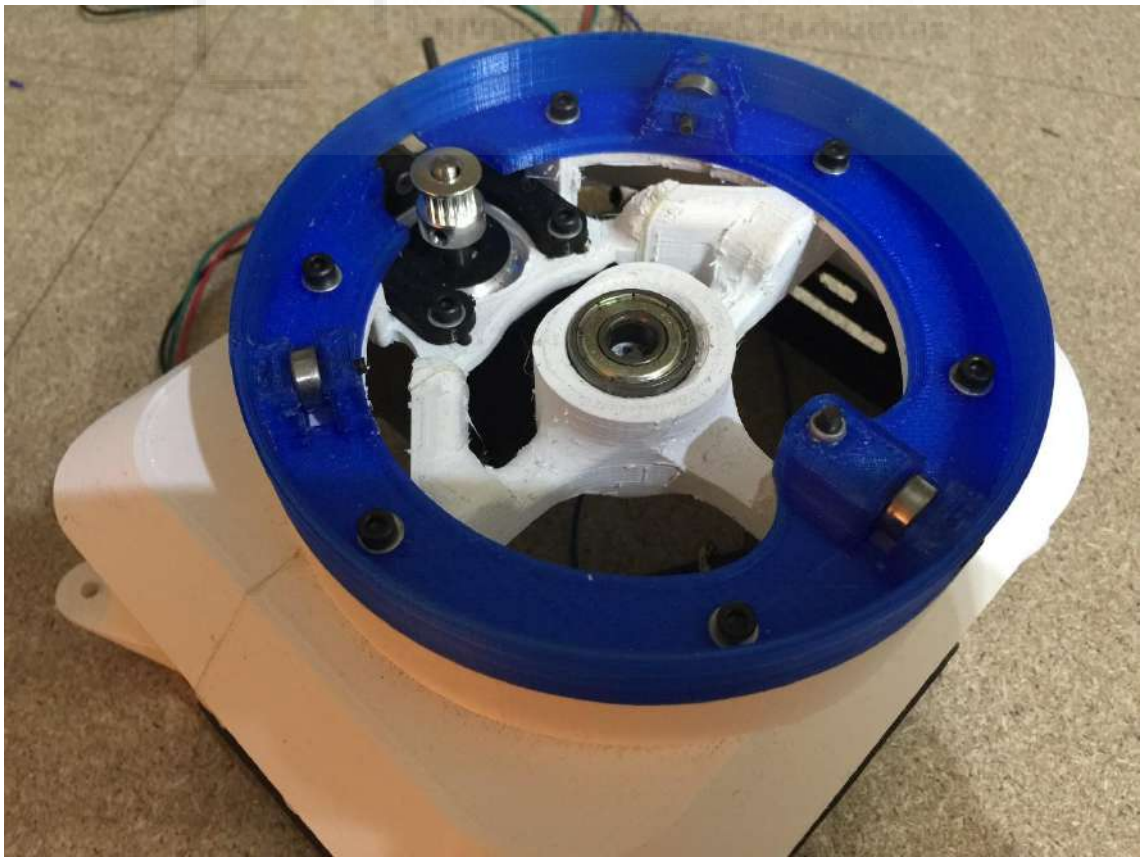


Figura 61 - Vista superior de la base del robot

El montaje de la parte superior se divide en dos partes: una primera parte donde se monta el motor, y una segunda donde se sujeta el soporte de los rodamientos.

Se ha de colocar, primero, la polea en el eje del motor dejando que el eje salga, aproximadamente entre 1 y 2 mm, tal y como se puede ver en la Figura 61 - Vista superior de la base del robot.

Para colocar el motor, se han de utilizar 4 tornillos M3x10 con las correspondientes arandelas. Previamente, se ha de insertar una tuerca M3 autoblocante en el orificio del tensor de correa. Esta tuerca se puede ver tras la polea del motor en la Figura 61 - Vista superior de la base del robot. Además, por la parte inferior del soporte de los rodamientos se ha de insertar un tornillo M3x15 que se sujeta con una tuerca M3 autoblocante. Este tornillo se sitúa en la parte opuesta del motor y se utiliza como tope de giro del hombro.

El montaje de los rodamientos 623ZZ se realiza con tornillos M3x15 y la sujeción del soporte de los rodamientos con la base del robot se realiza mediante 6 tornillos M3x10 con sus arandelas de M3 correspondientes.

En el apartado 9.4.2 - Hombro se completa el montaje de la base, la colocación de la correa que mueve el hombro y el ajuste de los rodamientos utilizados en el centro de la base del robot, y el uso del tornillo M8x60.

9.4.2. Hombro

Del mismo modo que la base, el fabricante cuenta con un vídeo donde se explica detalladamente el montaje de esta pieza [19]. Para montar la base se necesitan los elementos de la Tabla 14 - Lista de elementos del hombro.

ELEMENTO	OBSERVACIONES	CANTIDAD
Aro de rodamientos		1
Rueda de la base del hombro		
Soporte del hombro		1
Articulación del hombro		1
Tubo del hombro		1
Adaptadores de eje		2
Arandelas de eje	3 mm y 1 mm	2
Embellecedores		2
Tornillo M3x10	Allen pavonado	14
Tornillo M3x15	Allen pavonado	1
Tornillo M4x55	Allen pavonado	1
Tornillo M8x90	Allen pavonado	1
Tuerca M4	Autoblocante	1
Tuerca M8	Autoblocante	2
Rodamiento MR105	Cualquier tipo de tapas - Radial	2
Rodamiento 608ZZ	Cualquier tipo de tapas	2
Correa dentada	Ancho 6 mm – GT2 – 405 mm	1

Tabla 14 - Lista de elementos del hombro

El primer paso es colocar los dos rodamientos MR105 con un tornillo M3x15 en el lateral del hombro y la tuerca M8 autoblocante tal y como se puede ver en la Figura 62 - Montaje de los rodamientos MR105 y tuerca M8 de la base del hombro.

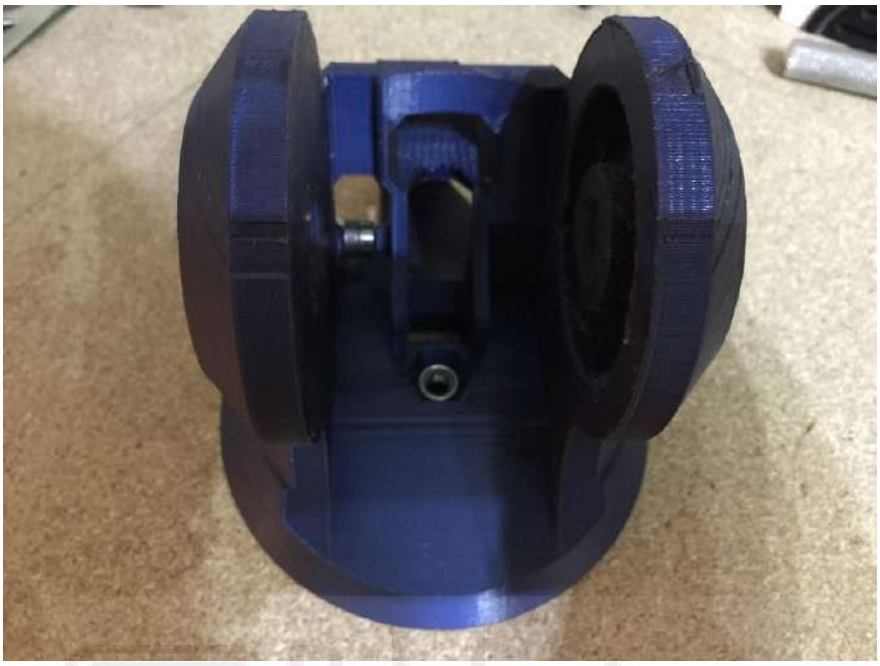


Figura 62 - Montaje de los rodamientos MR105 y tuerca M8 de la base del hombro

A continuación, se ha de montar la rueda de la base del hombro con 4 tornillos M3x10. Tal y como se puede ver en la Figura 63 - Montaje de la rueda de la base del hombro, la rueda de la base del hombro debe quedar con el agujero rectangular hacia arriba, coincidiendo con el hueco de la izquierda para pasar el cable de motor de la articulación q2, y con el tope de giro del hombro.

Figura 63 - Montaje de la rueda de la base del hombro

Para continuar con el montaje del hombro, se ha de montar, a continuación, la correa en la articulación del hombro, tal y como se puede ver en la Figura 64 - Montaje de la correa del hombro. La correa queda bien fijada con solo introducirla en su hueco, no es necesario añadir ningún tipo de adhesivo.

Figura 64 - Montaje de la correa del hombro

El siguiente paso es montar la articulación del hombro en la base del hombro. Para ello, en primer lugar, hay que premontar los adaptadores de eje con las arandelas de eje tal y como se muestra en la Figura 65 - Premontaje de los adaptadores de eje.

Figura 65 - Premontaje de los adaptadores de eje

Como se puede ver en la Figura 65 - Premontaje de los adaptadores de eje, las arandelas tienen distinto espesor (3 mm y 1 mm). La de 1 mm se ha de colocar en la cara plana interna, mientras que la de 3 mm se coloca en la cara opuesta. Estas arandelas son necesarias para centrar el hombro y alinear la correa. Esto es debido a que el diseño original cuenta con un muelle de torsión, pero en este trabajo se ha omitido su uso. Una vez colocados los adaptadores de eje, se ha de sujetar la articulación del hombro de manera que al presionar ambos adaptadores sujeten la articulación.

Tras haber sujetado la articulación del hombro, se ha de colocar un rodamiento 608ZZ a cada lado y se ha de sujetar con el tornillo M8x90 y una tuerca M8 autoblocante, Figura 66 - Montaje de la articulación del hombro. Se recomienda utilizar algún tipo de lubricante en el hueco del tornillo.

Figura 66 - Montaje de la articulación del hombro

Es muy importante no apretar en exceso la tuerca, pues los soportes de la articulación se cierran y frenan el movimiento de la articulación, además de que pueden llegar a partirse. Se ha de apretar lo suficiente como para que gire sin resistencia. Una vez apretado, se ha de colocar cada embellecedor con 3 tornillos M3x10.

El siguiente paso es unir el hombro con la base del robot, lo cual puede acarrear cierta dificultad a la hora de colocar la correa dentada en la polea del motor q1 y la rueda de la base del robot. Primero se ha de colocar la correa pasando por la polea del motor q1 y colocar el aro de rodamientos tal y como se puede ver en la Figura 67 - Colocación de la correa dentada y el aro de rodamientos.

Figura 67 - Colocación de la correa dentada y el aro de rodamientos

Una vez colocado se ha de poner el hombro y con el tornillo M8x60 atornillar la base al hombro. El apriete ha de ser el justo para que el movimiento del hombro sea suave. Tras atornillar el hombro se ha de colocar en su sitio la correa dentada. En la Figura 68 - Detalle de la correa de la base y el hombro se puede ver la correa colocada en su sitio y el tornillo debidamente apretado.

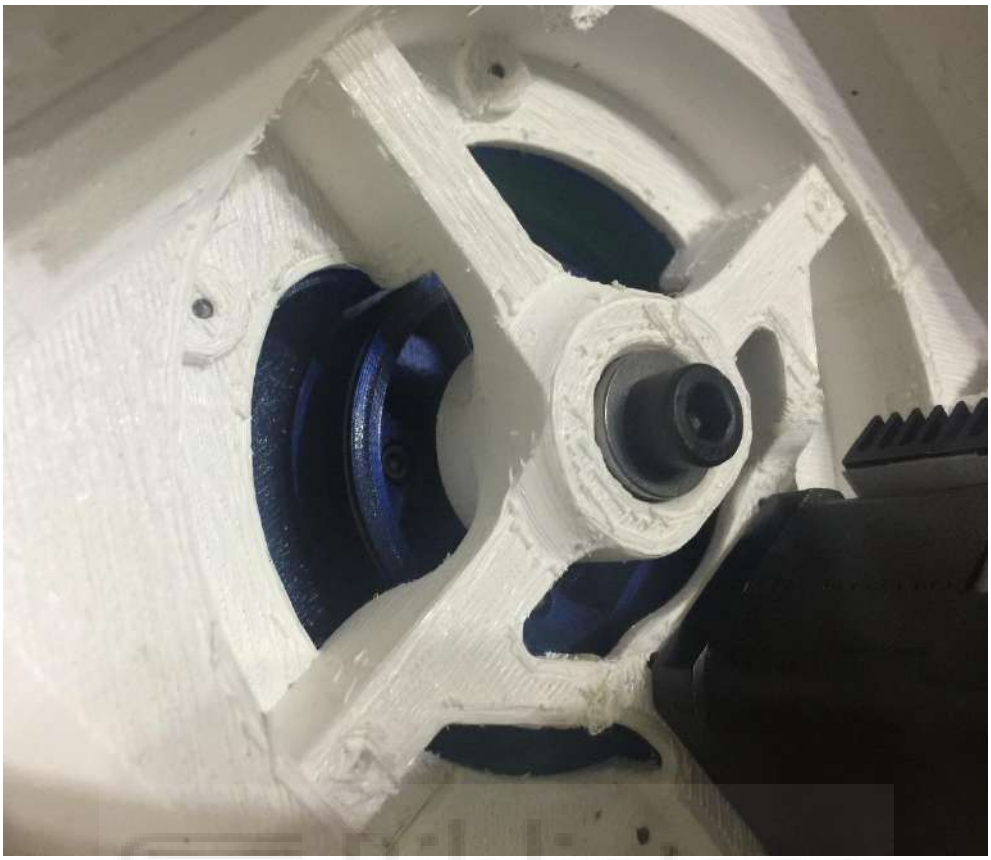


Figura 68 - Detalle de la correa de la base y el hombro

Finalmente, se ha de colocar el tubo del hombro con el tornillo M4x55 junto a una tuerca M4 autoblocante, Figura 69 - Montaje final del hombro.

Figura 69 - Montaje final del hombro

9.4.2.1. Caja reductora

Aunque este elemento se ha descartado del diseño final, se ha mantenido el ensamblaje ya que forma parte de las pruebas realizadas durante el desarrollo de este trabajo.

Para la caja reductora se necesitan los elementos de la Tabla 15 - Lista de elementos de la caja reductora.

ELEMENTO	OBSERVACIONES	CANTIDAD
Caja de la reductora		1
Tapa de la reductora		1
Espaciadores de engranajes	Espesores de 1 mm y 2 mm	2
Piñón de 10 dientes		1
Rueda de 17 dientes		1
Eje del piñón	Varilla lisa de Ø5 mm y 48 mm	1
Eje de la rueda	Varilla lisa de Ø5 mm y 114 mm	1
Piñón de 16 dientes	Taladro de Ø5 mm	1
Tornillo M3x16	Allen pavonado	16
Tuercas M3		4
Rodamiento MR105	Cualquier tipo de tapas - Radial	5
Lubricante o grasa	Indicado para plásticos	----
Resina epoxi		----
Motor articulación q2	17HS6401S	1
Acople para motor	Taladros de Ø5 mm	1
Tensor de correa		1
Tuerca M3	Autoblocante	1

Tabla 15 - Lista de elementos de la caja reductora

El primer paso para montar la caja reductora es eliminar cualquier imperfección que haya podido quedar tras el proceso de impresión en cualquier superficie plana de contacto entre partes fijas o móviles (contacto entre motor y tapa, entre tapa y caja o entre espaciadores y engranajes). El siguiente paso es insertar los rodamientos en sus orificios como se puede ver en la Figura 70 - Montaje de los rodamientos.

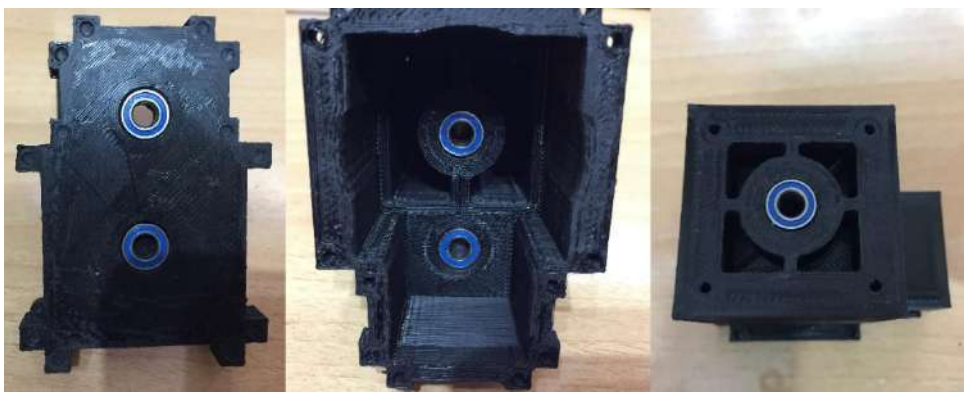


Figura 70 - Montaje de los rodamientos

Una vez colocados los rodamientos, se ha de ajustar la posición de los engranajes. Para ello se han de colocar los engranajes con los espaciadores como se ve en la Figura 71 - Preparación de los ejes de los engranajes.

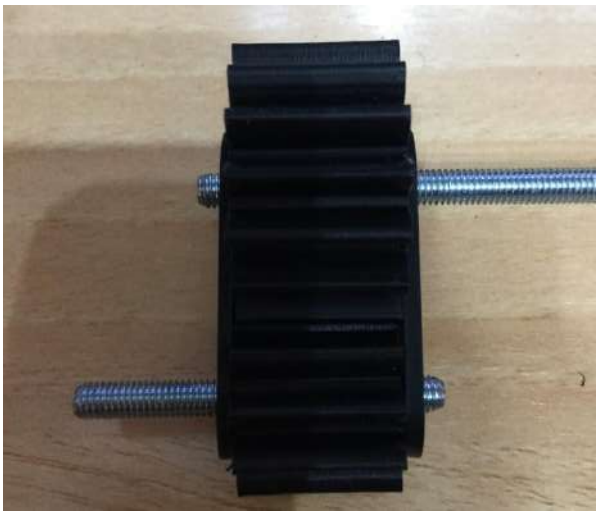


Figura 71 - Preparación de los ejes de los engranajes

Una vez colocados, se ha de dejar tanto para el eje de 48 mm y 114 mm, que sobresalga entre 3 y 4 mm. El siguiente paso se ha de hacer extremando el cuidado, pues hay que retirar los espaciadores, marcar el eje, cubrirlo con resina epoxi y roscar el engranaje hasta alcanzar la marca realizada. Una vez endurecida la resina se vuelven a colocar los espaciadores y se introduce el conjunto de ambos engranajes en la caja de la reductora, Figura 72 - Engranajes montados. Además, se ha de aplanar el eje del piñón para una mejor sujeción del acople del motor.

En caso de ser necesario se ha de eliminar el sobrante de la resina tras montar los engranajes con los ejes.

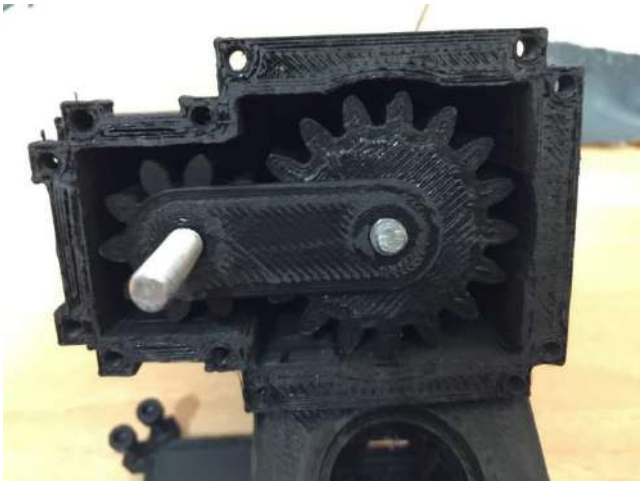


Figura 72 - Engranajes montados

Para sujetar el motor se han empleado tornillos M3x16 (Figura 73 - Detalle de los tornillos del motor de la reductora). No obstante, estos no son adecuados para sujetar el motor por su longitud, por lo que se han utilizado tuercas para compensar el exceso de longitud. Cabe destacar que, si se disponen de tornillos de menor longitud, se recomienda encarecidamente su uso.

Figura 73 - Detalle de los tornillos del motor de la reductora

Finalmente, la reductora se ha de sujetar con 4 tornillos M3x16 al hombro intercalando el tensor de correa al que previamente se le ha colocado una tuerca M3 autoblocante (igual que en el tensor de correa de la base), Figura 74 - Montaje de la caja reductora en el robot.

Figura 74 - Montaje de la caja reductora en el robot

Antes de colocar la polea, se ha de aplanar la varilla roscada para que la sujeción sea mejor, Figura 75 - Medición para aplanar el eje de la reductora.

Figura 75 - Medición para aplanar el eje de la reductora

9.4.2.2. Motor con caja reductora

El ensamblaje del motor con caja reductora requiere de los elementos de la Tabla 16 - Lista de elementos del motor con caja reductora.

ELEMENTO	OBSERVACIONES	CANTIDAD
Soporte del motor		1
Tensor de correa		1
Tornillo M3x10	Allen pavonado	4
Tornillo M3x16	Allen pavonado	1
Tornillo M3x20	Allen pavonado	4
Arandela M3	Cincada	4
Tuerca M3	Autoblocante	5
Piñón 20 dientes	Taladro de Ø8 mm	1
Motor articulación q2	17HS6401S-PG3.71	1

Tabla 16 - Lista de elementos del motor con caja reductora

El primer paso es insertar a presión las tuercas M3 autoblocantes en los orificios correspondientes, como se puede ver en la Figura 76 - Soporte del motor con reductora.

Figura 76 - Soporte del motor con reductora

A continuación, se ha de colocar la pieza del soporte del motor en la caja reductora el motor. La sujeción se realiza con 4 tornillos M3x10, tal y como se puede ver en la Figura 77 - Montaje del soporte en el motor.

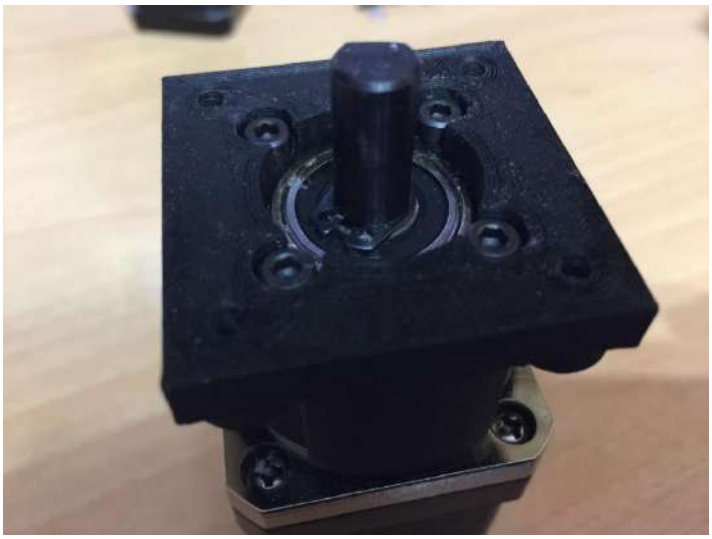


Figura 77 - Montaje del soporte en el motor

El siguiente paso es colocar el motor en el robot. Para ello, se ha de colocar el motor alineado con la correa y, éste, se debe sujetar con el tensor de correa mediante 4 tornillos M3x20, tal y como se puede ver en la Figura 78 - Motor sujeto junto al tensor de la correa del hombro.

Figura 78 - Motor sujeto junto al tensor de la correa del hombro

Finalmente, se ha de colocar la polea de 20 dientes, Figura 79 - Colocación de la polea de 20 dientes, y se ha de sujetar con los dos tornillos prisioneros. Uno de los tornillos ha de coincidir con la cara plana del eje. Una vez sujeto, se ha de tensar la correa de manera que quede tirante y no se aprecie que saltan los dientes, Figura 80 - Correa del hombro debidamente tensada. Para el tensor de correa, se utiliza un tornillo M3x16 junto a una tuerca M3 autoblocante.

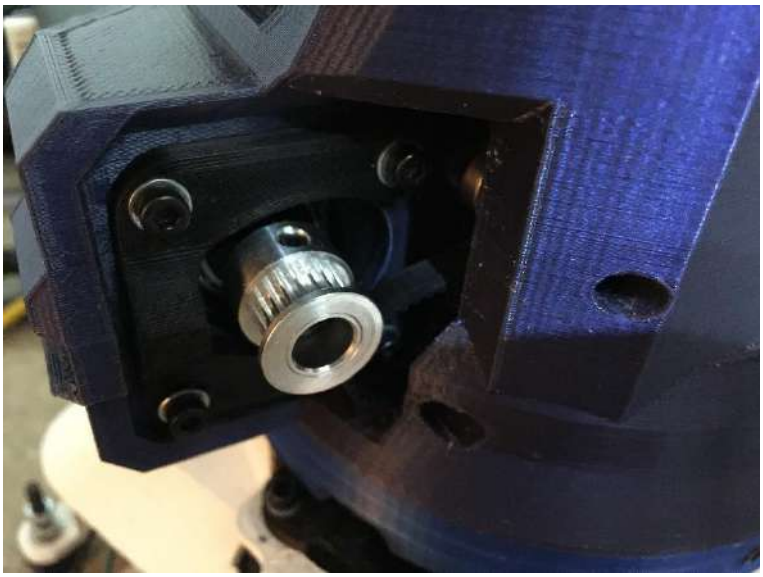


Figura 79 - Colocación de la polea de 20 dientes

Figura 80 - Correa del hombro debidamente tensada

9.4.3. Codo

El ensamblaje del codo requiere de los elementos de la Tabla 17 - Lista de elementos del codo.

ELEMENTO	OBSERVACIONES	CANTIDAD
Soporte del codo		1
Articulación del codo		1
Tensor de correa		1
Casquillo de ajuste		2
Arandela de ajuste		2
Embellecedor		2
Tapa de rodamiento		1
Tornillo M3x10	Allen pavonado	12
Tornillo M3x16	Allen pavonado	7
Tornillo M4x50	Allen pavonado	1
Tornillo M8x100	Allen pavonado	1
Tuerca M3	Autoblocante	1
Tuerca M4	Autoblocante	1
Tuerca M8	Autoblocante	1
Rodamiento 608ZZ	Cualquier tipo de tapa	2
Rodamiento 623ZZ	Cualquier tipo de tapa	2
Rodamiento 61807-2RS	Cualquier tipo de tapa	1
Acople flexible	Taladros de Ø5 mm y Ø8 mm	1
Piñón 16 dientes	Taladro de Ø5 mm	1
Correa dentada	Ancho 6 mm – GT2 – 325 mm	1
Motor articulación q3	17HS15-1504S-X1	1
Motor articulación q4	14HS17-0504S	1

Tabla 17 - Lista de elementos del codo

Para el montar el codo se han de preparar una serie de elementos previamente. En primer lugar, se ha de montar el tensor de correa, para ello, se ha de introducir una tuerca M3 autoblocante, y colocar los cuatro tornillos M3x10 tal y como se ve en la Figura 81 - Preparación del tensor de correa del codo.

Figura 81 - Preparación del tensor de correa del codo

El siguiente elemento que se debe preparar es el motor de la articulación q3. Se ha de colocar el piñón de 16 dientes enrasado con el extremo del eje del motor, tal y como se muestra en la Figura 82 - Montaje del piñón de la articulación q3.

Figura 82 - Montaje del piñón de la articulación q3

En el motor de la articulación q4 se ha de colocar el acople flexible a una distancia, aproximada de 7 mm entre el saliente del chasis del motor hasta la base del acople, Figura 83 - Colocación del acople flexible en el motor q4.

Figura 83 - Colocación del acople flexible en el motor q4

Los últimos elementos que se deben preparar son los casquillos de ajuste y las arandelas de ajuste tal y como se aprecia en la Figura 84 - Preparación de los casquillos y arandelas de ajuste del codo.

Figura 84 - Preparación de los casquillos y arandelas de ajuste del codo

Una vez preparados los elementos anteriores, se han de colocar los rodamientos 623ZZ con un tornillo M3x16, Figura 85 - Montaje de rodamientos 623ZZ del codo, y la tuerca M4 autoblocante, Figura 86 - Inserción de la tuerca autoblocante del codo.

Figura 85 - Montaje de rodamientos 623ZZ del codo

Figura 86 - Inserción de la tuerca autoblocante del codo

El siguiente paso es montar la correa del codo, el motor de la articulación q4 y el rodamiento 61807-2RS y la tapa del rodamiento con 4 tornillos M3x10, tal y como se muestra en la Figura 87 - Montaje del motor de la articulación q4 y la correa del codo, y la Figura 88 - Montaje del rodamiento 61807-2RS y tapa del rodamiento. El motor se ha de sujetar con 4 tornillos M3x10.

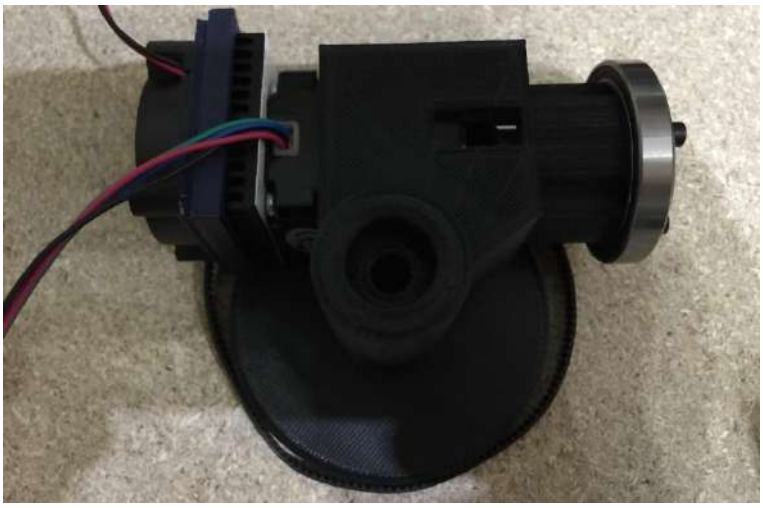


Figura 87 - Montaje del motor de la articulación q4 y la correa del codo

Figura 88 - Montaje del rodamiento 61807-2RS y tapa del rodamiento

A continuación, se ha de montar el motor de la articulación q3 de manera provisional. Se ha de sujetar firmemente con el tensor de correa del mismo modo que en la Figura 75 - Medición para aplanar el eje de la reductora, donde se puede ver a la izquierda el tensor de la correa sujetando la caja reductora.

Una vez sujeto el motor, se ha de colocar la articulación del codo en el soporte del codo. Para ello, se han de realizar los mismos pasos que en 9.4.2 - Hombro. A continuación, con el tornillo M8x100, la tuerca M8 autoblocante y los dos rodamientos 608ZZ, se han de fijar las dos piezas de la articulación (Figura 89 - Fijación de la articulación del codo).

Figura 89 - Fijación de la articulación del codo

El siguiente paso, colocar la correa de la articulación en el piñón del motor, puede presentar cierta dificultad. Para facilitar este paso, se ha de aflojar el motor de la articulación q3 de manera que quede lo más libre posible y orientarlo de manera que facilite la inserción de la correa. Además, se ha de girar la articulación para ayudar a la correa a entrar en su lugar. Una vez colocada la correa se ha de volver a apretar firmemente el motor haciendo que quede la correa tensa. Adicionalmente, por la parte inferior de la base del codo, hay un orificio para un tornillo M3x16 necesario para ayudar a tensar la correa. Finalmente, se han de colocar los embellecedores con 3 tornillos M3x16 cada uno (Figura 90 - Montaje final del codo).

Figura 90 - Montaje final del codo

9.4.4. Antebrazo

El ensamblaje del antebrazo es uno de los más sencillos de todo el robot. Para ello se necesitan los elementos de la Tabla 18 - Lista de elementos del antebrazo.

ELEMENTO	OBSERVACIONES	CANTIDAD
Tubo del antebrazo		1
Pétalos		3
Tornillo M3x10	Allen pavonado	12
Tornillo M8x40	Cabeza hexagonal cincado	1
Tuerca M8	Autoblocante	1
Arandela M8	Cincada	1

Tabla 18 - Lista de elementos del antebrazo

El primer paso es preparar el tornillo M8x40 aplanando el extremo roscado con una lima para que pueda sujetarse mejor con el acople flexible del motor. Una vez preparado se introduce en el tubo y se fija a él con la arandela M8 y la tuerca M8 autoblocante (Figura 91 - Detalle del vástago del antebrazo).

Figura 91 - Detalle del vástago del antebrazo

El siguiente paso es colocar el tubo del antebrazo en la articulación del codo. No obstante, para facilitar el montaje, se recomienda colocar, previamente, los tornillos M3x10 en los pétalos, tal y como se puede ver en la Figura 92 - Pétalo con tornillos.

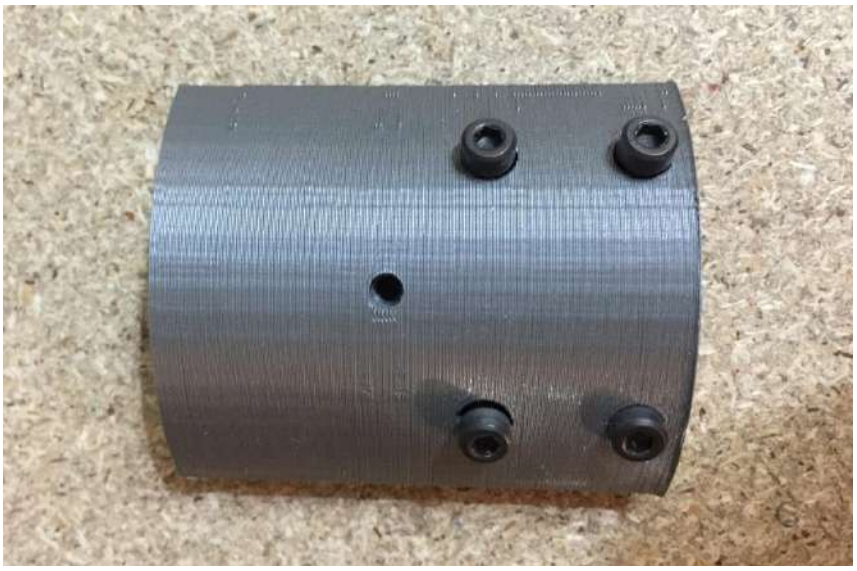


Figura 92 - Pétalo con tornillos

Para fijar el antebrazo, se recomienda colocar uno de los pétalos, introducir el vástago del antebrazo en el orificio del acople flexible del motor y apretar los tornillos del pétalo. El tubo del antebrazo quedará fijado, y en este momento se han de apretar los dos tornillos del acople flexible (Figura 93 - Colocación del primer pétalo).

Figura 93 - Colocación del primer pétalo

Es muy recomendable apretar los tornillos en cruz para que el contacto con el interior de los pétalos y el tubo del antebrazo sea mayor. Finalmente, se han de colocar los dos pétalos restantes para que el antebrazo quede completamente fijado (Figura 94 - Montaje final del antebrazo).

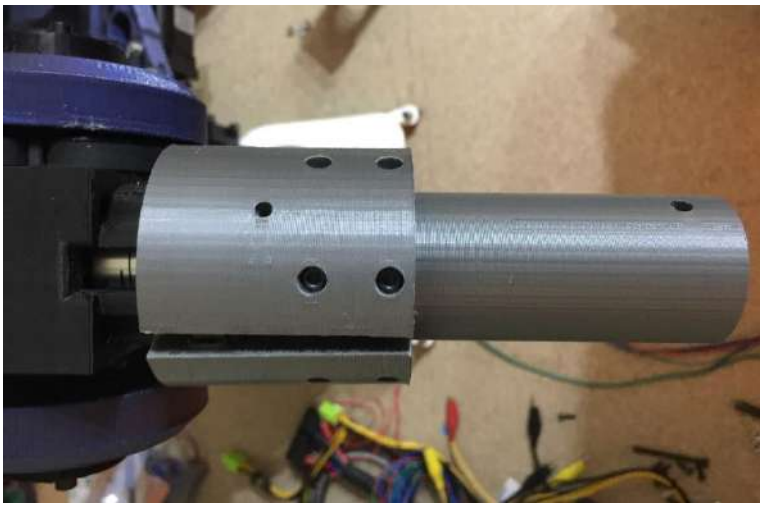


Figura 94 - Montaje final del antebrazo

9.4.5. Muñeca

Para la muñeca se necesitan los elementos de la Tabla 19 - Lista de elementos de la muñeca.

ELEMENTO	OBSERVACIONES	CANTIDAD
Bastidor de la muñeca		1
Caja de la muñeca		1
Tapa de la muñeca		1
Rejilla de protección del ventilador		1
Ventilador 12 V - 40x40x10 cm	Introduce aire	1
Tornillo M3x10	Allen pavonado	12
Tornillo M3x16	Allen pavonado	4
Tornillo M4x50	Allen pavonado	1
Tornillo M8x20	Cabeza hexagonal	1
Tuerca M3	Cincada	8
Tuerca M4 autoblocante	Cincada	1
Tuerca M8 autoblocante	Cincada	1
Arandela M3	Ala normal	8
Arandela M8	Ala normal	1
Rodamiento 608	Cualquier tipo de tapas - Radial	1
Motor articulación q5	14HS10-0404S	1
Motor articulación q6	14HS10-0404S	1

Tabla 19 - Lista de elementos de la muñeca

La muñeca del robot es el eslabón más complejo de montar, pues hay que hacerlo en varias partes. Además, hay que diferenciar dos elementos básicos de la muñeca: la caja de la muñeca, donde se montan los motores; y el bastidor de la muñeca, donde va sujeta la caja de la muñeca.

En el primer paso se ha de preparar el eslabón (o bastidor) de la muñeca introduciendo un rodamiento 608 (es indiferente el tipo de tapas que se utilicen en el rodamiento) a presión (Figura 95 - Rodamiento de la muñeca insertado).

Figura 95 - Rodamiento de la muñeca insertado

La inserción del rodamiento se puede realizar con un tornillo de banco o con un tornillo de métrica 8 junto a dos arandelas de ala ancha y una tuerca de métrica 8 como se puede ver en la Figura 96 - Sugerencia del proceso de inserción del rodamiento.

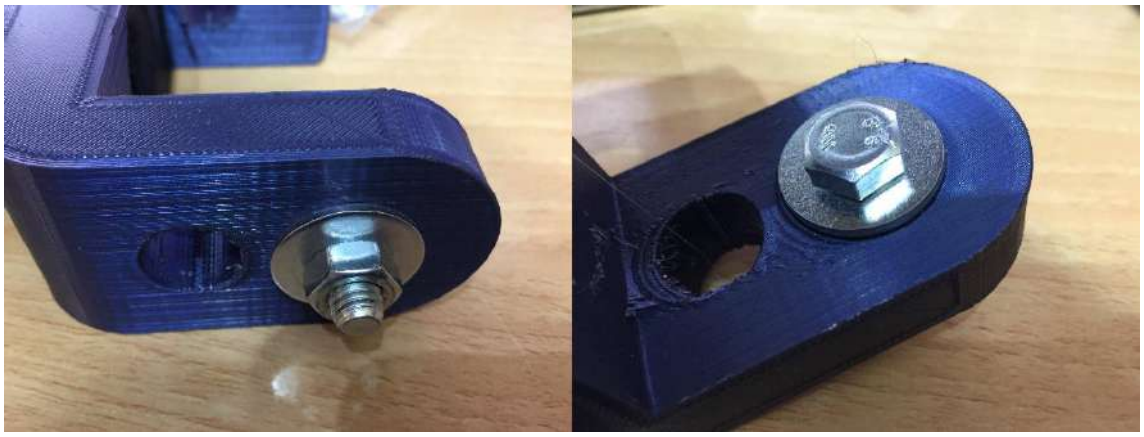


Figura 96 - Sugerencia del proceso de inserción del rodamiento

El siguiente paso es preparar la muñeca en sí. Lo primero que se ha de hacer es sujetar el ventilador junto a la rejilla protectora en la tapa de la caja de la muñeca con cuatro tornillos M3x16 (Figura 97 - Montaje del ventilador de la muñeca).

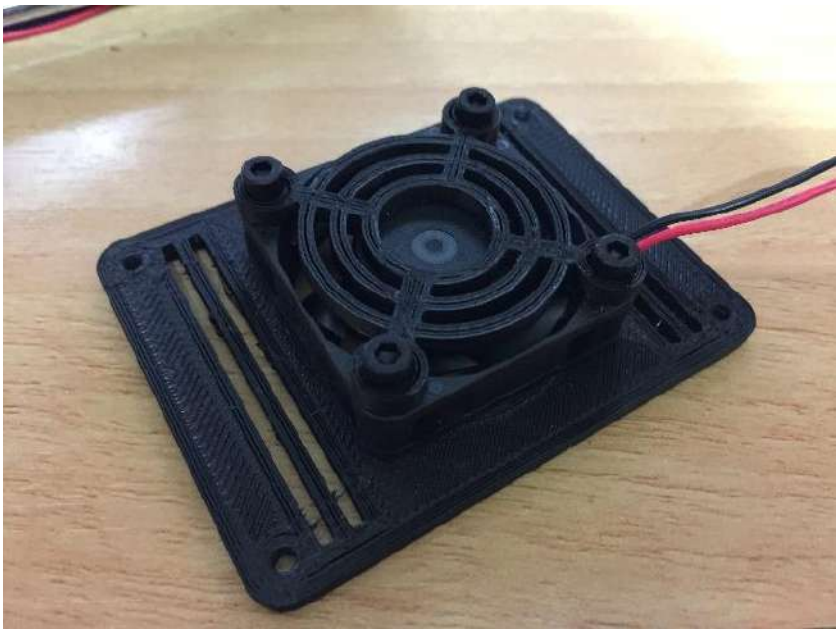


Figura 97 - Montaje del ventilador de la muñeca

Los cables del ventilador se han de pasar por los huecos de ventilación que hay junto al ventilador. Posteriormente, se debe pasar el cable por el hueco trasero de la caja de la muñeca preparado para sacar los cables de los motores y de la herramienta.

A continuación, se ha de introducir a presión un tornillo de cabeza hexagonal M8x20 en el hueco adecuado, pero no es necesario roscar previamente el agujero. Para facilitar la introducción del tornillo, se recomienda el uso de una tuerca para que estire de él. Una vez introducido y se tenga el hueco de la cabeza del tornillo listo se ha de extraer para el posterior montaje en el bastidor (Figura 98 - Inserción del tornillo-eje de la caja de la muñeca).

Figura 98 - Inserción del tornillo-eje de la caja de la muñeca

Una vez extraído el tornillo del paso anterior, se ha de montar el motor (14HS10-0404S) de la articulación q6 (Figura 99 - Montaje del motor de la articulación q6). Para ello, se recomienda el uso de tornillos M3x10 junto a tuercas M3 para acortar su longitud y arandelas de métrica 3 para mejorar el contacto con la superficie de la caja de la muñeca con la cabeza del tornillo. Si se dispone de tornillos más cortos, también se pueden utilizar siempre y cuando la cabeza del tornillo tenga contacto con el plástico de la caja de la muñeca.

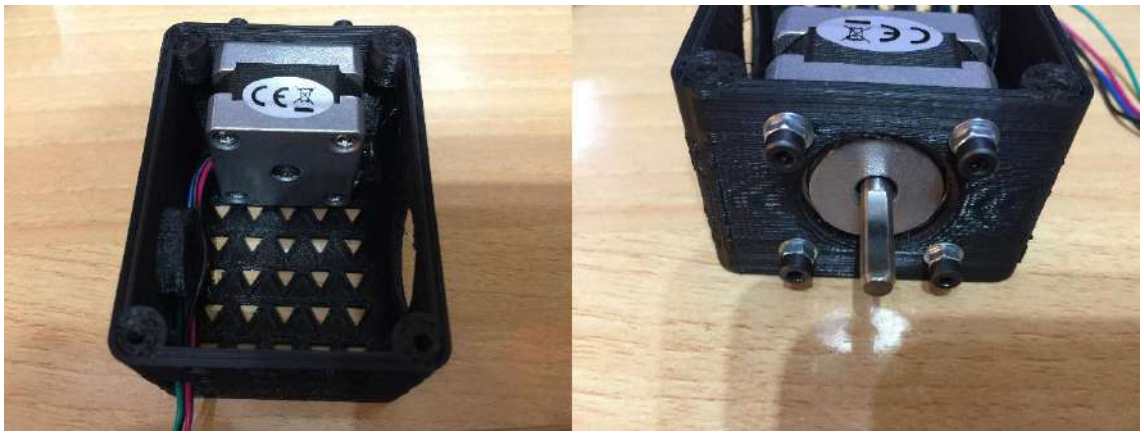


Figura 99 - Montaje del motor de la articulación q6

Con el motor de la articulación q5 (14HS10-0404S) se ha de introducir el eje en el agujero del bastidor de la muñeca, como se puede ver en la Figura 100 - Premontaje del motor de la articulación q5. Se recomienda hacerlo antes de montar el motor en la caja de la muñeca, de este modo el montaje final tendrá menor dificultad.

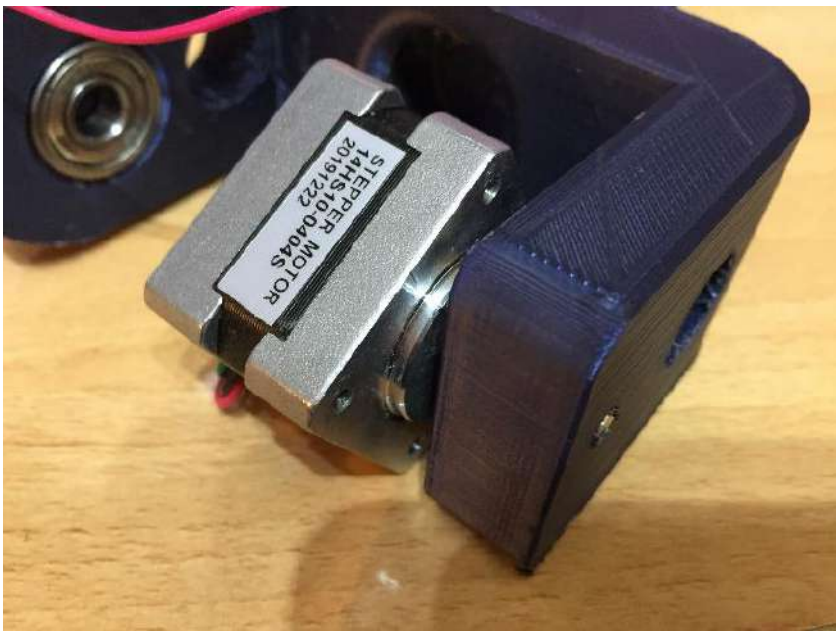


Figura 100 - Premontaje del motor de la articulación q5

Una vez extraído el motor de la articulación q5, se ha de insertar el tornillo M8x20 de cabeza hexagonal en la caja de la muñeca. Además, en la parte externa se ha de colocar una arandela cincada de M8 y la tuerca autoblocante de M8. Se ha de apretar la tuerca hasta llegar al final del recorrido permitido por el tornillo (Figura 101 - Montaje del tornillo-eje de la muñeca).

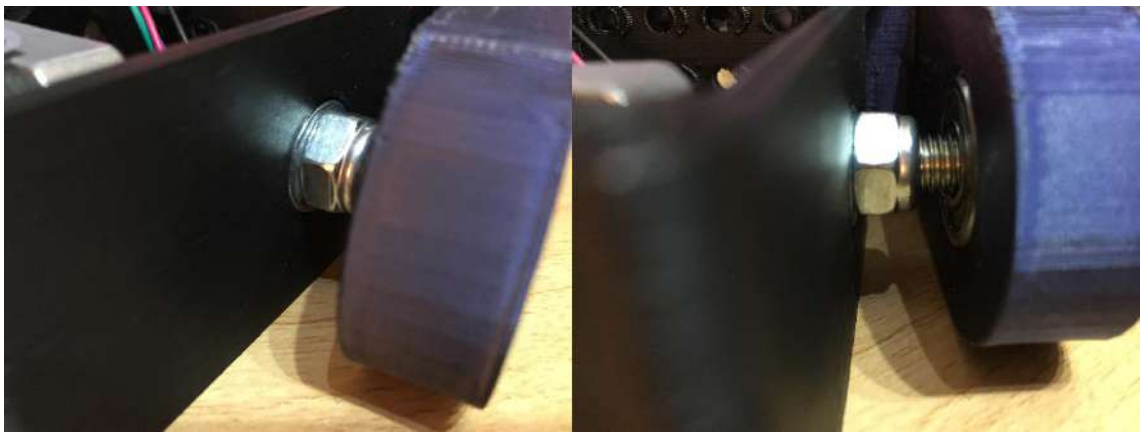


Figura 101 - Montaje del tornillo-eje de la muñeca

El siguiente paso es colocar el motor de la articulación q5. Para ello tan solo hay que introducir a presión el eje del motor en el hueco del bastidor de la muñeca, y atornillar el motor del mismo modo que el motor de la articulación q6 (Figura 102 - Montaje del motor de la articulación q5).

Figura 102 - Montaje del motor de la articulación q5

El bastidor de la muñeca cuenta con un agujero para insertar un tornillo de M3x10 en el caso de que el hueco para el eje del motor de la articulación q5 quede holgado. Se puede ver en la Figura 102 - Montaje del motor de la articulación q5.

Finalmente, se ha de colocar la tapa de la caja de la muñeca con cuatro tornillos de M3x10 y pasar el cable del ventilador por el agujero trasero, Figura 103 - Montaje final de la muñeca y agujero para pasar los cables.

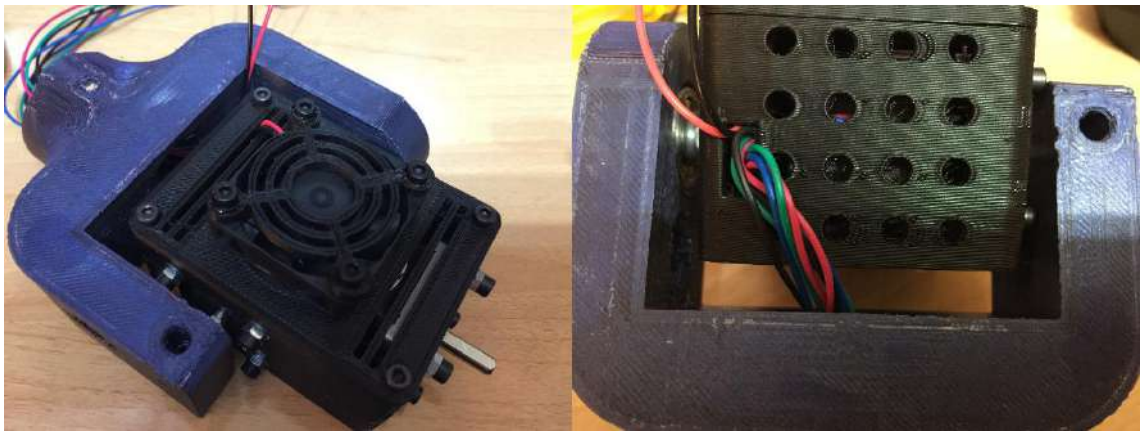


Figura 103 - Montaje final de la muñeca y agujero para pasar los cables

Adicionalmente, se ha de colocar una tuerca autoblocante M4 en la parte trasera del bastidor de la muñeca (Figura 104 - Tuerca de sujeción de la muñeca). Se ha de introducir a presión y puede que sea necesario ajustar el hueco de la tuerca para que entre.

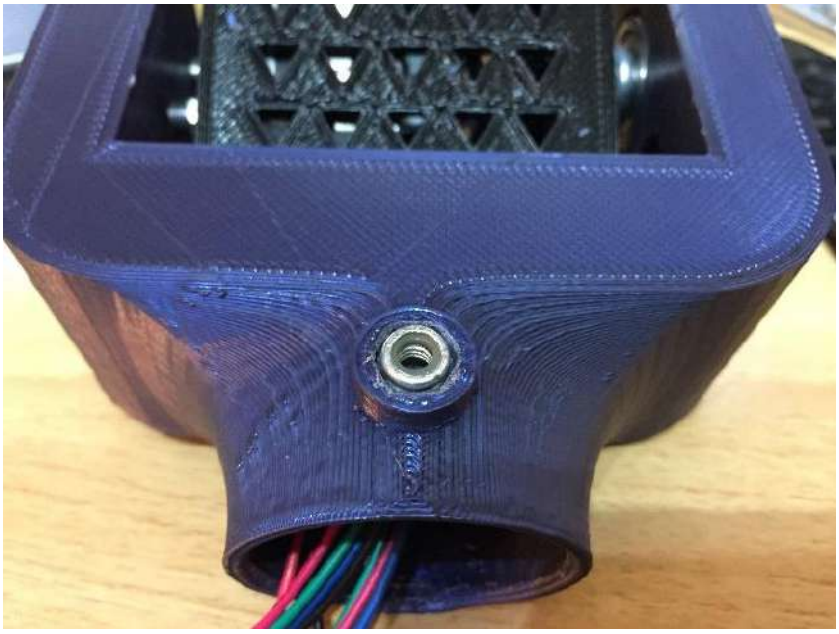


Figura 104 - Tuerca de sujeción de la muñeca

Finalmente, con el tornillo M4x50 se ha de sujetar con el antebrazo (Figura 105 - Unión de la muñeca y el antebrazo).

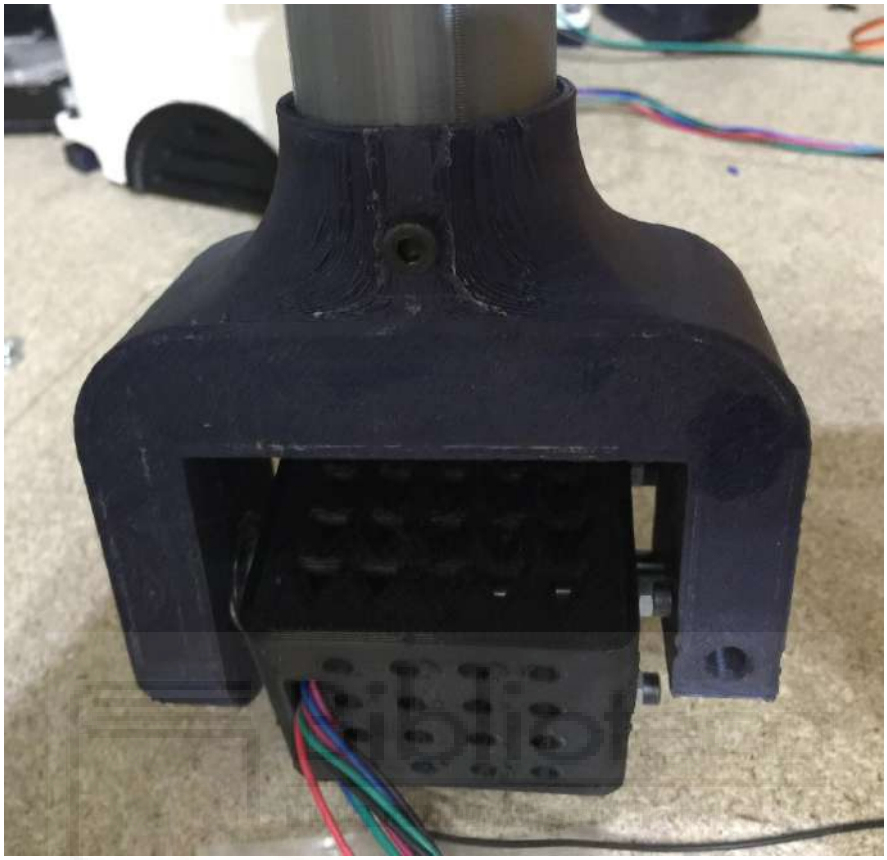


Figura 105 - Unión de la muñeca y el antebrazo