UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO, IMPLEMENTACIÓN Y CONTROL DE UNA PLATAFORMA DE 4 RUEDAS OMNIDIRECCIONALES CON BRAZO ROBÓTICO DE 6 GRADOS DE LIBERTAD

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA MECATRÓNICA

AUTOR:
Adrián Yueyan Tham Ochoa

TUTOR:
Vicente Fermín Casanova Calvo

RESUMEN

Este trabajo engloba todos los procesos y estudios pertinentes para la concepción de un manipulador móvil constituido por una plataforma omnidireccional y un brazo robótico de 6 grados de libertad, orientado a tareas de Pick&Place.

Abarca desde el diseño de los diferentes componentes utilizando el software CATIA V5, la creación de los modelos de simulación usando Matlab 2018b y su producto Simscape Multibody para anticipar su comportamiento; hasta su implementación real usando piezas de impresión 3D y el microcontrolador Arduino Due junto a diferentes elementos electrónicos y mecánicos. Para el control del brazo se crea una aplicación Android con interfaz personalizada para su uso, que comunica con el microcontrolador vía Bluetooth. El poder llevar a cabo la construcción desde cero de un aparato mecatrónico es lo que me motivó e impulsó a realizar este proyecto, a lo largo del cual he tenido la oportunidad de poner en práctica y ampliar el vasto abanico de competencias que he adquirido durante mis estudios de máster.

SUMMARY

The project encompasses all the processes, ranging from the initial idea to the assembly of a fully functional mobile manipulator consisting of an omnidirectional platform and a 6- Degree-of-Freedom robotic arm whose objective is to carry out Pick&Place tasks.

It looks into the design of various components using the CAD/CAM software CATIA V5 and the development of numerous simulation models employing Matlab2018b and its add-on product, Simscape Multibody, to anticipate the behaviour of the mechatronic system. Also included are the assembly and implementation phases operating with 3D printed parts and the Arduino Due microcontroller used in conjunction with diverse electronic and mechanical elements. A customized Android App is developed in order to control the robotic arm using Bluetooth connection. Being able to develop the construction of a mechatronic system from scratch was what motivated and propelled me to making this project become a reality, during which I have had the opportunity to put into practice and expand the vast range of know-how and skills I have acquired during my master's studies.

ÍNDICE

1 INTRODUCCIÓN Y OBJETIVOS

Este proyecto nace de la motivación por hacer realidad la implementación de un sistema mecatrónico, abarcando todas las fases necesarias. Desde la concepción de la idea inicial, el diseño de las piezas que lo darán forma y funcionalidad, la selección de los componentes, el diseño de los diferentes modelos de simulación que tratan de anticipar el comportamiento del sistema hasta el montaje y puesta en funcionamiento de este. Habiendo realizado el Grado en Ingeniería Mecánica, la parte referente al diseño se considera inherente, pero todo aquello relativo al control, la programación y la electrónica, en especial la parte de diseño de entornos de simulación y sistemas de control, que constituyen el grueso de este estudio, suponen un reto por mi corta experiencia trabajando en estas áreas, unida prácticamente de manera exclusiva a este último año y medio de máster. Por ello, me parece el colofón ideal para mis estudios, pues tengo la oportunidad desarrollar un proyecto haciendo uso de conocimientos ya adquiridos de antemano a la vez que pongo en práctica y trato de ampliar mis competencias en áreas más novedosas y desconocidas para mí.

En primer lugar, el brazo robótico es a día de hoy uno de los sistemas mecatrónicos cuyo uso está más extendido a nivel industrial y en cada vez más ámbitos de la vida pública y privada, donde destaca el desarrollo de robots colaborativos o cobots, destinados a trabajar de manera conjunta con los seres humanos. Sea cuál sea su morfología o naturaleza, todos tiene el mismo fin, hacer la vida más fácil a las personas. Ya sea automatizando tareas tediosas y repetitivas o aquellas que requieren un alto nivel de precisión y repetibilidad.

Personalmente, durante el año que trabajé en el sector de la automoción, los brazos robóticos, concretamente de 6 grados de libertad, estaban por todas partes. La alimentación de formatos a las prensas de estampación en frío (CSL) y de estampación en caliente (HSL) así como la recogida y apilamiento de estos, listos para ser sometidos al siguiente proceso. Las células de soldadura estaban formadas por una mesa giratoria donde se colocaban los formatos y eran dos robots los que realizaban todos los puntos y cordones de soldadura. Cuando dejé mi trabajo en el departamento de Corte Láser, ya se estaba estudiando la posibilidad de sustituir al operario que realizaba la carga y descarga de formatos por un brazo robot, siendo ya uno de ellos el que realizaba la trayectoria de corte. Es decir, su aplicación y despliegue es exponencial y de carácter global.

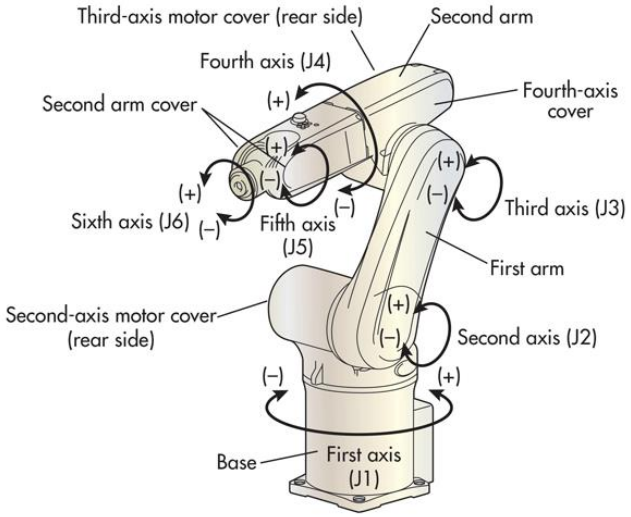


Ilustración 1: Brazo robótico de 6 grados de libertad (Fuente: Google Images)

En cuanto a la plataforma omnidireccional, forma parte de un grupo de vehículos cuya característica principal es poder realizar movimientos en cualquier dirección sin necesidad de orientarse, pese a que también poseen dicha capacidad. Esto les dota de una flexibilidad fundamental a la hora de moverse en espacios reducidos o con múltiples obstáculos. Hoy día van en notable aumento las fábricas que utilizan AGVs (Automatic Guided Vehicle) para transportar piezas y componentes de manera automática y en los últimos años empresas punteras están desarrollando los AIV (Automatic Intelligent Vehicle), los cuales en vez de seguir una referencia magnética subterránea, se mueven libremente por un recinto que previamente han mapeado, usando técnicas de balizamiento. Es a estos últimos a los que más se parece la Plataforma Omnidireccional, con diferencias de concepto y de funcionamiento, pero en naturaleza tienen el mismo principio, trasladarse de manera automática y con libertad de movimiento de un punto de salida a uno de destino.

Ilustración 2: Concepto de AIVs trabajando en entorno industrial (Fuente: Omron)

Analizando como ambos sistemas mecatrónicos son constantemente desarrollados y ampliamente utilizados, surge la pregunta de por qué no aunar ambos conceptos en un mismo sistema que permita la recogida, transporte y deposición de objetos o el movimiento hacia una determinada posición dónde realizar labores de manipulación. Esto último cobra especial interés en lugares de difícil acceso, que conlleven un riesgo para el ser humano o en situaciones en las que simplemente sea más fácil y rentable que lo realice un robot diseñado para ello.

Así surge la idea de este proyecto, tratar de realizar de manera íntegra un manipulador móvil, que es el nombre utilizado para este tipo de robots que combinan un brazo robotizado sobre una plataforma móvil, que sea capaz de desplazarse de manera omnidireccional y automática siguiendo unas trayectorias predefinidas así como realizar labores de manipulación, sobre todo orientado a Pick&Place, durante el proceso.

Actualmente la referencia a nivel internacional en manipuladores móviles es Robotnik. En colaboración con empresas referentes del sector como Universal Robots o Schunk, han desarrollado un portfolio de manipuladores móviles con diferentes fines dentro de los procesos industriales, como el que se muestra a continuación.

Ilustración 3: Manipuladores móviles RB-Kairos de Robotnik (Fuente: Robotnik)

Para el manipulador se decide usar un brazo robótico de 6 grados de libertad ya que, como se ha comentado, su uso considerablemente extendido y su versatilidad de movimientos, unido a lo familiar que resulta por haber trabajado previamente con brazos de dicha configuración en otras asignaturas del máster lo convierten en una elección idónea.

El objetivo será, en una primera instancia desarrollar un modelo de simulación lo más fiel posible a la realidad en el que se simule su comportamiento frente a diversas referencias de posición usando diferentes sistemas de control. Una vez realizados los estudios pertinentes y habiendo comprobado la viabilidad del sistema se iniciará el proceso de montaje, en el cual se pretende obtener una réplica completamente funcional del modelo de simulación, con un control vía Bluetooth usando una aplicación Android creada de manera específica para su manejo.

En el caso de la Plataforma, hay múltiples combinaciones que resultan en un vehículo omnidireccional si se tiene en cuenta la estructura de este o el tipo y número de ruedas. Para el caso particular que concierne este estudio se decide que tenga una forma circular y 4 ruedas omnidireccionales. Este tipo de configuración resulta en una serie de ventajas frente a su alternativa más común, el de 3 ruedas. En primer lugar posee redundancia, para lograr cualquier movimiento en el plano existen varias combinaciones de velocidades de sus ruedas. También son más potentes, tienen más tracción y menos deslizamiento, para la corrección del cual además existe un método para detectar si alguna de las ruedas está sometida a él en algún momento [1]. Obviamente estas ventajas van en detrimento del principal beneficio de un robot de 3 ruedas, su simplicidad de diseño y un menor consumo energético.

El enfoque será el mismo que en el caso del Brazo, crear un entorno y modelo de simulación de manera que se repliquen las condiciones reales de la manera más aproximada posible y un sistema de control para que el vehículo se comporte de la manera deseada, siguiendo de la forma más exacta posible una trayectoria de movimiento definida a partir de cuantos puntos de destino se deseen. Una vez hecho esto, de nuevo el plan es continuar con el proceso de montaje y puesta a punto para su funcionamiento real.

Para terminar se unirán ambos modelos en un único entorno de simulación, con objeto de observar lo que sería su funcionamiento en el mundo real, para después compararlo con el sistema real implementado.

Por desgracia (esto está escrito a posteriori) debido a las circunstancias en España y el Mundo causadas por el brote del Coronavirus y cómo estas han alterado el correcto y habitual funcionamiento de casi la totalidad de servicios y actividades económicas y educativas, la fase de implementación real de la Plataforma Omnidireccional no ha podido llegar a realizarse para decepción del autor. Por ello, se incluirá este proceso dentro del apartado de Futuras Mejoras: 7.2 Implementación real de la plataforma omnidireccional.

2 ESTUDIO TEÓRICO

Cuando se trata de brazos robóticos, los pilares fundamentales de su estudio se dividen en dos campos, la cinemática y la dinámica. La primera de ambas, se divide a su vez en dos áreas, el problema cinemático directo y la cinemática inversa, que como sus propios nombres indican aportan enfoques opuestos respecto al movimiento del sistema. La dinámica en cambio, se centra en la relación entre las fuerzas que actúan sobre un cuerpo y el movimiento que en él se origina. El estudio de ambas es tremendamente extenso, detallado y avanzado pero no es el objeto de este proyecto, pues a pesar de estar utilizando un manipulador de 6 grados de libertad, el enfoque es que los servomotores que confieren movimiento a sus articulaciones sean controlados de manera manual a través de la aplicación diseñada para ello o realizando trayectorias en función de la introducción de valores de referencia para sus articulaciones, por lo que no es necesario abordar dichas áreas de conocimiento.

Pese a esto, se presenta de manera sucinta en qué consiste el problema cinemático pues se considera que es la base para el estudio de brazos robot y puede ser de ayuda al lector si no está familiarizado con ellos.

El problema cinemático directo *consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot [2]*. La cinemática inversa, como su propio nombre indica, hace la labor contraria, *resuelve la configuración que debe adoptar el robot para una posición y orientación del extremo conocidas [2]*. Estos conceptos se ven reforzados con la imagen que sigue, que se trata de un diagrama que los relaciona.

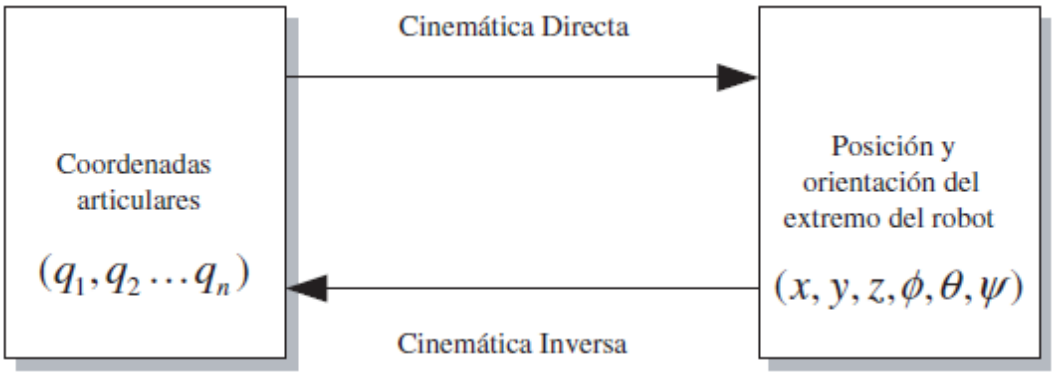


Ilustración 4: Relación entra la Cinemática Directa y la Cinemática Inversa [2]

Como se ha dicho, al no ser tratados estos casos de estudio a lo largo del proyecto, no se ahondará más en ellos. De todas formas, si algún lector tiene interés en conocer

más sobre este campo, se recomienda la lectura de la obra *Fundamentos de Robótica* [2].

En lo que concierne a la Plataforma Omnidireccional la manera de abordar el problema sí requerirá de especial atención a su estudio cinemático, pues al tratarse de un vehículo cuya principal característica es su habilidad de moverse en todas direcciones, el cómo conseguirlo es la base sobre la que se desarrolla esta parte del proyecto.

En primer lugar recordar que el vehículo elegido tiene forma circular y lo componen 4 ruedas omnidireccionales. Por ello, el estudio se basa en lo expuesto en el artículo científico *Design and Analysis of a Four-wheel Omnidirectional Mobile Robot* [3].

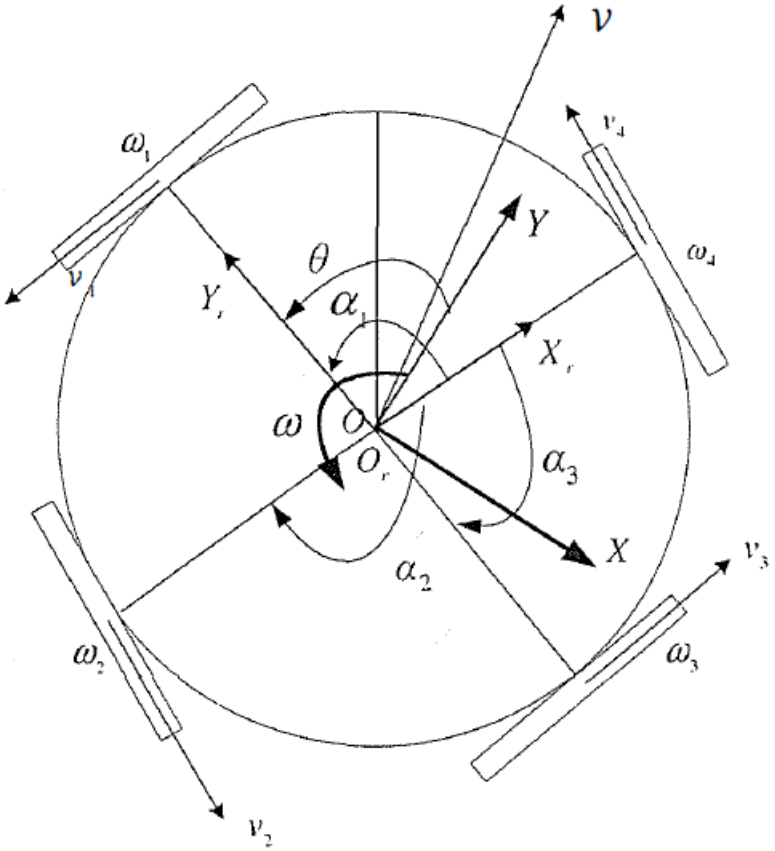


Ilustración 5: Robot circular de 4 ruedas omnidireccionales [3]

La figura superior muestra la vista cenital del robot en cuestión. El sistema de coordenadas formado por $X_rO_rY_r$ está fijado al robot mientras que el sistema *XOY* se corresponde con el sistema de coordenadas global de referencia. El radio de las ruedas y el radio del robot, se denominan *r* y *b* respectivamente. Los ángulos entre los ejes de las ruedas son denotados como α_i $(i = 1,2,3,4)$. La velocidad angular de cada rueda es denotada como ω_i $(i = 1,2,3,4)$. La dirección de la velocidad lineal del

centro de cada rueda con respecto al eje de coordenadas $X_rO_rY_r$ se indica como v_i $(i = 1,2,3,4)$. El ángulo entre los sistemas de coordenadas $X_rO_rY_r$ y XOY se representa como θ. Las velocidades linear y angular del robot son denotadas como $v = [v_x\ v_y]^T$ y ω respectivamente. Para los cálculos se asume que el eje X_r está alineado con el eje de la 4ª rueda. De esta manera, existe la siguiente relación cinemática:

$$r\omega_i = b\omega + {v_r}^T v_i\ (i = 1,2,3,4)$$

Donde

$$v_1 = [-\sin(\alpha_1) \quad \cos(\alpha_1)]^T \tag{1}$$

$$v_2 = [-\sin(\alpha_2) \quad \cos(\alpha_2)]^T \tag{2}$$

$$v_3 = [-\sin(\alpha_3) \quad \cos(\alpha_3)]^T \tag{3}$$

$$v_4 = [0 \quad 1]^T \tag{4}$$

$$v_r = [v_{rx} v_{ry}]^T$$

$$v_{rx} = v_x \cos\theta + v_y \sin\theta$$

$$v_{ry} = -v_x \sin\theta + v_y \cos\theta$$

De las ecuaciones (1) a (4) las velocidades angulares de las ruedas son derivadas como función de las velocidades lineal y angular del robot.

$$\begin{pmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{pmatrix} = \frac{1}{r}\begin{pmatrix}-\sin(\alpha_1) & \cos(\alpha_1) & b\\ -\sin(\alpha_2) & \cos(\alpha_2) & b\\ -\sin(\alpha_3) & \cos(\alpha_3) & b\\ 0 & 1 & b\end{pmatrix}\begin{pmatrix}v_{rx}\\ v_{ry}\\ \omega\end{pmatrix} \tag{5}$$

De esta manera, dadas la velocidad lineal $v = [v_x\ v_y]^T$ y la velocidad angular ω deseadas, se pueden obtener de manera directa las velocidades necesarias de las ruedas ω_i $(i = 1,2,3,4)$ usando la expresión (5).

Pero como se ha mencionado al inicio, el objetivo es que la plataforma realice una determinada trayectoria. Por tanto, se ha de saber qué valores de $v = [v_x\ v_y]^T$ y ω son necesarios en cada instante. Esto, pese a tener también una componente teórica, decide explicarse con más detenimiento en 4.2.2 Simulación de trayectorias junto a las estructuras de control diseñadas para ello, pues se entiende que su entendimiento será más intuitivo para el lector.

3 DISEÑO DE COMPONENTES SIMULADOS

Esta fase abarca la obtención en formato STEP, propio de los programas de diseño CAD/CAM, de las diferentes piezas que conforman tanto el brazo robótico como la plataforma omnidireccional. Dichos archivos serán utilizados en la fase de diseño del modelo de simulación.

La consecución de dichos archivos tiene dos orígenes diferentes. Una parte se ha diseñado por el autor de este trabajo, en particular aquellos componentes de especial interés para el correcto funcionamiento del sistema. Para el diseño de estos, se ha utilizado el programa de diseño CAD/CAM CATIA V5. Otros elementos como en el caso de los diferentes servomotores o motores de corriente continua utilizados, han sido obtenidos de repositorios ubicados en la red.

3.1 Brazo Robot

3.1.1 Base giratoria

Es la base sobre la cual se sustenta el brazo robótico y que irá anclada al bastidor. Permite un giro de 360º en torno al eje perpendicular a la base del bastidor mencionado. Posee una serie de taladros para su sujeción al Servomotor 1 que le conferirá el movimiento rotacional así como un alojamiento para el Servomotor 2 que hará lo propio con el siguiente eslabón del brazo. Aclarar que el orden de notación de los servomotores se rige por su posición, siendo el primero el más cercano al bastidor y el último el más alejado, situado en el Gripper. No todos los servos son iguales, como se explica en el apartado 3.1.6 Servomotores dentro de este capítulo.

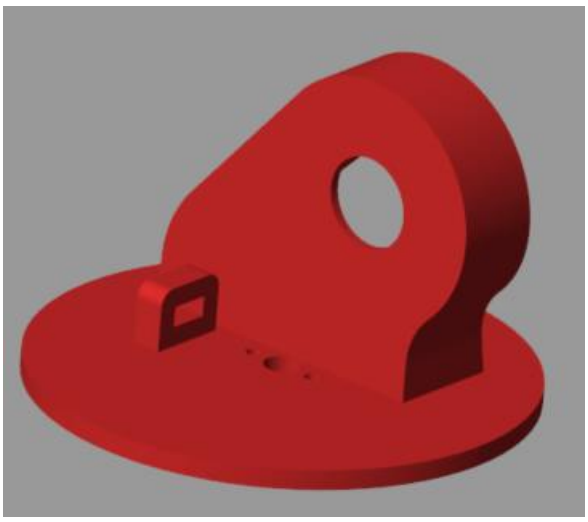


Ilustración 6: Base giratoria

3.1.2 Brazo 1

El segundo componente del brazo robótico es un eslabón cuyo movimiento será rotacional solidario al del servomotor alojado en la base giratoria mencionado anteriormente. Es totalmente simétrico respecto a su eje transversal. Alberga una serie de taladros para su sujeción a los Servomotores 2 y 3 y un conducto que lo atraviesa longitudinalmente para el paso del cableado de los diferentes servomotores utilizados en el montaje. Este tiene como fin conseguir una maniobrabilidad más práctica así como lograr un mayor acabado estético.

Ilustración 7: Brazo 1

3.1.3 Brazo 2

El Brazo 2, como se ha decidido denominarlo en el conjunto, se caracteriza por albergar dos servomotores en los huecos que en él han sido diseñados. Siguiendo el orden (comienzo por la base y final en la herramienta), el Servomotor 3 se sitúa en el vaciado que se encuentra más próximo al extremo redondeado de este eslabón. Este otorga un movimiento rotacional al Brazo 2 y se ancla tanto a este como al Brazo 1 a través de los tornillos que irán alojados en los taladros realizados para ello. En la cavidad situada en el otro extremo se aloja el Servomotor 4, encargado de la motricidad del siguiente eslabón.

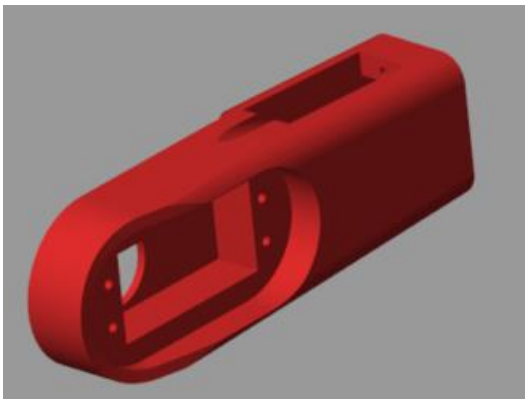


Ilustración 9: Vista frontal Brazo 2

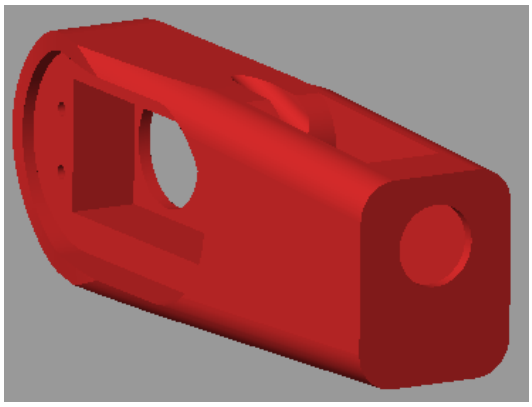


Ilustración 8: Vista trasera Brazo 2

3.1.4 Brazo 3

Este eslabón se mueve rotacionalmente en torno al eje que pasa perpendicularmente por el centro de la circunferencia que define el agujero mostrado en la Ilustración 8: Vista trasera Brazo 2. De manera que se define comúnmente como "muñeca" del robot por su movimiento análogo al de dicha articulación. También posee una cavidad para situar el Servomotor 5, encargado del movimiento de la base del Gripper.

Ilustración 10: Brazo 3

3.1.5 Gripper

Este componente entraña una mayor complejidad que los expuestos anteriormente debido a que está conformado por diferentes subelementos, sin tener en cuenta servomotores. Aunque a efectos prácticos se trata de dos eslabones en uno. El primero lo constituye una base sobre la cual se anexionan los diferentes subelementos. El segundo, los elementos citados, los cuales se mueven en armonía. Entre ellos se encuentran dos engranajes, uno de ellos, el motriz, acoplado al Servomotor 6. Dos pinzas cuya función es la de sujetar el objeto manipulado y cuatro elementos de unión entre estas y la base. Con el giro del Servomotor 6 en un sentido o en otro, se consigue la apertura o cierre de las pinzas manipuladoras.

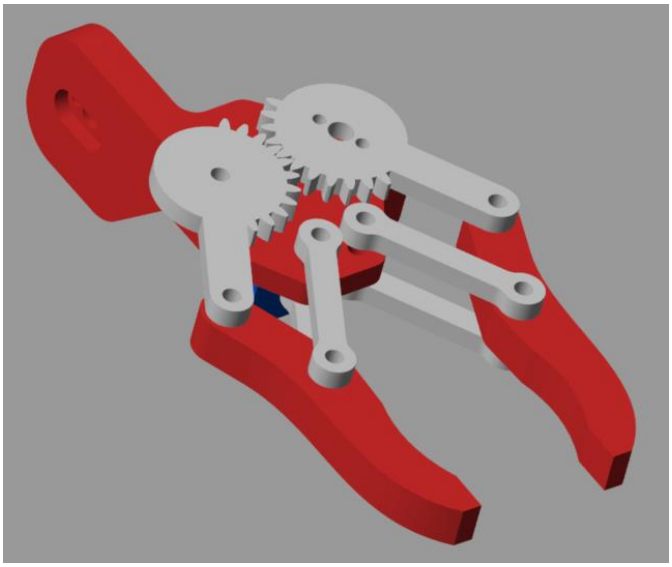


Ilustración 11: Gripper

3.1.6 Servomotores

Son los encargados de conferir movimiento a la estructura y para este proyecto se han utilizado dos tipos diferentes, en función de las necesidades de par en cada caso concreto (el proceso de dimensionamiento de estos se trata en el apartado 5.1.2 Servomotores). De manera que se ha utilizado el modelo MG996R para los Servomotores 1,2 y 3. Y el modelo SG90 para los Servomotores 4,5 y 6.

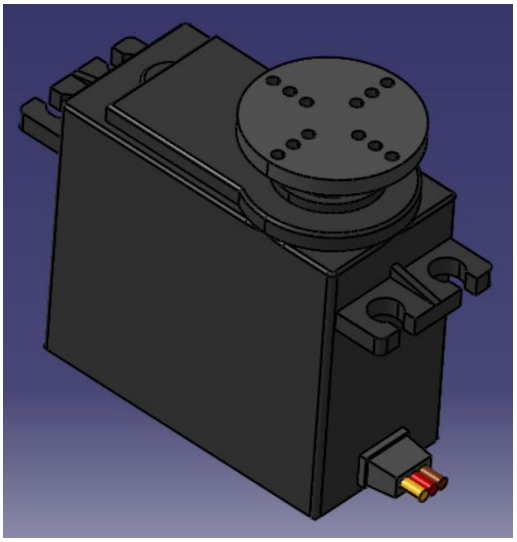


Ilustración 12: Servomotor MG996R

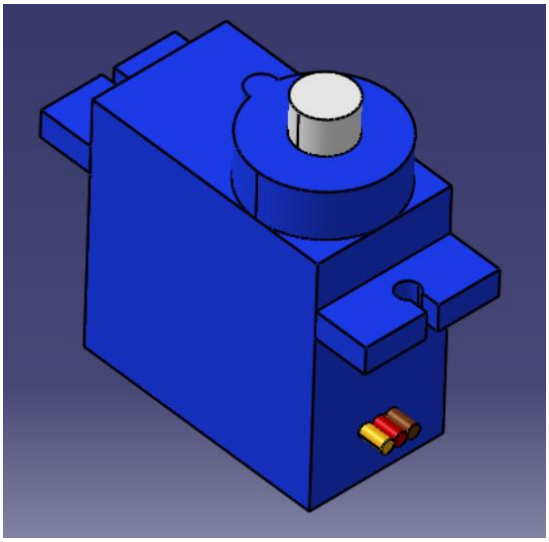


Ilustración 13: Servomotor SG90

3.2 Plataforma Omnidireccional

3.2.1 Chasis

Se trata de la pieza principal de la plataforma, pues es la que albergará el microcontrolador en el que se ejecutará el control tanto del brazo como de la plataforma omnidireccional junto a todas sus conexiones con los diferentes elementos actuadores y de sensórica. En su interior se encontrarán alojados también los motores de corriente continua encargados de conferir movimiento a las ruedas

de la plataforma, los cuales irán anclados con tornillería a las paredes de este. Cuenta con varias rejillas de ventilación tanto en la pared como en la base inferior en las zonas más próximas a los motores, con el fin de facilitar la refrigeración de estos y cualquier componente que pudiera calentarse. Por último notar que se ha diseñado en dos partes diferentes: el cuerpo principal del chasis y la tapa. Esto se ha hecho así para facilitar la colocación de los diferentes componentes que van en su interior a la hora de realizar el modelo de simulación en Simscape Multibody.

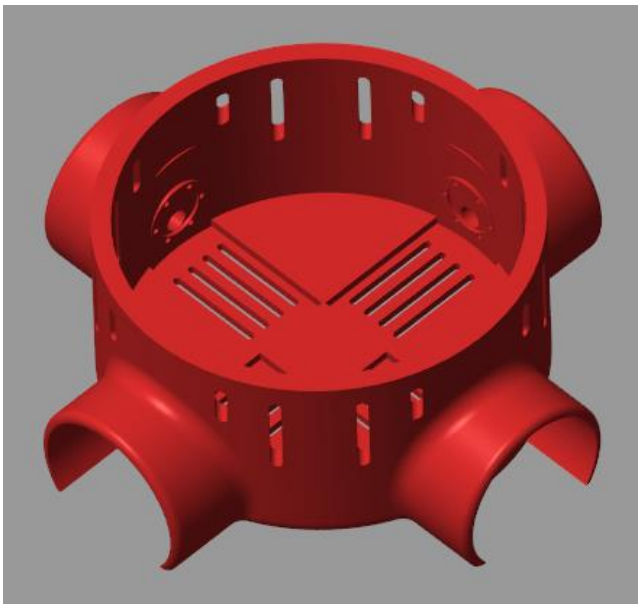


Ilustración 14: Chasis plataforma omnidireccional

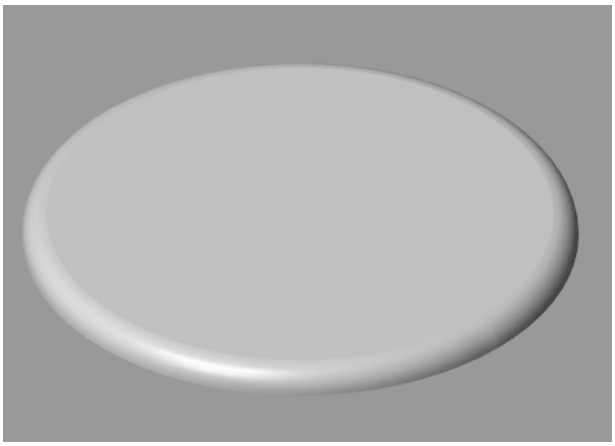


Ilustración 15: Tapa del chasis

En referencia a los taladros de la tapa para el anclaje de la base giratoria del robot, se ha decidido no ponerlos y realizarlos manualmente en la fase de montaje, por si hubiera que añadir alguna guía o soporte para reducir los momentos flectores producidos por el brazo sobre la unión durante el funcionamiento.

3.2.2 Motores CC

Para impulsar la plataforma se hace uso de 4 motores de corriente continua de la marca Pololu, uno por cada rueda, concretamente de la gama 37D. De dicha gama se selecciona, dentro de los modelos que cuentan con un encoder integrado para conocer su posición, aquel que tiene una relación de transmisión 70:1, utilizado en anteriores proyectos de magnitud similar y que por tanto se considera apropiado para este estudio.

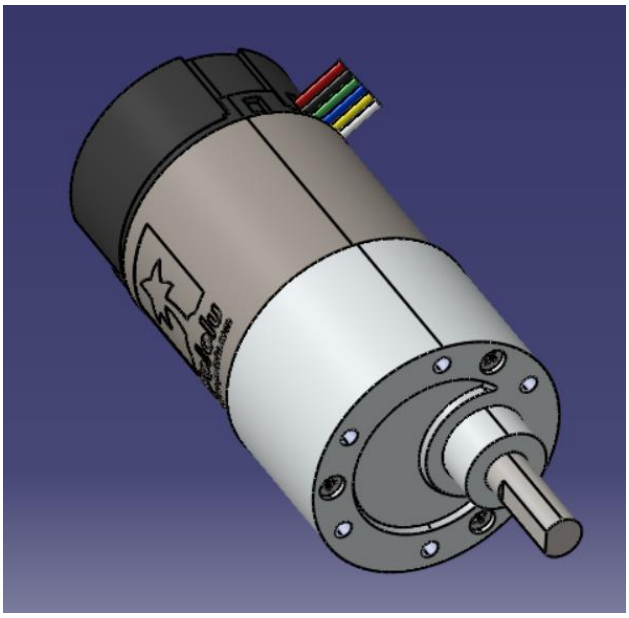


Ilustración 16: Motor CC Pololu 37D-70

3.2.3 Ruedas omnidireccionales

Para que la plataforma posea la capacidad omnidireccional deseada, es necesario que las ruedas sean las apropiadas para ello. Si bien hay otras opciones que confieren dicha característica como las ruedas Mecanum, el estudio teórico mencionado al inicio del trabajo se basa en la utilización de las conocidas como ruedas omnidireccionales. En concreto el modelo a utilizar es la VEX Omni-Directional Wheel 2.75’ de doble rodillo, desarrollado por la conocida distribuidora de productos relacionados con la robótica VEX Robotics.

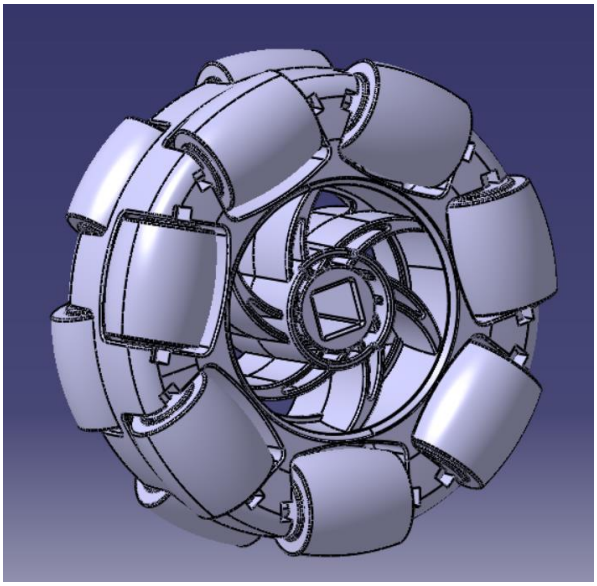


Ilustración 17: Rueda Omnidireccional VEX 2.75' de doble rodillo

3.2.4 Acople

Debido a la discordancia de forma entre el eje del motor, el cual como se puede ver en la Ilustración 16: Motor CC Pololu 37D-70 es cilíndrico con una muesca para aumentar la sujeción, y el alojamiento del eje de la rueda de la imagen superior que tiene forma cuadrada; se diseña un acoplamiento para la correcta transmisión de movimiento entre las partes. Se trata de un modelo aproximado que cumplirá su función en la simulación, pero a la hora de la implementación real será necesario sondear el mercado para encontrar un acople que garantice una restricción absoluta del movimiento relativo entre los componentes.

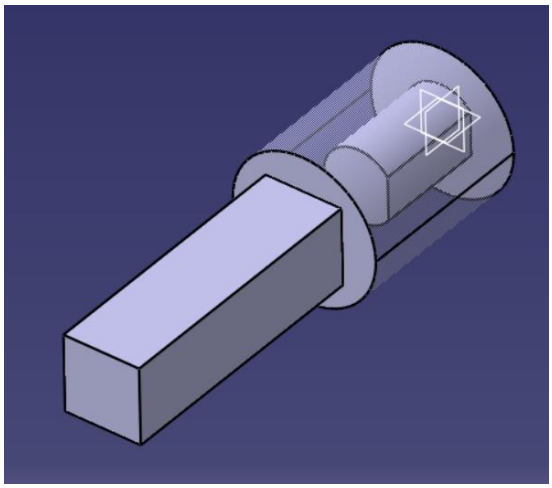


Ilustración 18: Acople eje motor-rueda

4 SIMULACIÓN

Para el proceso de simulación se utiliza Simscape Multibody, una herramienta del entorno de programación en diagrama de bloques Simulink, que a su vez es un producto del programa Matlab, cuya versión utilizada es 2018b.

4.1 Brazo Robot

Esta fase del proyecto de divide en diferentes subetapas, entre las que conviene diferenciar:

- Diseño del modelo de simulación.
- Simulación de trayectorias predefinidas.
- Simulación de movimiento manual.
- Pick & Place

A continuación se procede a explicar las tareas llevadas a cabo en cada una de los puntos anteriores al igual que los resultados obtenidos.

4.1.1 Diseño del modelo de simulación

El método para el diseño del modelo consiste en utilizar bloques de la librería de Simscape Multibody en conjunción con bloques genéricos de las librerías de Simulink y definir tanto sus propiedades como las relaciones que hay entre ellos. Se procede a explicar de manera breve los principales bloques de Simscape Multibody utilizados, pues se entiende que pueden ser los más desconocidos para el lector.

- *Solid:* Este bloque permite definir figuras tridimensionales así como sus valores de inercia, el aspecto que mostrarán durante la simulación o su situación y orientación en el espacio de simulación y con respecto a otros sólidos. En este caso, como se ha expuesto anteriormente, las piezas que conforman el brazo robótico ya han sido diseñadas en CATIA V5, por lo que basta con subir el archivo correspondiente en formato STEP y definir las características citadas.

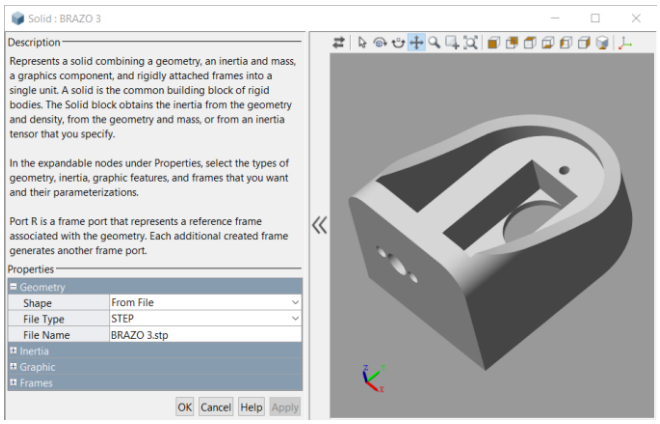


Ilustración 19: Interfaz del bloque Solid

- *Rigid Transformation:* Permite definir las relaciones de traslación y orientación de los sólidos de una manera más completa y en muchas ocasiones más sencilla y efectiva que el bloque Solid. Posibilita introducir de manera numérica la distancia y la orientación entre dos sólidos o entre un sólido y el origen del espacio de simulación.

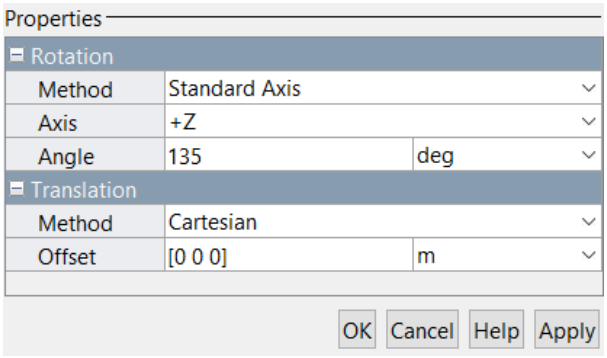


Ilustración 20: Interfaz del bloque Rigid Transformation

- *Joints:* Es un bloque genérico que hace referencia a las diferentes articulaciones que pueden unir los bloques *solid.* En el caso concreto de este estudio se han utilizado exclusivamente articulaciones de revolución, pues para el funcionamiento del Brazo Robot no es preciso ninguna otra. Estos bloques ofrecen un amplio abanico de posibilidades para el diseño, siendo posible definir en la articulación de revolución usada los puntos por los que se quiere que pase durante la simulación, su mecánica interna y equiparla con sistemas de actuación y sensórica.

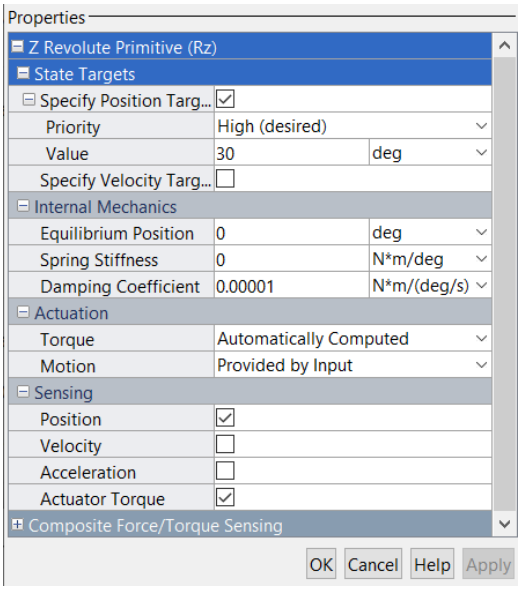


Ilustración 21: Interfaz del bloque Revolute Joint

Haciendo uso de estos y otros bloques disponibles en las diferentes librerías de Simulink se construye el modelo de simulación. El método utilizado comienza con la unión de la Base Giratoria a los tres bloques rodeados en rojo en la Ilustración 22: Modelo de simulación. Siendo cada uno de estos de arriba abajo: *Configuración de los parámetros del solver*, *Sistema de referencia global World* y *Configuración del mecanismo*. Este último permite determinar la aceleración de la gravedad para la simulación. A partir de la base se añaden los diferentes componentes de manera secuencial, determinando las articulaciones y relaciones de transformación necesarias. El orden seguido tiene como fin hacer más intuitivo el proceso de diseño y por ende simplificarlo.

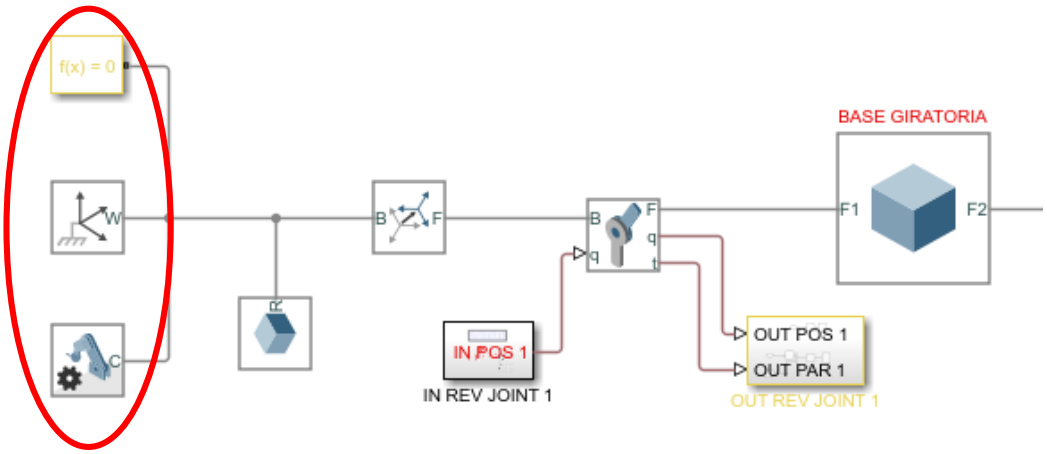


Ilustración 22: Modelo de simulación Brazo Robot

Como se pude observar en la figura, cada articulación (bloque *Joint*) tiene un subsistema de entrada y uno de salida.

El primero, cuyo nombre es IN REV JOINT X (siendo X el número de la articulación en cuestión) está conectado a la entrada *Motion* de la articulación, que hace referencia a su posición. En él se define la referencia de posición introducida a la articulación durante la simulación y los bloques que lo componen dependerán de si las trayectorias a seguir son predefinidas o manuales, por lo que se profundizará sobre ello en los apartados siguientes (4.1.2 Simulación de trayectorias predefinidas, 4.1.3 Simulación de trayectorias introducidas manualmente y 4.1.4 Pick & Place).

En cuanto al subsistema OUT REV JOINT X se encuentra conectado a las mediciones de par y posición de la articulación. Esta última se compara con la referencia de posición introducida en el subsistema anterior. Así se permite la visualización del seguimiento de la referencia introducida y por tanto evaluar el funcionamiento del sistema. Dichas gráficas se mostrarán en apartados siguientes, para estudiar el comportamiento del sistema ante los diferentes tipos de referencia de posición introducidos.

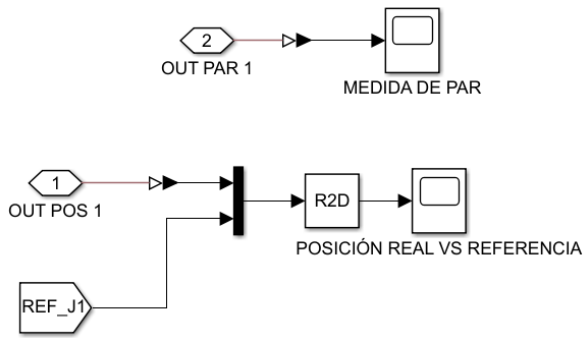


Ilustración 23: Subsistema OUT REV JOINT 1 expandido

Así se suceden los eslabones hasta llegar al Gripper, que al entrañar más complejidad desde el punto de vista constructivo, que no de control, se ha decidido agrupar dentro de un subsistema con el fin de simplificar el modelo principal. De igual manera cuenta con subsistemas para definir su posición y graficarla. A parte de las articulaciones de revolución ya vistas, para el contacto entre engranajes es necesario definir una articulación específica cuyos parámetros son la distancia entre centros de los engranajes y la relación de transmisión.

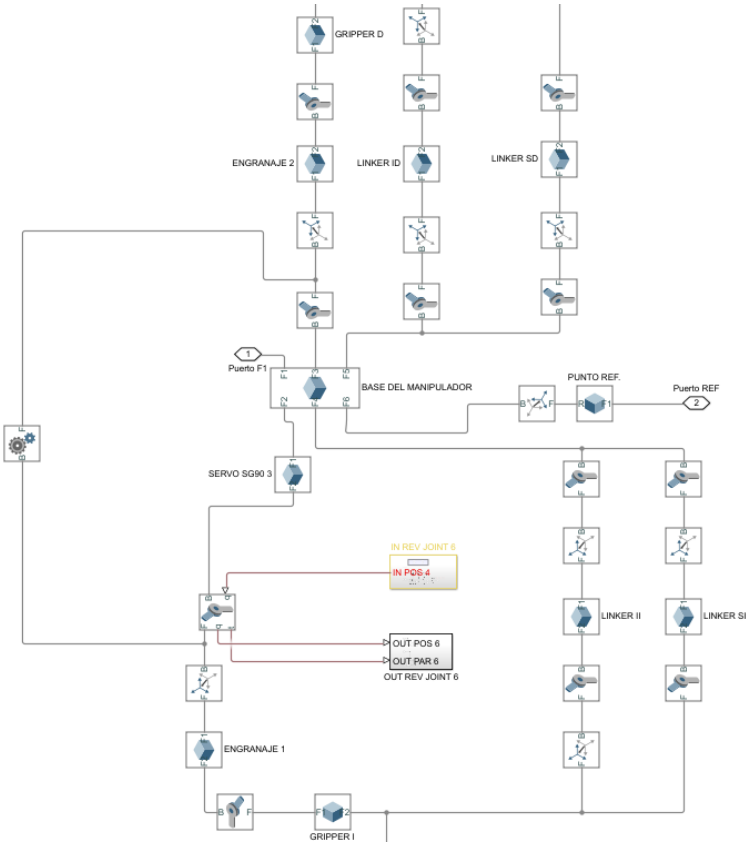


Ilustración 24: Subsistema GRIPPER expandido

Tienen especial interés el Puerto F1 que se muestra en la figura cuyo objetivo es la conexión del *Gripper* al eslabón anterior. Puerto REF está conectado a un bloque *Solid* llamado Punto REF situado en el punto de manipulación del Gripper (donde se encontraría la pieza manipulada, equidistante a ambas pinzas). Este bloque Punto REF tiene carácter auxiliar, de manera que habilita el seguimiento del centro del eslabón manipulador del Brazo Robot, para el que reviste especial interés conocer su localización en coordenadas X,Y,Z respecto a nuestro sistema global de referencia *World.* Para ello se define el punto medio entre las pinzas y se hace su seguimiento haciendo uso del bloque *Transform Sensor* que permite medir posición, velocidad o aceleración entre otras, respecto un sistema de referencia. Los valores medidos, se guardan en el *Workspace* de MATLAB en forma de *array* para poder usarlos en la elaboración de gráficas de trayectoria.

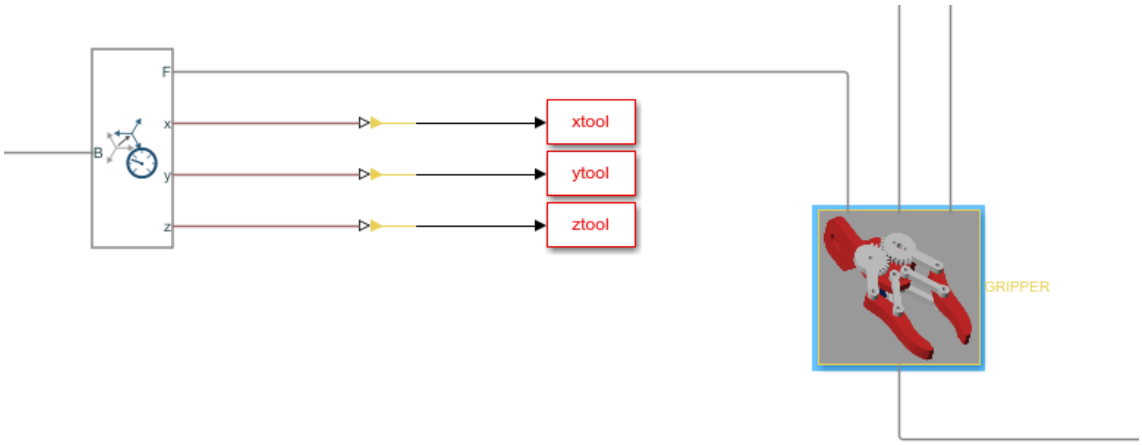


Ilustración 25: Subsistema GRIPPER y bloque Transform Sensor

Por último, se muestra el modelo de simulación al completo, el cual se verá modificado cuando se simulen referencias manuales, pero eso será más adelante. Aunque es complejo discernir los nombres de los bloques, el fin de la ilustración es mostrar el modelo en su conjunto. Se entiende además que con lo explicado anteriormente se puede identificar de manera rápida y sencilla cada una de las partes que se muestran.

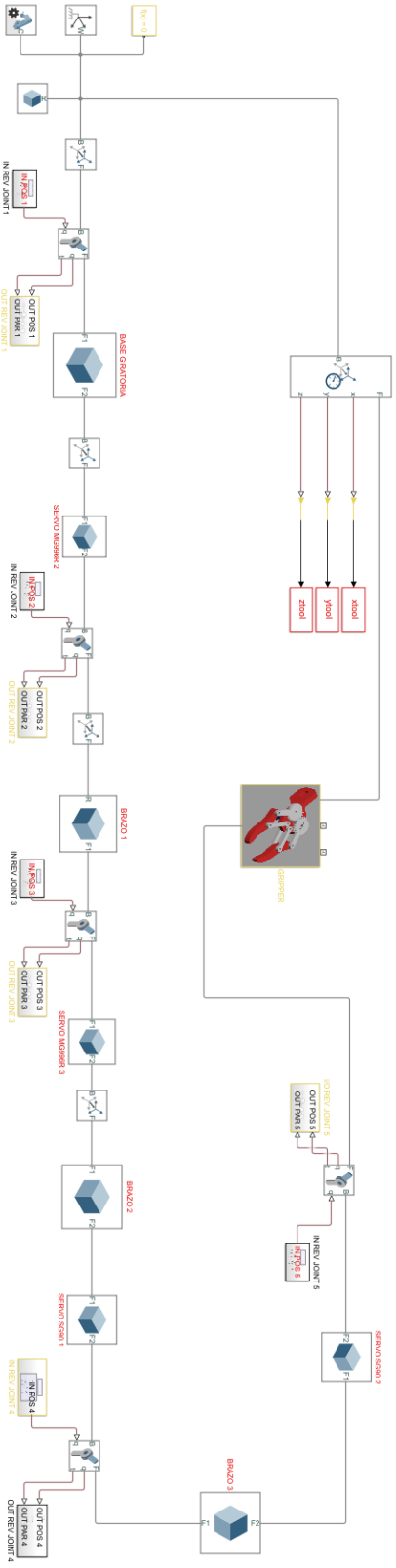


Ilustración 26: Modelo de simulación Brazo Robot (2)

4.1.2 Simulación de trayectorias predefinidas

A cada articulación se le introducen unas secuencias de movimiento utilizando diferentes bloques de tipo *Source.* Es importante resaltar que sea cual sea el tipo de referencia que se introduce a la articulación debe ser filtrada por el bloque *Rate Limiter.* Este limita la pendiente de la referencia introducida y se parametriza en función de las características técnicas del servomotor en cuestión. En el caso de las tres primeras articulaciones el servomotor utilizado es el MG996R cuya velocidad angular de giro trabajando a una tensión de alimentación de 4.8V es de 60º/0.19s. Convirtiendo las unidades a radianes y haciendo una regla de tres, en un segundo el servomotor podría girar 5,51 radianes. Este valor es el introducido como pendiente máxima en el limitador. De igual manera se hace para el servomotor SG90 utilizado en las tres articulaciones restantes. Según su datasheet posee una velocidad angular de 60º/0.12s trabajando a 4.8V. Calculando de manera análoga al caso anterior la pendiente máxima será de 8.72 rad/s. Así, una vez especificada la dinámica de los diferentes servomotores de nuestro modelo, se incluye el *Rate Limiter* en la entrada de cada una de las *Joints* con el valor de pendiente máxima correspondiente al servo que mueva dicha articulación. Este filtro debe estar presente en todas las simulaciones que se hagan con este modelo, pues se trata de una limitación física del material utilizado.

A continuación se muestra la disposición de bloques para la generación de trayectorias predefinidas, que no son modificadas durante la simulación. Como se puede ver, se usan tanto bloques *Step* cuya acción tiene lugar en diferentes momentos de la simulación, bloques S*inwave* que aplican una referencia en forma de onda senoidal o bloques *Repeating Sequence*, que repiten una secuencia de valores definidos en cada instante de tiempo.

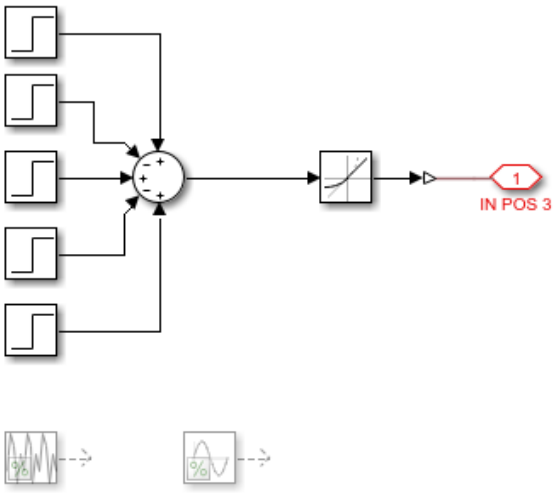


Ilustración 27: Generación de referencias

En la Ilustración 28: Seguimiento de referencia en Joint 2 se puede observar la trayectoria de la Articulación 2, la que posibilita el movimiento entre la Base Giratoria y el Brazo 1, en comparación con la referencia de posición introducida antes de comenzar la simulación.

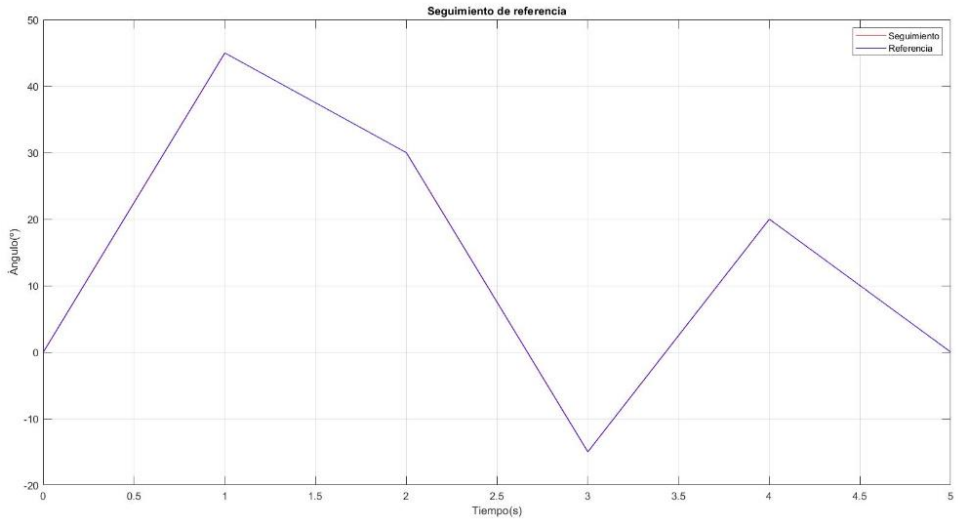


Ilustración 28: Seguimiento de referencia en Joint 2

A primera vista parece que el seguimiento es perfecto, pues se aprecia que ambas líneas de puntos están prácticamente superpuestas. Para evaluar si el posicionamiento de la articulación es tan exacto como parece se toman capturas con un grado de detalle superior. Con el fin de asegurar una comprobación completa, se realiza la captura abarcando el instante más desfavorable, aquel en el que se produce un cambio brusco de magnitud y sentido.

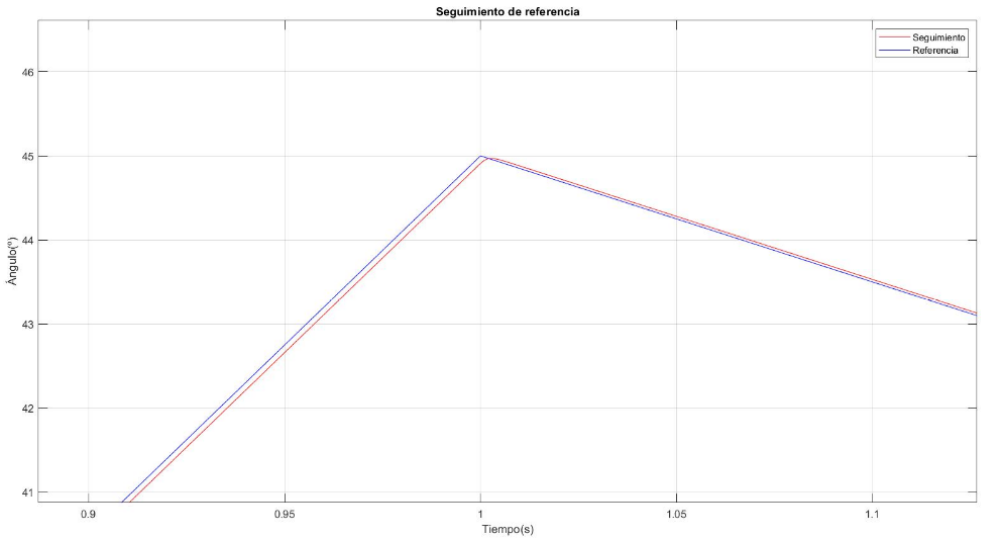


Ilustración 29: Detalle del seguimiento de referencia en Joint 2

Haciendo zoom hasta el punto de reducir los intervalos del eje de abscisas al orden de 5 centésimas de segundo, se puede advertir un ligero retardo que no reviste mayor importancia. Esto es debido a que el retraso entre la referencia y el posicionamiento real de la articulación es del orden de las milésimas de segundo, un valor ínfimo y que no incide de manera significativa en el comportamiento de nuestro modelo de simulación.

Se muestra también un ejemplo del comportamiento de otra de las articulaciones del modelo, en concreto la Articulación 5, que une el Brazo 3 con la base del Gripper. En este caso la referencia de posición es de carácter senoidal.

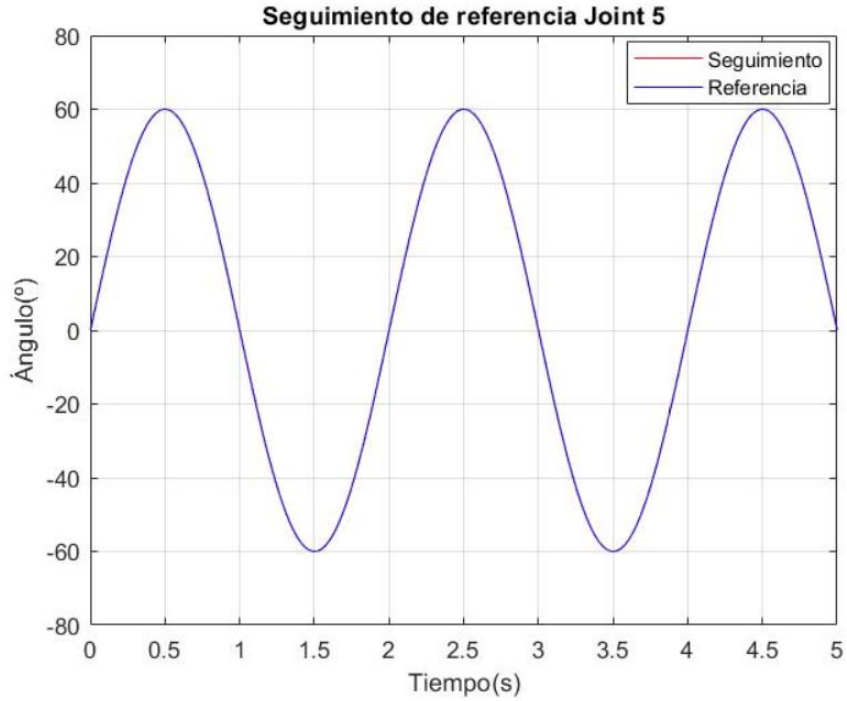


Ilustración 30: Seguimiento de referencia en Joint 5

Una vez más, para garantizar que el seguimiento es realmente preciso se hace zoom para poder interpretar la gráfica de manera más exhaustiva.

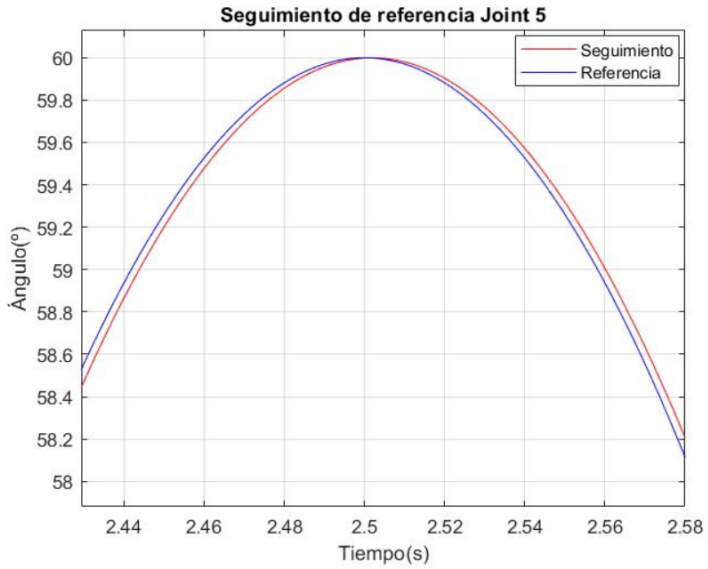


Ilustración 31: Detalle del seguimiento de referencia en Joint 5

De esta forma, queda demostrado el fiel seguimiento de referencias de diferente tipo usando ambos tipos de servomotores (MG99R en Joint 2 y SG90 en Joint 5).

Los valores de referencia utilizados son valores dentro de lo que se considera el funcionamiento normal y típico que tendrá el brazo robótico. Pero es importante considerar también aquellas referencias que no pueden ser seguidas tan precisamente y por ello se considera necesario plasmar un ejemplo en el que se diera tal circunstancia.

Viendo los ejemplos anteriores y sabiendo que nuestro modelo se encuentra limitado por la dinámica de los servomotores, parece obvio que el brazo robótico tendría dificultades cuando los cambios de referencia sean de gran magnitud en un corto periodo de tiempo. Esto queda representado en la Ilustración 32: Seguimiento de referencia en Joint 3 . Se observa como los cambios de posición son tan drásticos que el sistema no puede seguirlos de manera inmediata.

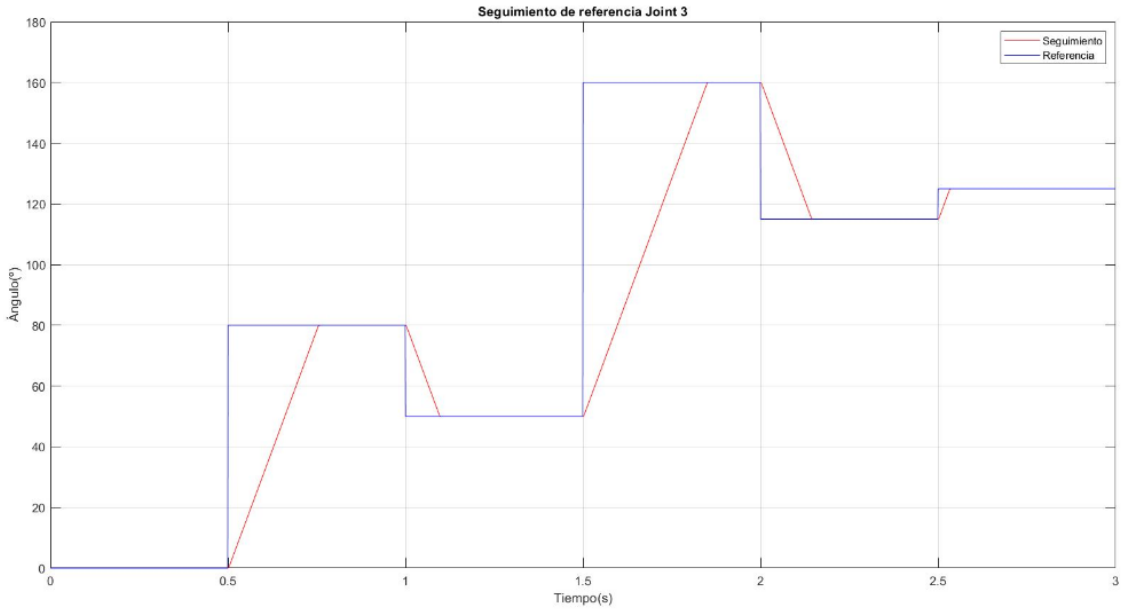


Ilustración 32: Seguimiento de referencia en Joint 3

Podría parecer un problema, pero si se observa con atención la gráfica se aprecia que la articulación recorre más de 100º en menos de 0.5s. Y se trata de la Joint 3, que es movida por un servo MG996R, que son los más lentos que se van a utilizar. Como se ha mencionado anteriormente en la página 26, el MG996R es capaz de cubrir 5.51 radianes en un segundo, eso son unos 315º, lo que supone casi una vuelta completa. Esa velocidad angular es más que suficiente para las labores de manipulación de objetos que se pretende que lleve a cabo el robot. De hecho, una mayor velocidad de giro y por tanto un mejor seguimiento de este tipo de referencias podría suponer un peligro para el funcionamiento del brazo, ya que podrían producirse momentos de inercia que pusieran en peligro la integridad de los componentes.

4.1.3 Simulación de trayectorias introducidas manualmente

La principal ventaja que tiene esta forma de introducir las trayectorias a seguir es que se pueden realizar en tiempo real mientras la simulación está en marcha, de manera que dependiendo de la posición del robot en cada instante se le puede ordenar una acción u otra en función de lo que se quiera conseguir.

Con el fin de que la definición de trayectorias sea más intuitiva y sencilla, se decide usar un mando equipado con joysticks y botones, similar al que se usaría para jugar a la consola.

Para ello se utiliza un *add-on* de Matlab llamado *Simulink 3D Animation* que contiene bloques específicos para reconocer este tipo de periféricos.

La siguiente figura muestra como cada una de las salidas del mando se visualizan usando bloques *Display* y se envían a través de bloques *Goto* a los diferentes subsistemas IN REV JOINT X, donde son filtradas y posteriormente enviadas a la entrada de posición de cada una de las articulaciones del modelo.

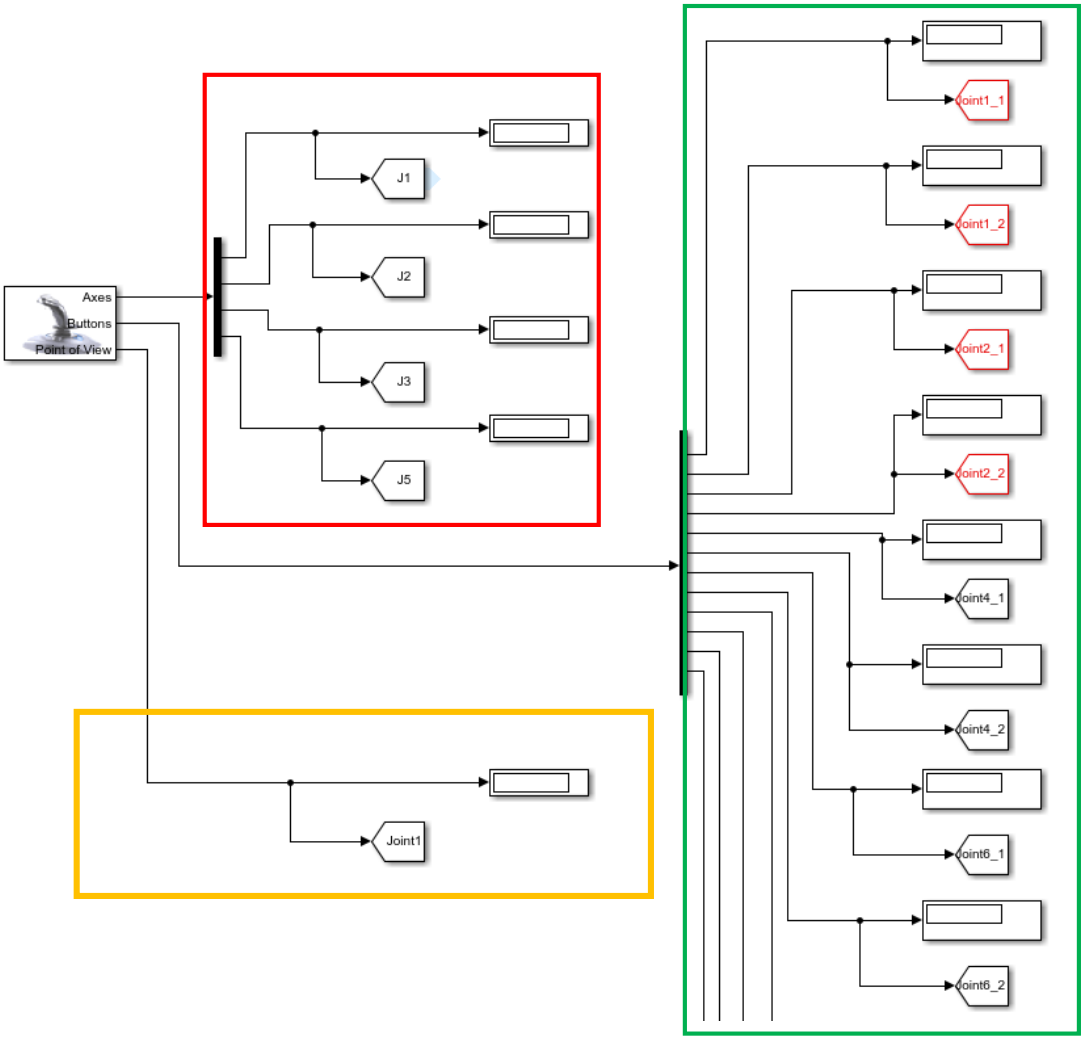


Ilustración 33: Salidas del gamepad o controlador

Observando la figura, las salidas rodeadas en rojo son las referentes a los joysticks del mando y son las encargadas de gobernar el movimiento de las articulaciones 1, 2, 3 y 5. Las rodeadas en verde corresponden a los botones del controlador y en principio sólo serán necesarios los botones 5, 6, 7 y 8 (relativos a los triggers, que son los botones situados en la parte trasera del mando como se indica en la Ilustración 35: Relación entre movimientos del gamepad y movimientos del robot 2) para dirigir las Articulaciones 4 y 6. Se escogen estos botones para que el control se haga usando

joysticks y botones traseros confiriendo una mayor comodidad en el manejo, ya que es difícil manipular los botones delanteros y los joysticks a la vez. De todas formas, se decide mostrar y recoger las salidas del resto de botones ya que se ha configurado un segundo método de uso en el que las articulaciones movidas por los joysticks se puedan manejar con los botones restantes en caso de mal funcionamiento o fallo de los primeros. Por último, la salida rodeada en amarillo corresponde a las flechas del mando y su magnitud es angular. En un principio se decide que sea la que gobierna el movimiento de la Articulación 1 debido a lo intuitivo que resulta, pero se desestima por la poca precisión que tiene la señal.

La relación entre los movimientos del mando y las órdenes enviadas se muestra en las siguientes ilustraciones:

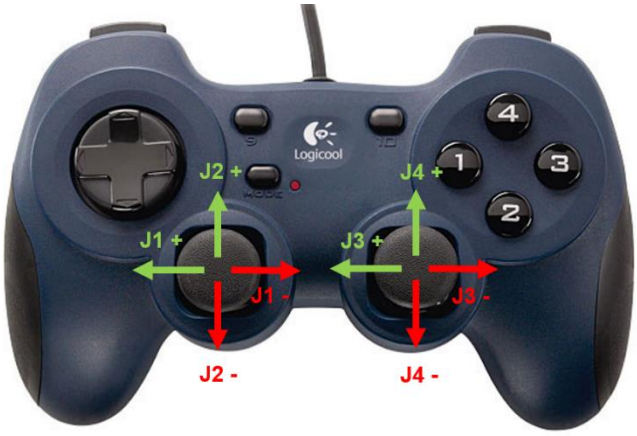


Ilustración 34: Relación entre movimientos del gamepad y movimientos del robot 1

Ilustración 35: Relación entre movimientos del gamepad y movimientos del robot 2

El convenio de signos utilizado es el siguiente:

- En los joysticks, los movimientos considerados como positivos hacen referencia a un giro antihorario de la articulación en cuestión, tomando como plano de referencia el alzado donde no están visibles los servos de mayor tamaño, como muestran las imágenes que se muestran seguidamente.
- En los triggers, para la articulación 5 el positivo indica un giro antihorario. Para la articulación 6 el positivo indica apertura de la pinza y el negativo el cierre de esta.

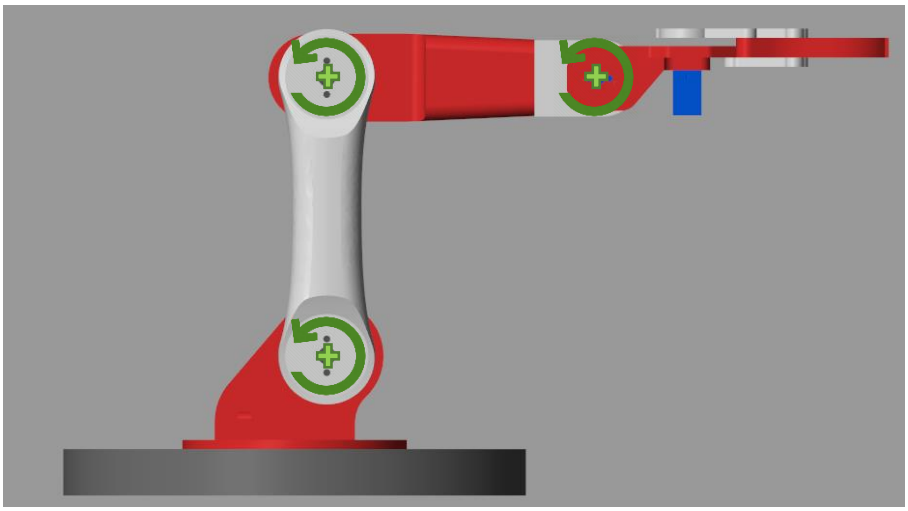


Ilustración 36: Convenio de signos y movimientos del robot 1

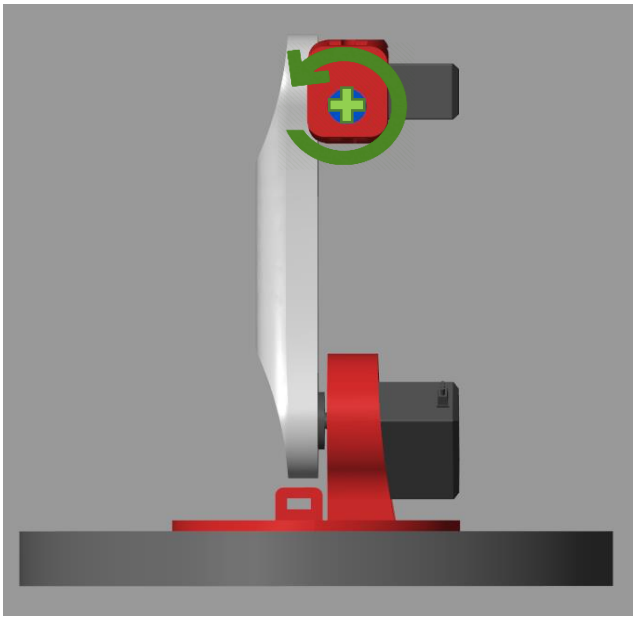


Ilustración 37: Convenio de signos y movimientos del robot 2

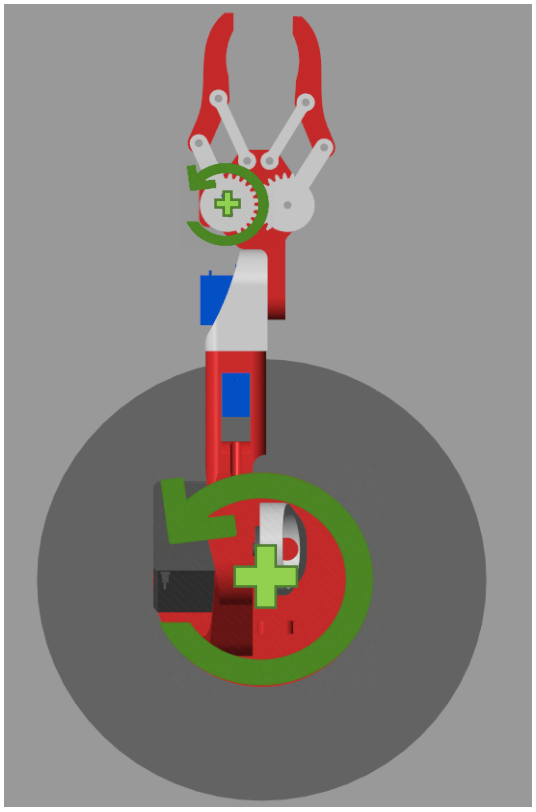


Ilustración 38: Convenio de signos y movimientos del robot 3

Una vez explicado cómo se realiza el manejo del brazo y por ende la generación de señales de referencia, se procede a mostrar cómo se integran dichas señales en el subsistema de entrada de cada articulación:

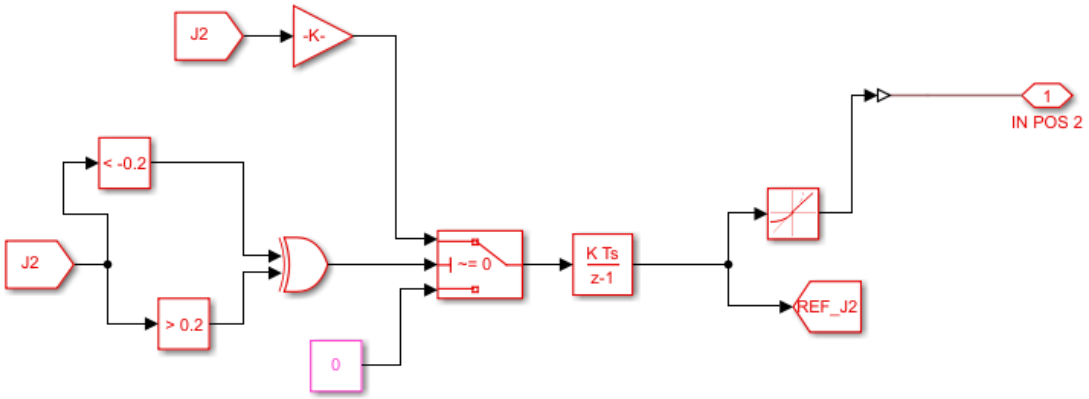


Ilustración 39: Integración de las salidas del gamepad como entradas del modelo

El funcionamiento del subsistema es sencillo. Lo primero se incluyen unos filtros para cancelar el ruido de la señal debido a que cuando el joystick vuelve a su posición central no devuelve un valor igual a 0 sino que tiene ligeras desviaciones. En caso de que la señal no supere el threshold de ±0.2, estando la salida del joystick comprendida entre ±1, se garantiza que no se envían órdenes no deseadas (se envía la constante 0). Si el valor supera el threshold impuesto, se envía la señal correspondiente. Esto se gestiona haciendo uso de la puerta XOR y el bloque switch previos al integrador. El integrador permite obtener funciones de movimiento en base a los valores emitidos por el joystick y almacenar el último valor en caso de que no se envíen nuevos. El bloque *Goto* llamado REF_JX (siendo X el número de cada articulación) lleva la señal de referencia al subsistema de visualización representado en la Ilustración 23: Subsistema OUT REV JOINT 1 expandido para su comparación con la posición real.

Queda explicado, por tanto cómo se hace la generación de trayectorias haciendo uso del *gamepad* o controlador así como la inserción de estas en nuestro modelo de simulación. El seguimiento será exactamente igual que en el apartado anterior, ya que la dinámica de los servos no ha cambiado y por tanto sigue siendo el componente limitador en este aspecto.

Para terminar este apartado, se presenta una gráfica tridimensional de la trayectoria seguida por la herramienta manipuladora del brazo, el punto medio entre las pinzas del Gripper, durante una simulación en la que se usaba el *gamepad* para controlar el brazo.

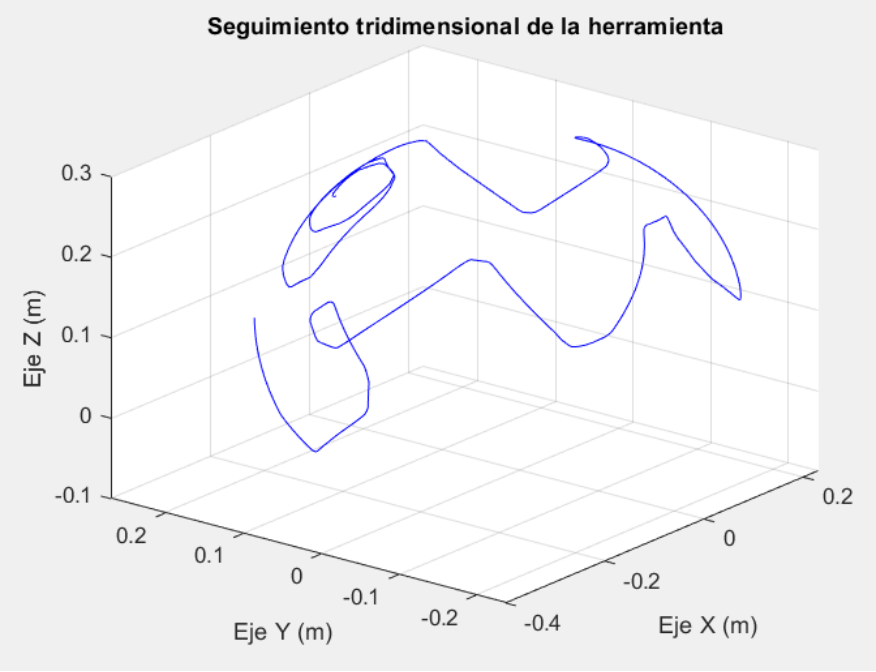


Ilustración 40: Seguimiento tridimensional de la herramienta durante movimiento manual

La figura representa una trayectoria al azar introducida en tiempo real usando el controlador, de ahí la aleatoriedad del movimiento. Si se compara con la trayectoria que sigue la herramienta a partir de unas trayectorias estudiadas y predefinidas usando la metodología del apartado 4.1.2 Simulación de trayectorias predefinidas, se puede apreciar la diferencia de forma.

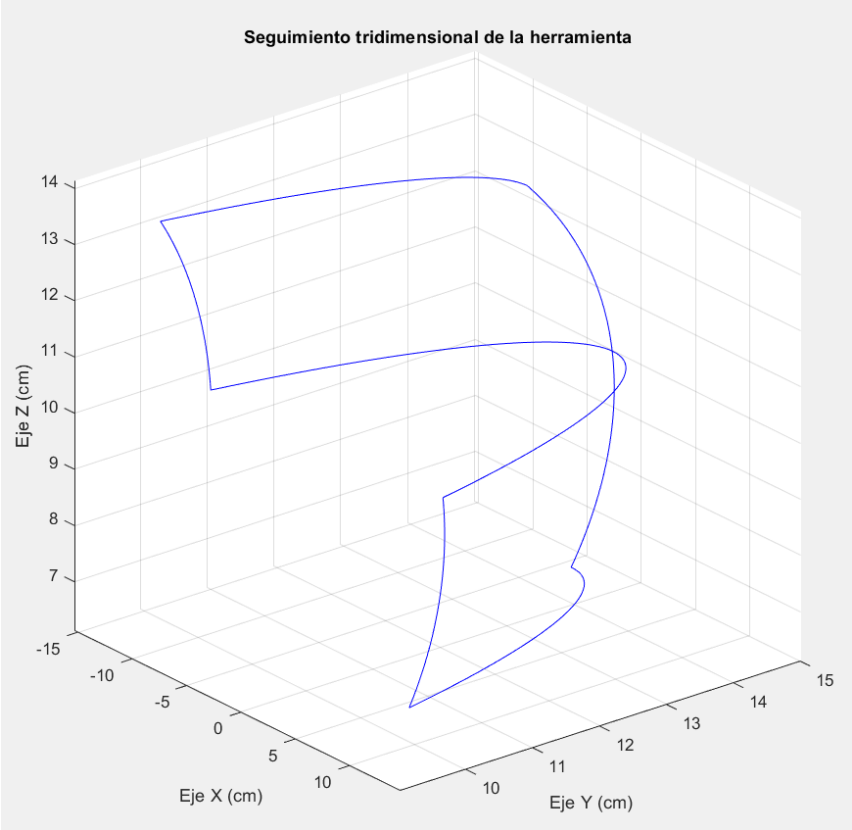


Ilustración 41: Seguimiento tridimensional de la herramienta usando trayectorias predefinidas

4.1.4 Pick & Place

El último de los pasos de este capítulo consiste en la simulación de un proceso de Pick & Place, que es la actividad que está destinada a realizar por el robot. Para ello, partiendo del modelo de la Ilustración 26: Modelo de simulación han de añadirse una serie de bloques que permitan la interacción entre cuerpos y relación entre las fuerzas que se dan durante el proceso.

La librería a utilizar para ello es *Simscape Multibody Contact Forces Library* y los cambios añadidos en el nuevo modelo de simulación se proceden a explicar haciendo uso de las imágenes que se muestran a continuación.

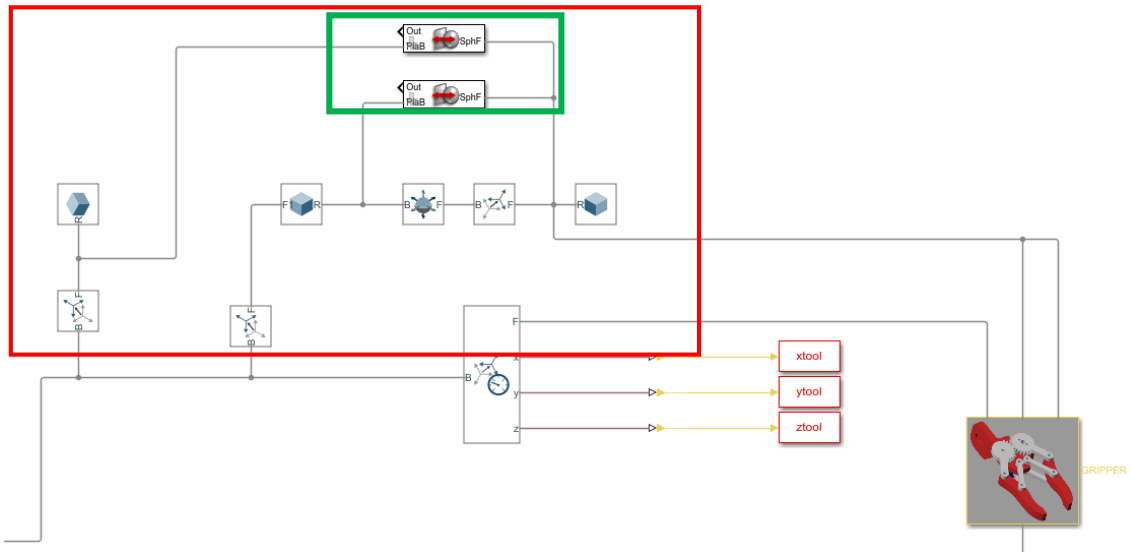


Ilustración 42: Bloques añadidos para la simulación de Pick & Place 1

El grupo de bloques señalados dentro del marco de color rojo son los añadidos en este apartado. Se ha decidido representarlos junto bloques ya existentes para que sea más fácil ubicarlos dentro del modelo.

Los bloques *Solid* situados más a la izquierda en la figura representan dos piezas hexaédricas que serán las bases donde se recoge y entrega de la pieza a manipular. Esta última se trata de una esfera, cuya posición respecto del punto de recogida se define a través de un *Rigid Transform* para que se encuentre justo sobre la base. También se define un tipo de *Joint* que otorga a la esfera libertad para moverse y girar a lo largo y alrededor de X,Y,Z (*6 DOF*) dentro del sistema de referencia *World*.

Estos bloques pertenecen a librerías ya utilizadas anteriormente, pero son igualmente importantes para el correcto funcionamiento de la simulación.

En lo que concierne directamente a la interacción entre sólidos se utilizan los bloques señalados por el marco verde. Estos sí forman parte de *Simscape Multibody Contact Forces Library*, en concreto se denominan *Sphere to Plane Force* y definen la relación de fuerzas que se produce en el contacto entre un plano (cara superior de la base hexaédrica) y una esfera. Para ello es necesario precisar parámetros como las dimensiones de la esfera y el plano, coeficientes de fricción estática y dinámica o coeficientes de amortiguamiento y rigidez del contacto.

Esta fase alberga especial dificultad por la poca información que hay al respecto así como por la íntima relación entre los parámetros mencionados que hace que un ligero cambio en uno de ellos altere el funcionamiento de todo el conjunto.

Una vez modelado el contacto entre la esfera y los puntos de recogida y entrega, se procede a realizar lo propio entre la esfera y las pinzas del Brazo Robot.

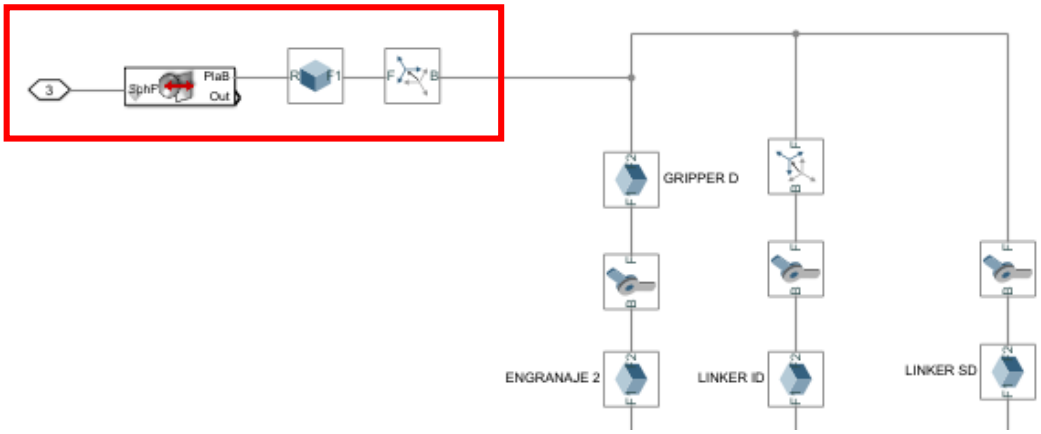


Ilustración 43: Bloques añadidos para la simulación de Pick & Place 2

Igual que en el caso anterior, se encuadran dentro de un marco rojo los bloques añadidos para el Pick & Place. Como se puede observar hay un bloque *Solid* unido a la pinza derecha denominada *Gripper D*, esto es debido a que la librería que se está utilizando sólo permite modelar un número limitado de tipologías de contacto, siendo estas entre figuras geométricas muy sencillas como pueden ser Plano-Esfera, Esfera-Esfera o Esfera-Cilindro. Esto no supone ningún problema en el caso anterior, pero si plantea un desafío en este, pues la morfología de la pinza no está contemplada en ningún bloque. De manera que para abordar el problema se ha unido en cada pinza (de manera análoga, se incluyen los mismos bloques en la pinza izquierda denominada *Gripper I*) una pieza hexaédrica auxiliar. La localización de la cara de contacto estará situada lo más superpuesta posible al área de la pinza destinada a la fricción con la esfera. Así, el contacto se está realizando entre la esfera y las superficies auxiliares, pero el aspecto será lo más parecido posible a un contacto entre la pinza y la esfera. Todos los parámetros relativos tanto a los bloques auxiliares (proporciones, densidad, etc.) como al bloque *Sphere to Plane Force* son iguales que si se hiciera el contacto directo con la pinza, por lo que el resultado final es prácticamente el mismo y el problema queda solventado. Cabe comentar que estas piezas auxiliares son ocultadas durante la simulación con objeto de mejorarla en lo que a aspecto visual se refiere.

Para finalizar este apartado, se muestran unas imágenes en las que se observan las superficies auxiliares del Gripper así como una serie de capturas del proceso de simulación de Pick & Place, cuyo vídeo se incluye en la memoria de este trabajo, en las que se pueden apreciar la interacción entre sólidos.

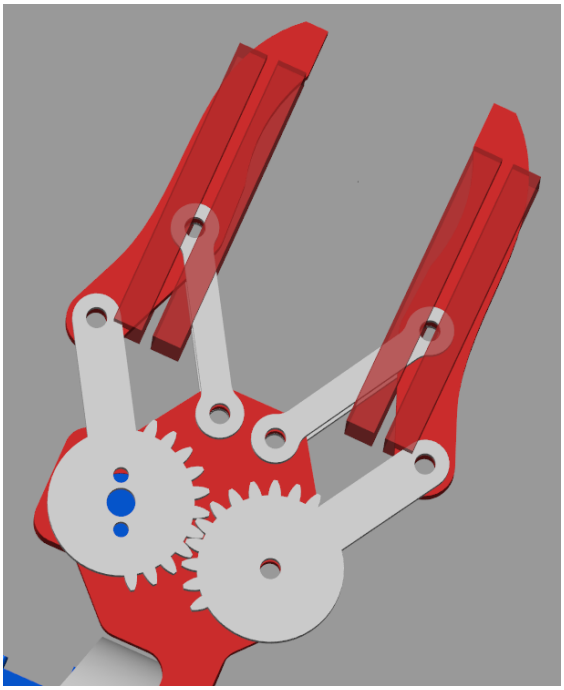


Ilustración 44: Localización y aspecto de las superficies auxiliares de contacto

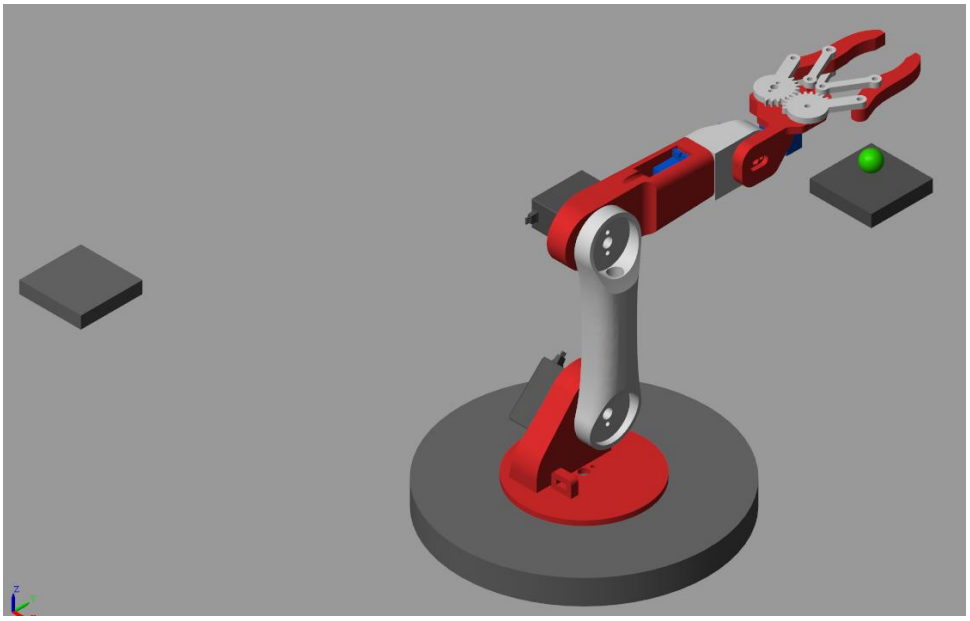


Ilustración 45: Captura de simulación de Pick & Place 1

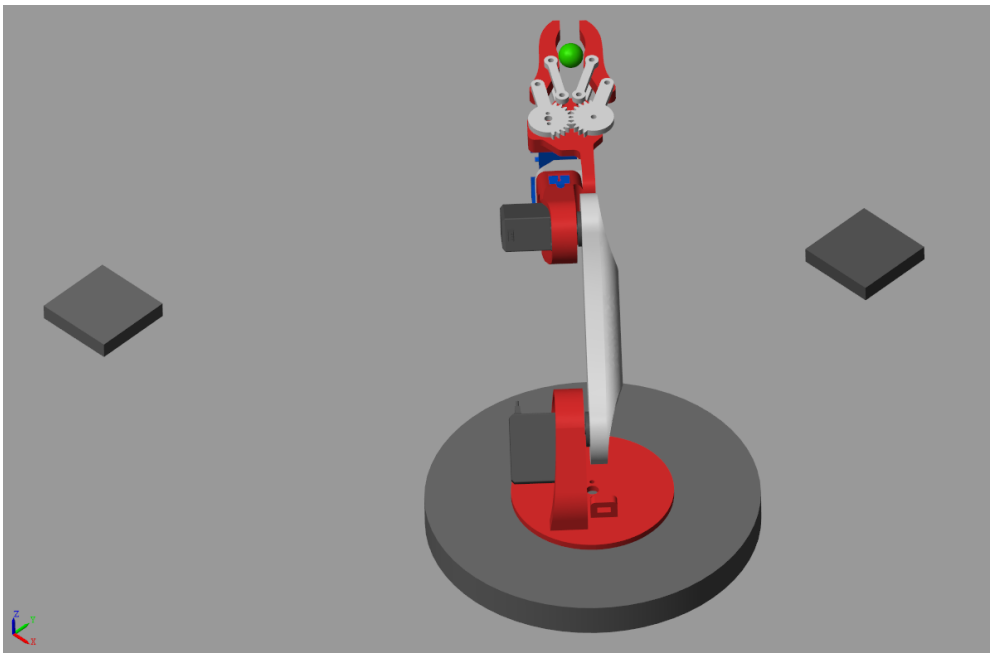


Ilustración 46: Captura de simulación de Pick & Place 2

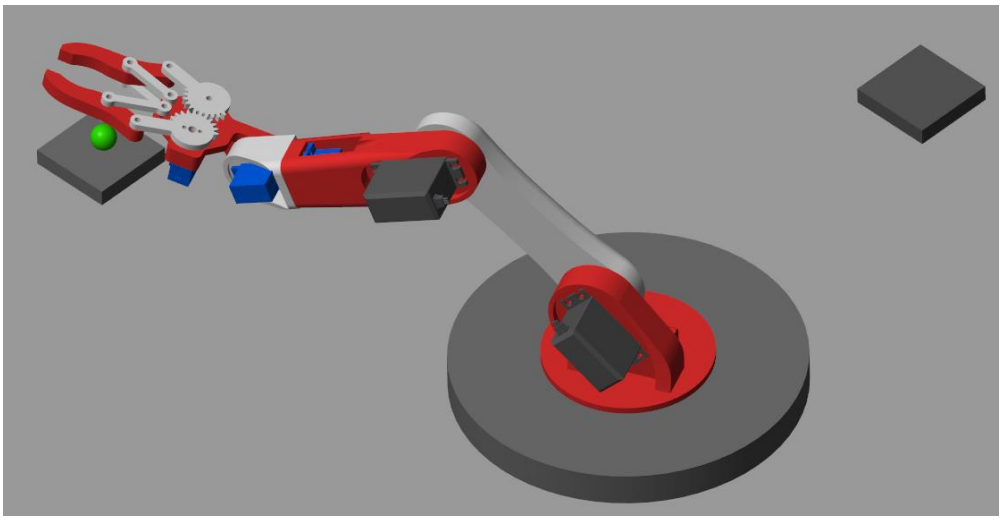


Ilustración 47: Captura de simulación de Pick & Place 3

4.2 Plataforma Omnidireccional

Esta sección será similar a la anterior, pues también se trata la concepción de un modelo de simulación de un sistema mecatrónico, siendo en este caso la plataforma omnidireccional sobre la que se sitúa el brazo robótico. Al final ambos modelos formarán uno solo, pero para la fase de diseño y puesta a punto se decide presentarlos de manera separada debido a la complejidad individual y la naturaleza completamente distinta de cada uno de ellos

4.2.1 Diseño del modelo de simulación

Al haberse explicado detalladamente los componentes y el entorno de simulación en la sección anterior relativa al Brazo Robot, se procede a explicar la creación de la Plataforma Omnidireccional de forma más directa y breve.

En primer lugar se define el suelo o superficie por la cual se traslada la plataforma y el núcleo de esta que es el chasis, así como las relaciones de movimiento que entre ellos existen. A partir de ahí siguiendo las metodologías de diseño explicadas anteriormente se va construyendo el modelo.

Como se puede ver en la imagen que sigue, con objeto de simplificar el aspecto, los bloques se encuentran dentro de diferentes subsistemas que hacen referencia a las partes principales que conforman la plataforma.

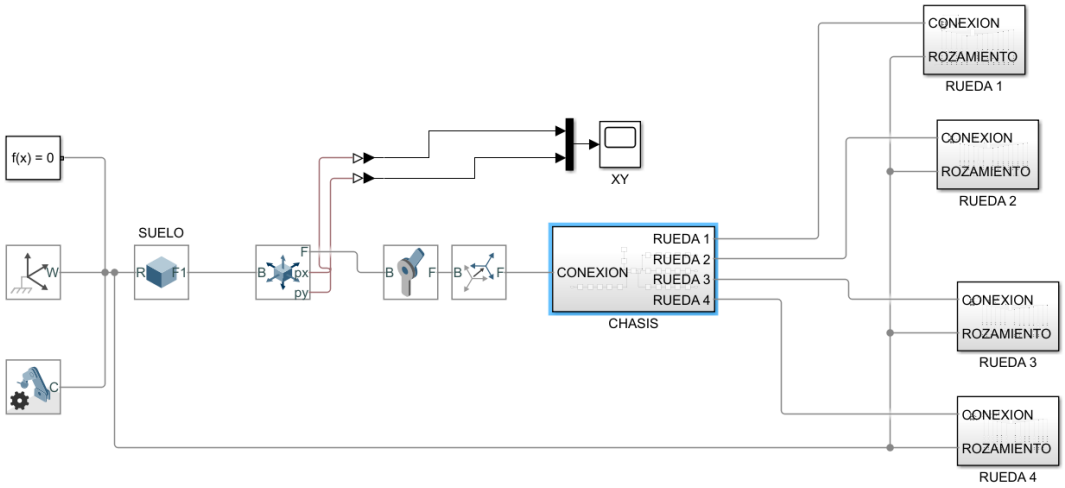


Ilustración 48: Modelo de simulación de la Plataforma Omnidireccional

El primero de ellos denominado chasis, se desarrolla alrededor de un sólido con el mismo nombre que hace las veces de alojamiento para los motores y acoples de cada una de las ruedas. Así como de cualquier otro elemento que se quisiera incluir en su interior cuya influencia se quisiera tener en cuenta durante la simulación, siempre habiendo estudiado previamente su viabilidad.

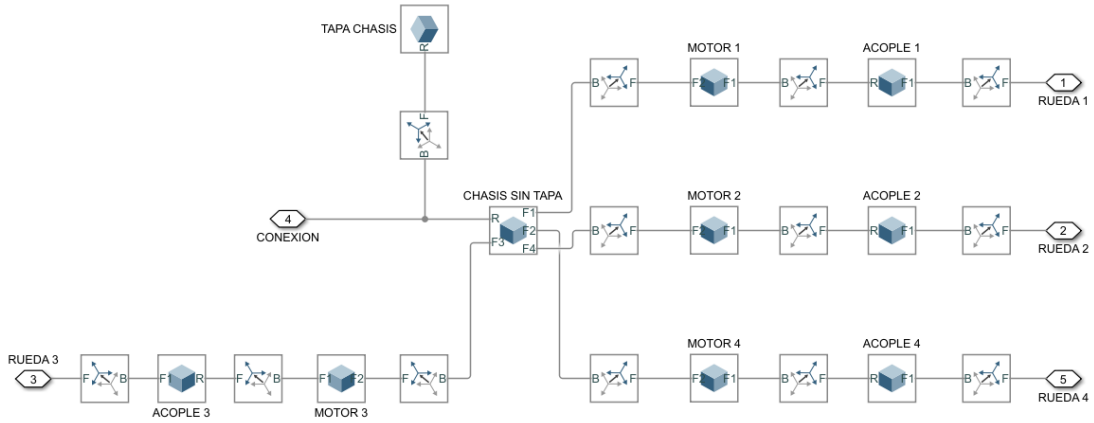


Ilustración 49: Subsistema CHASIS

Para cada rueda, se utiliza un subsistema cuyo diseño es idéntico, aunque en la fase de control posterior se incluirán diferencias fundamentales. Al tratarse de ruedas omnidireccionales revisten mayor complejidad que las ruedas empleadas habitualmente, por ello se dividen en Cuerpo Rueda, que es aquella parte de la rueda que gira solidaria a su eje y actúa de armazón de esta; y los diferentes rodillos que la confieren la capacidad de moverse en diferentes direcciones como la paralela al eje motriz de la rueda. La unión entre el primer elemento y cada uno de los segundos mencionados se lleva a cabo con una articulación de revolución.

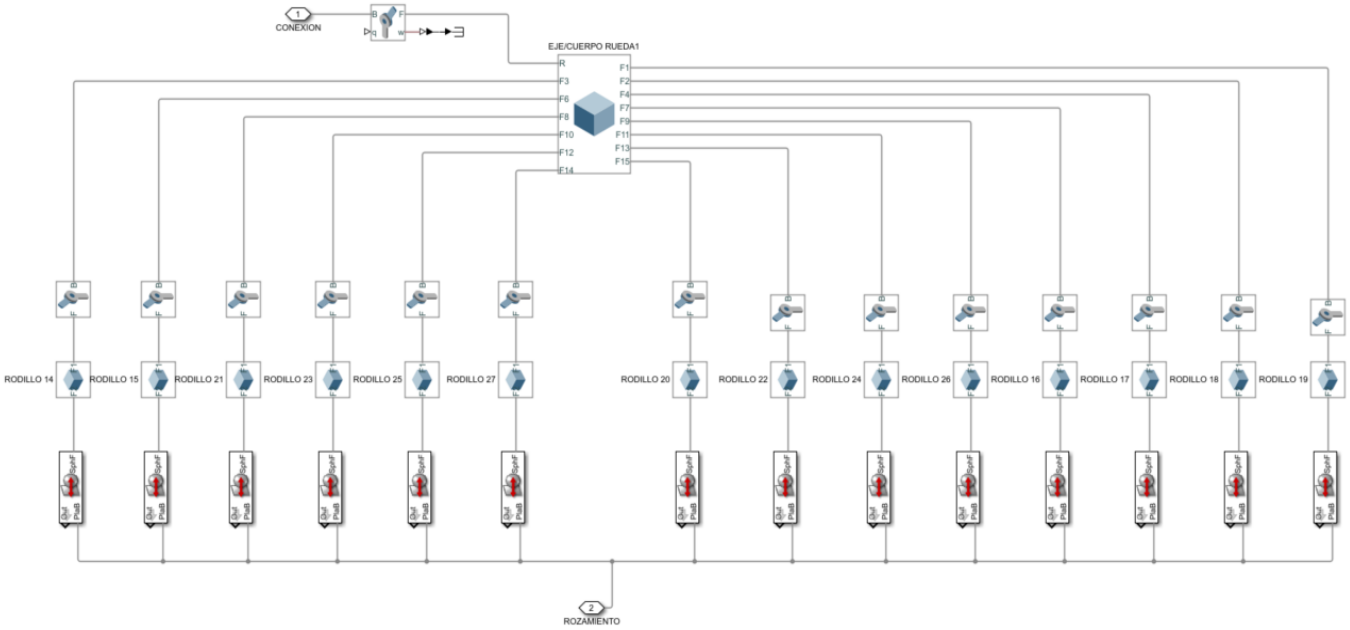


Ilustración 50: Subsistema RUEDA 1

También está modelado el contacto de dichos rodillos con el bloque Suelo, usando los bloques de la librería *Simscape Multibody Contact Forces* utilizada en 4.1.4 Pick & Place.

En cuanto a los puertos de conexión que tiene el subsistema con el sistema principal, el denominado Conexión es para la unión física de la Rueda con el Chasis mientras que Rozamiento une el puerto *PlaB* de cada uno de los bloques de contacto con el bloque Suelo.

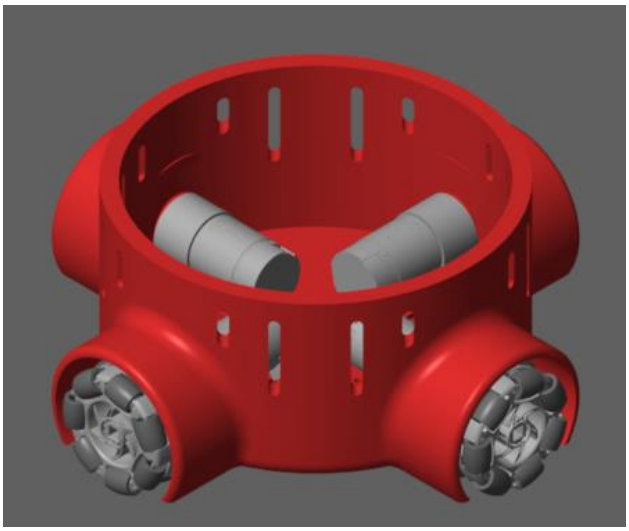


Ilustración 51: Aspecto del conjunto Plataforma Omnidireccional (sin la tapa)

4.2.2 Simulación de trayectorias

Para la plataforma, la intención es que a partir de una trayectoria introducida al sistema, este la realice de la manera más fiel posible. Para ello, antes de pensar en trayectorias complejas lo primero es conseguir llegar con el menor error posible a una coordenada concreta. Este primer paso es clave pues en caso de hacerse bien, nos garantiza una base sólida sobre la que ir añadiendo nuevos puntos y en última instancia el seguimiento fiable durante un recorrido.

Para ello conviene recordar lo expuesto en 2 Estudio teórico, prestando especial atención a las fórmulas que se presentan al final del capítulo. Según se ha explicado, a partir de la velocidad lineal en X (Vx), la velocidad lineal en Y (Vy) y la velocidad angular (ω) deseadas, se obtienen mediante las expresiones citadas la velocidad angular de cada una de las ruedas que componen la plataforma. Esa es la premisa básica sobre la cual se basa el control a diseñar. Queda por tanto, dilucidar cómo saber qué valores de Vx, Vy y ω permiten ir de la manera más inmediata del punto de salida al de destino.

Para ello, se parte de la idea de que para llegar de un punto a otro en línea recta, se debe recorrer una trayectoria cuya dirección está definida por un vector. Este vector dirección vendrá determinado por el ángulo de la recta que une ambos puntos entre sí con la horizontal. Sabiendo la distancia en X (Δx) e Y (Δy), de un punto a otro, este ángulo puede conocerse haciendo uso de la función inversa de la tangente entre ambas deltas, pues son las longitudes de los dos catetos que definen el triángulo rectángulo cuya hipotenusa es el módulo del vector dirección que une ambos puntos en línea recta. Esto quedaría de la siguiente forma:

$$\beta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)$$

Conociendo dicho ángulo β, utilizando las funciones trigonométricas del coseno y el seno se conocen de manera directa las componentes en X e Y respectivamente, normalizadas entre [-1,1]. Estos valores serán los utilizados como las referencias de velocidad lineal, escalándolos antes entre [-0.5, 0.5] con el fin de que se adecúen más a las velocidades de funcionamiento real, limitadas por el material empleado.

Este proceso, queda implementado en el modelo de Simulink de la siguiente forma:

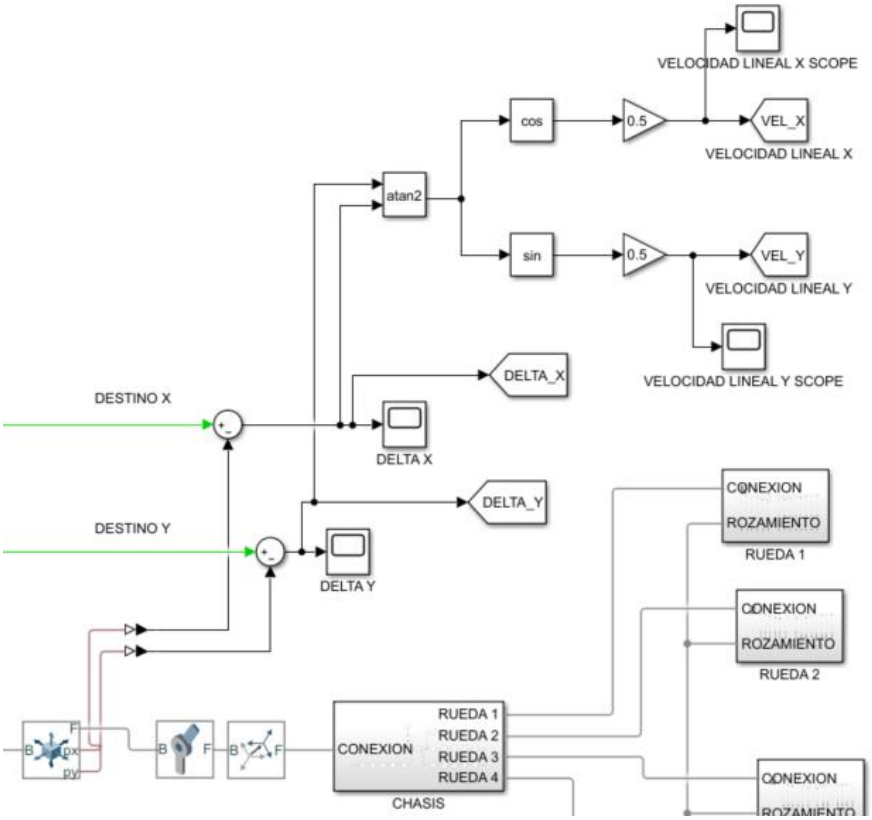


Ilustración 52: Estructura que controla las velocidades en X e Y

Como se puede apreciar, se calcula la diferencia entre la posición actual de la plataforma en X e Y, medida usando las opciones de sensórica del bloque *Cartesian Joint,* y las coordenadas X e Y del punto de destino. Usando la opción *atan2* del bloque *Trigonometric Function* se calcula el ángulo cuya tangente se obtiene con los valores introducidos de dichas diferencias DELTA X y DELTA Y. Aplicando otros dos bloques *Trigonometric Function* eligiendo la función *cos* en uno y la función *sin* en otro, se obtienen las componentes de velocidad lineal en X e Y que se introducen en dos bloques *Goto* denominados VEL_X y VEL_Y respectivamente.

Estas componentes son recalculadas a medida que avanza el vehículo, pues como se ha visto para obtenerlas se utiliza la posición actual de este. Esto nos garantiza que la plataforma corregirá su dirección de movimiento frente a perturbaciones con el objetivo de alcanzar su destino. Pero, ¿qué hará la plataforma cuándo alcance dicho destino?, ¿se quedará orbitando a su alrededor cambiando de dirección constantemente? El control de transición de un punto a otro se tratará con más profundidad posteriormente en este capítulo, lo que se ha hecho durante la puesta a punto con un único punto de destino es introducir una referencia de velocidad nula cuando el módulo del vector cuyas componentes son DELTA X y DELTA Y es inferior a 0.05. En esta primera fase se han ajustado los parámetros relativos al contacto entre superficies hasta dar con valores que minimizan a niveles aceptables el deslizamiento de la plataforma desde que se paran los motores de las ruedas hasta que se para ella.

Habiendo explicado la obtención de las velocidades lineales, queda ver cómo se ha implementado su relación con las velocidades de las ruedas, en base a las expresiones mencionadas al inicio de este apartado y que, de nuevo, se pueden encontrar al final de 2 Estudio teórico.

Para el control de las ruedas, se reciben los valores de velocidad en X e Y de los bloques *Goto* VEL_X y VEL_Y mencionados anteriormente, usando bloques *From* referenciados a ellos. En el caso de la velocidad angular, interesa que sea nula, es decir que la plataforma no gire alrededor de su eje z durante el movimiento, pues no es necesario que lo haga gracias a su capacidad omnidireccional.

Al tener las expresiones que nos permiten obtener de forma directa las velocidades angulares de las ruedas ω_i (i=1,2,3,4), se implementan dichas operaciones en el modelo, de manera que se obtiene el valor para cada una de ellas. Este valor se integra, pues la entrada a la articulación se hace por *Motion,* que indica la posición de rueda. Todo esto queda representado en la imagen que se presenta a continuación.

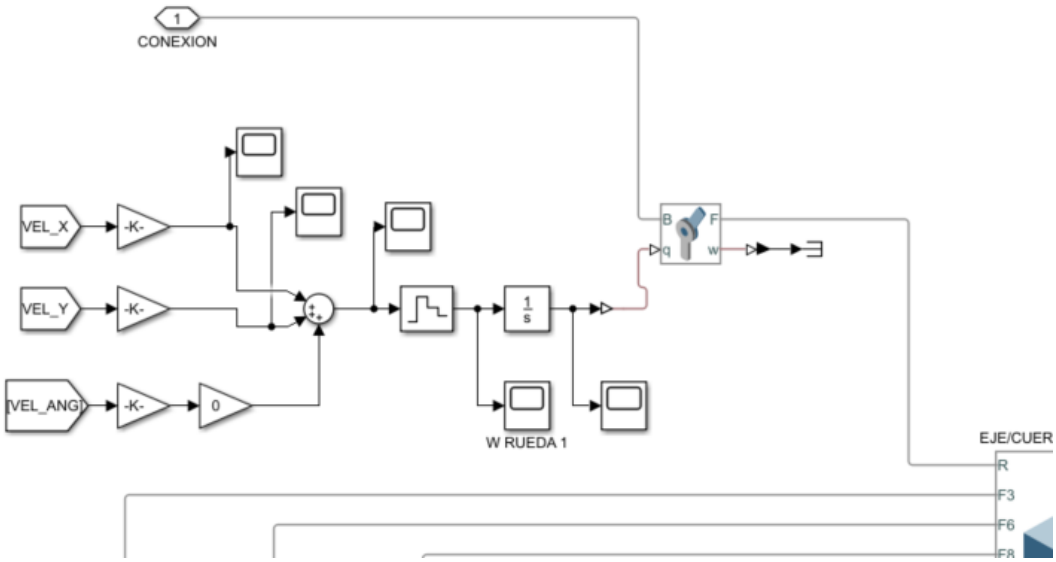


Ilustración 53: Sistema de control de las ruedas

Habiendo visto ya cómo se realiza el control de posición de la plataforma queda únicamente escalar el desplazamiento de un punto a otro al seguimiento de una trayectoria. Una trayectoria, en esencia, es una serie de puntos de paso cuyo orden está establecido. Por lo que lo que hay que hacer, es definir los puntos de la trayectoria que se quiere seguir y que la plataforma los vaya siguiendo de manera ordenada.

Para ello se diseña un sistema de control que en base a dos *arrays* de valores, posiciones de destino en X (POSX) y posiciones de destino en Y (POSY), envía un nuevo punto de destino cuando la plataforma ha llegado al destino actual. Para establecer que la plataforma ha llegado al punto de destino y sustituirlo por el siguiente se evalúa el módulo del vector (Vvd) que une el punto situado en el centro de la plafatorma (Xv,Yv) con el punto de destino actual (Xd,Yd). Si es menor a una constante, que equivale al radio de la rueda de la plataforma, se establece un nuevo punto de destino. En el momento en el que se alcanza el último punto del recorrido la simulación se termina o vuelve a iniciarse, en función de la opción elegida.

Lo expuesto en el párrafo anterior se puede ver implementado en las imágenes que siguen, en la primera de las cuales se muestra también parte del modelo para que al lector le sea más fácil identificar la posición de los nuevos bloques y su función.

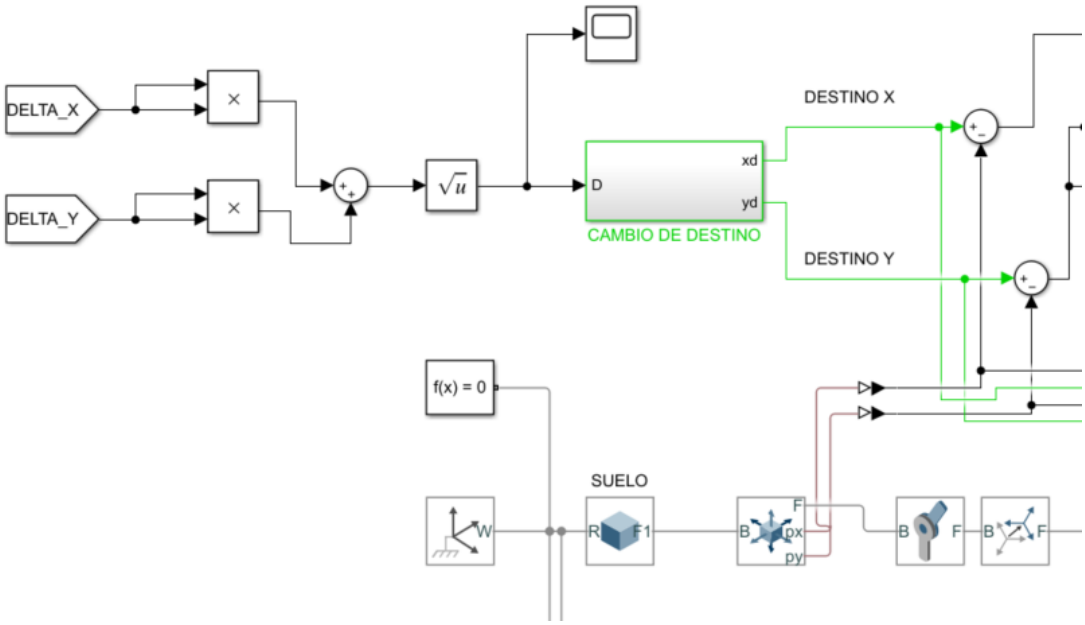


Ilustración 54: Sistema de control para el cambio de destino

El módulo del vector Vvd es la entrada al subsistema de CAMBIO DE DESTINO y sus salidas son los puntos destino Xd e Yd. Es dentro del subsistema dónde se compara el módulo citado y en función del resultado, avanza posición dentro de los *arrays* POSX y POSY. También incluye la terminación de la simulación al llegar a la última referencia así como la opción de recorrer la trayectoria una sola vez o de manera cíclica como se ha citado antes.

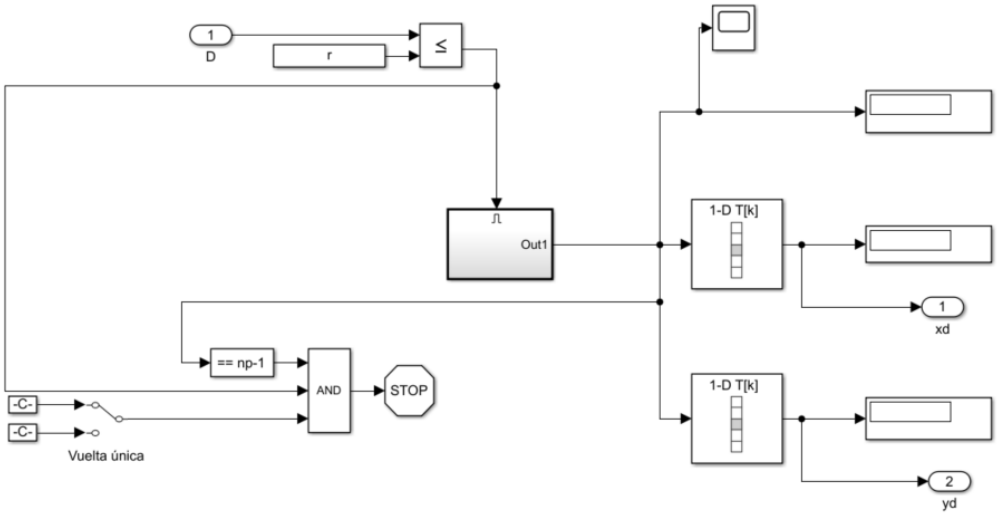


Ilustración 55: Subsistema de cambio de destino

Para la introducción de los vectores de posición POSX y POXY se utiliza la consola de comandos o un script de Matlab. Tras una serie de pruebas de trayectorias sencillas como líneas diagonales o cuadrados con resultados positivos se decide realizar el seguimiento de una trayectoria de mayor complejidad. Los resultados de la simulación correspondiente serán los que se presenten al final de este apartado, pues se considera que son los que más interés revisten.

Las trayectorias escogidas son Curvas de Lissajous, que son la gráfica del sistema de ecuaciones paramétricas correspondiente a la superposición de dos movimientos armónicos simples (MAS) en direcciones perpendiculares. Se pueden obtener diferentes formas en función del desfase que existe entre ambos movimientos y de la relación entre sus frecuencias.

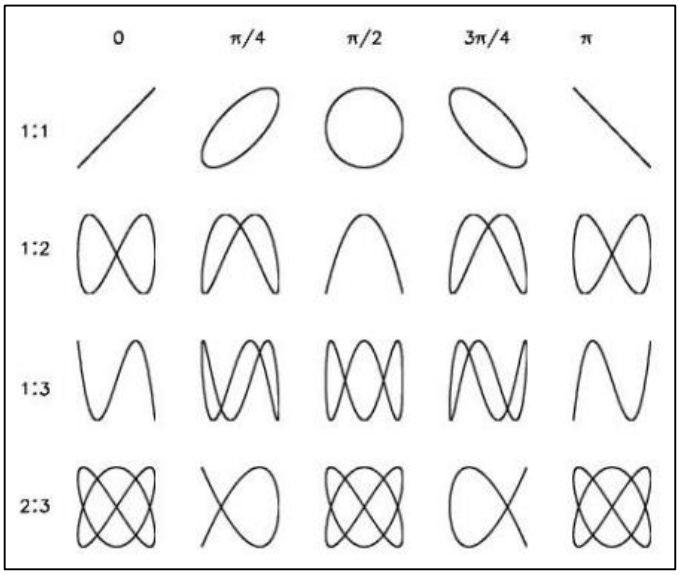


Ilustración 56: Curvas de Lissajous

Se ha creado un script denominado *TrayGen.m* pensado para ser ejecutado antes de la simulación que, además de permitir establecer los valores de todos aquellos parámetros relativos a la simulación, genera una curva de Lissajous en función de las frecuencias introducidas para cada movimiento armónico simple (siempre con un desfase nulo). Una vez creada la curva se obtienen a partir de ella las coordenadas en X e Y en cada periodo de muestreo, el cual también se define en el propio script, y se guardan en los vectores de posición POSX y POSY. Una vez establecidos todos los parámetros de simulación sólo resta ejecutarla.

```
%Definimos los valores de simulación
T=0.001; %Periodo de muestreo de la simulación
kf=4; %Coeficiente de fricción cinemática
sf=7; %Coeficiente de fricción estática
dc=10; %Coeficiente de amortiguamiento
vt=0.001; %Velocity Threshold
r=0.035; %Constante evaluada
t=[0:0.001:1]; %Periodo de muestreo de la curva

%Generamos la curva de Lissajous
f1=2; %Frecuencia de MAS1
f2=3; %Frecuencia de MAS1
liss=plot(sin(2*pi*f1*t),sin(2*pi*f2*t),'.'); %Generación de la Curva

%Obtenemos los puntos de dicha curva
h=findobj(gca,'Type','line');
X=get(h,'Xdata');
Y=get(h,'Ydata');
np=numel(X);
```

Ilustración 57: Script para la generación de trayectorias

Habiéndose mostrado todo lo necesario para llevar a cabo la simulación de la trayectoria, se procede a presentar y analizar los resultados obtenidos. La trayectoria seguida es la curva de Lissajous resultante de dos MAS con una relación de frecuencias de 2/3, el cual se puede observar en la esquina inferior izquierda de la Ilustración 56: Curvas de Lissajous.

En primer lugar se muestran las gráficas que comparan la trayectoria seguida por la plataforma con los valores de los vectores de referencia POSX y POSY para dar una idea de la precisión del resultado.

Viendo las imágenes que se presentan en la página siguiente se observa como el seguimiento a simple vista es muy bueno, pues ambas gráficas están prácticamente solapadas. Con el fin de corroborar las buenas impresiones iniciales se aplica zoom a la segunda de ellas en un periodo desfavorable de la simulación, esto es, cuando se produce un cambio brusco de referencia. Esto se puede apreciar en la tercera foto que sigue a este párrafo.

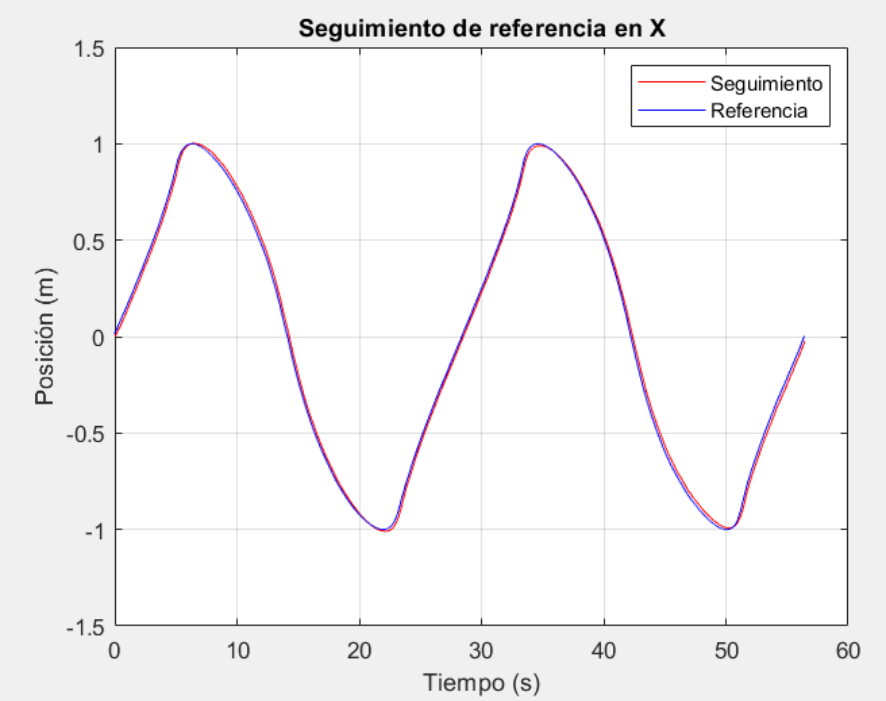


Ilustración 58: Seguimiento de la trayectoria referencia en X

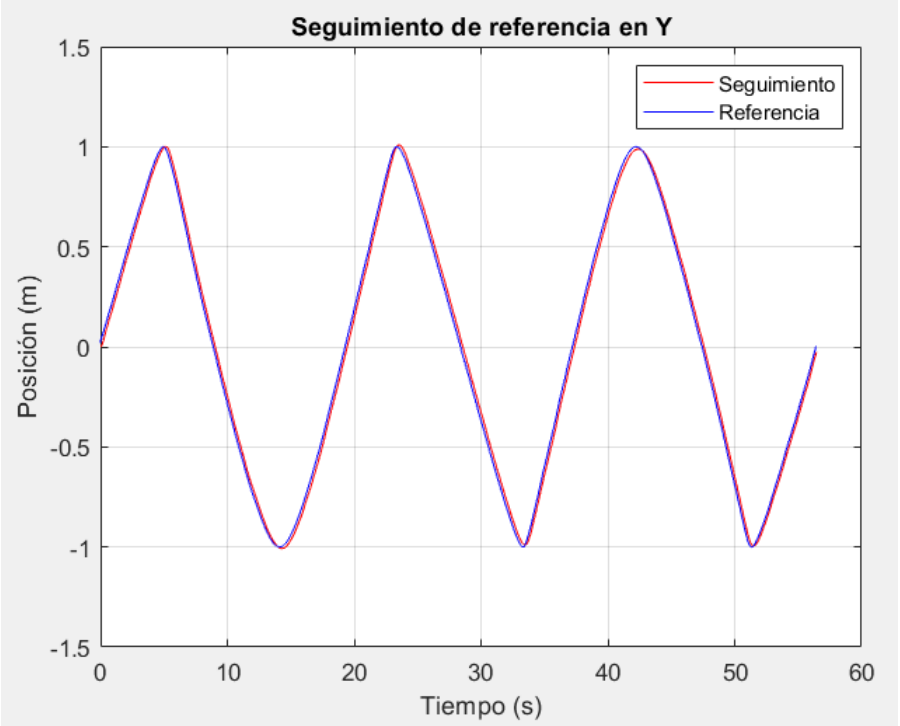


Ilustración 59: Seguimiento de la trayectoria referencia en Y

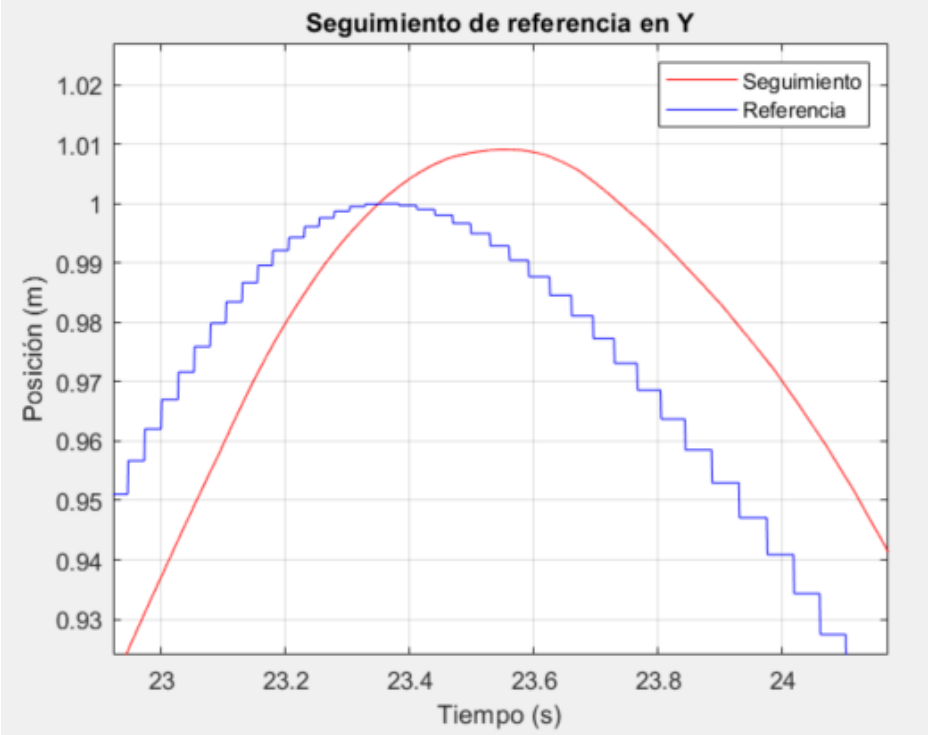


Ilustración 60: Detalle seguimiento de referencia en Y

Se aprecia que en uno de los puntos críticos de la simulación, la desviación de posición máxima es de 0.03 m. Por su magnitud y el hecho de que se ajusta al criterio establecido de paso de un punto de destino a otro, ya que es menor en todo momento al radio de la rueda de la plataforma (0.035 m), se considera que no es un error significativo y no altera el buen funcionamiento del sistema. En cuanto al retardo, los valores oscilan entre 0.1 y 0.2 segundos, esto responde al deslizamiento inicial unido al tiempo de respuesta por parte del sistema. De nuevo, no supone un problema para la obtención de buenos resultados.

Viendo que el seguimiento de las coordenadas en X e Y por separado es correcto, se procede a mostrar el seguimiento en XY, que es el más representativo e intuitivo pero que aúna los errores de los dos anteriores, por lo que este paso previo era necesario, ya que si el seguimiento por separado de las referencias no fuera acertado, esto sería aún más visible en el seguimiento XY.

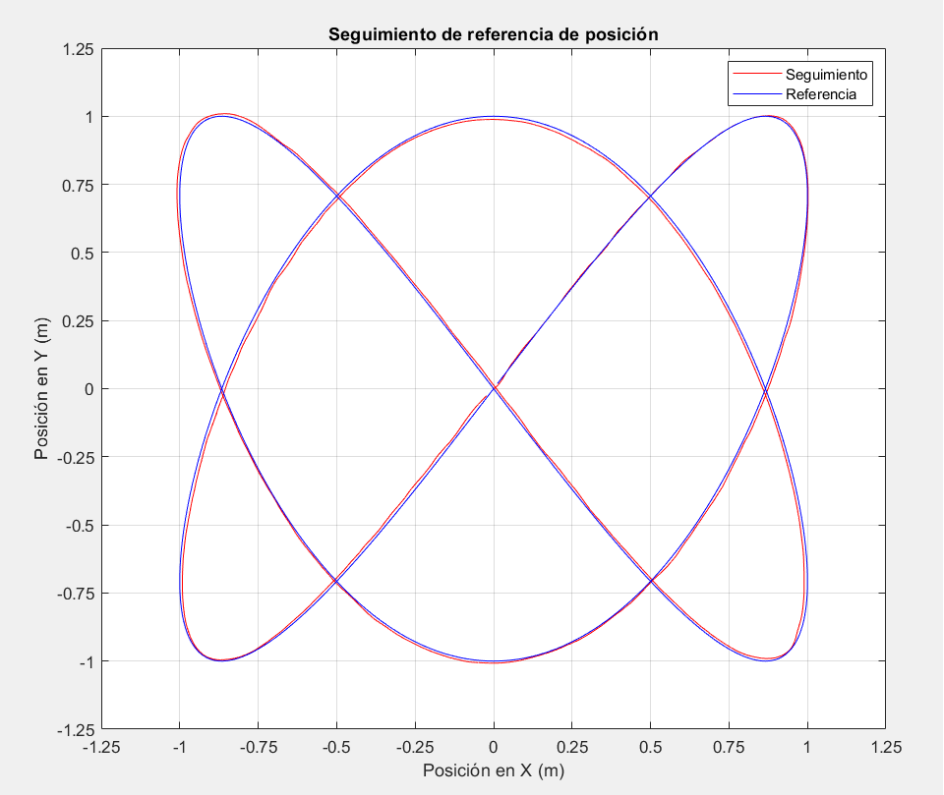


Ilustración 61: Seguimiento de referencia de posición

A la vista del seguimiento de la referencia de posición llevado a cabo por el vehículo durante la simulación se concluye que el sistema mecatrónico de la Plataforma Omnidireccional lleva a cabo su función de manera correcta y fiable.

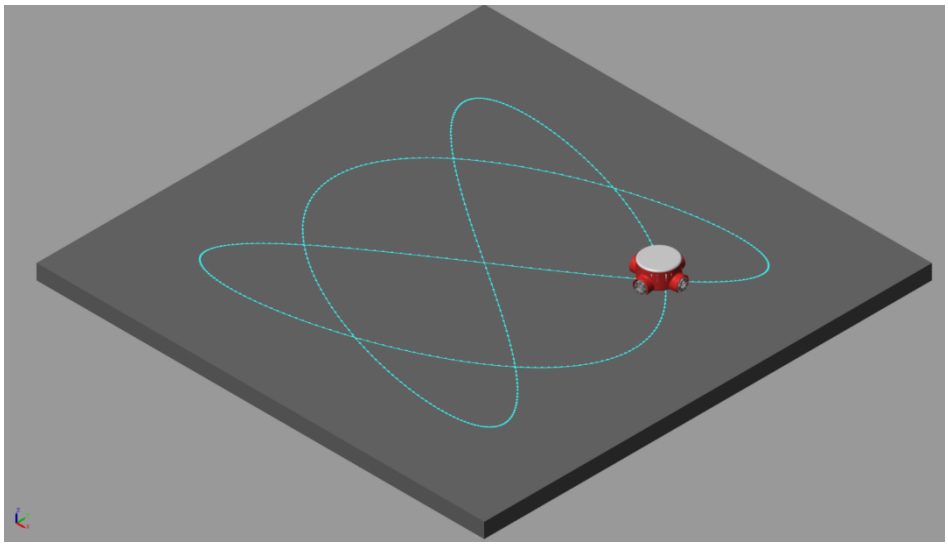


Ilustración 62: Captura de simulación de la Plataforma Omnidireccional

4.3 Manipulador móvil

En este apartado se combinan los dos modelos anteriores para obtener el modelo de simulación de lo que sería el producto final que se quiere conseguir. A continuación se presenta el aspecto del modelo final, ya con las diferentes estructuras explicadas en los apartados anteriores compactadas en subsistemas con el fin de simplificar el aspecto y facilitar la localización de cada parte del sistema.

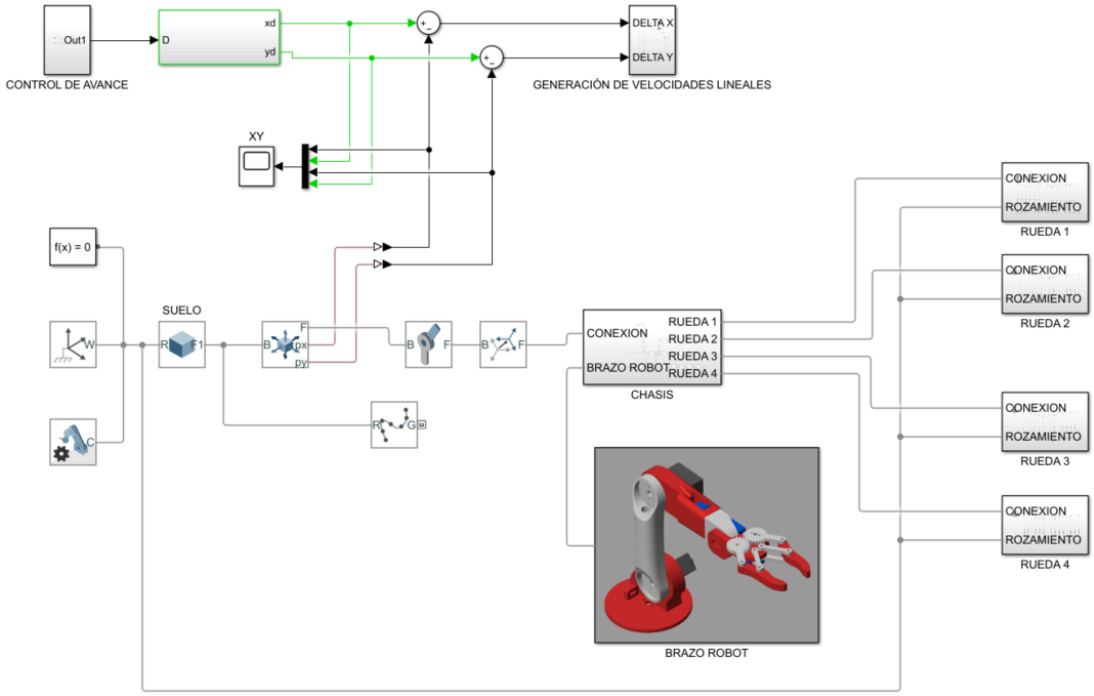


Ilustración 63: Modelo de simulación del Manipulador Móvil

El archivo de Simulink resultante, al tratarse de la unión de dos modelos que ya de por si albergan cierta complejidad, supone un elevadísimo coste computacional al ser simulado. Esto sucede pese a la simplificación de varias piezas e incluso de su eliminación, pues el mínimo número de componentes para garantizar la integridad del modelo siguen suponiendo una cuantiosa cantidad de información para procesar, por lo que deciden mantenerse los componentes originales.

Pese a ello se trata de simular un procedimiento de Pick & Place como el desarrollado en 4.1.4 Pick & Place pero de todo el sistema. Desgraciadamente la inmensa cantidad de tiempo requerida por el ordenador para procesar tal acción obliga a llevar a cabo una prueba menos exigente desde el punto de vista computacional. Esto no supone mayor contratiempo, pues el principal motivo de llevar a cabo dicha simulación era el aspecto visual, habiéndose comprobado ya la viabilidad del sistema para su implementación. Debido a esto, se decide realizar la simulación de un proceso de Pick & Place en el que el Manipulador Móvil realice todas aquellas acciones que realizaría en la realidad pero sin incluir la pieza a mover, reduciendo así el coste computacional de la simulación y facilitando el ajuste de los parámetros involucrados. Dicha

simulación, es uno de los archivos que se entregarán junto a este documento como memoria del proyecto. Se presentan como punto final a este capítulo, una serie de capturas del proceso mencionado.

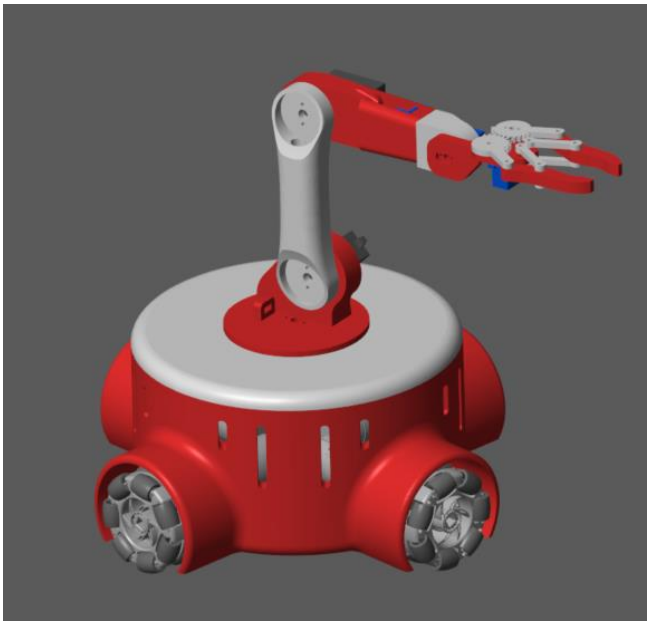


Ilustración 64: Aspecto final del Manipulador Móvil simulado

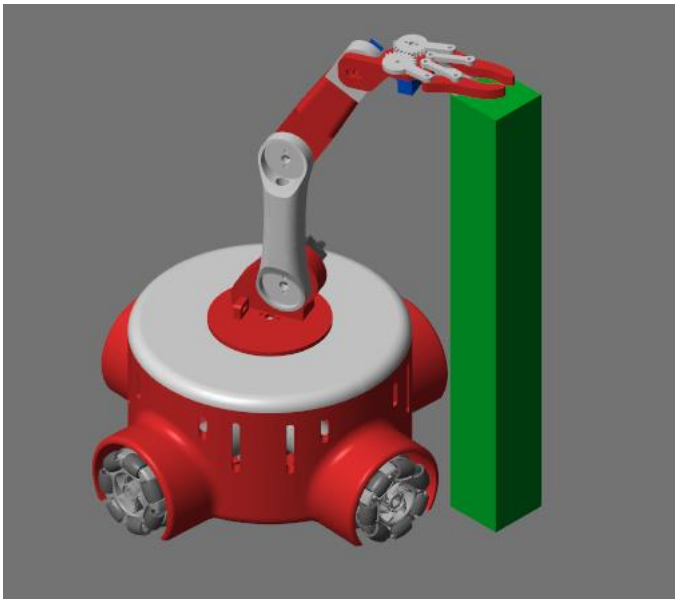


Ilustración 65: Manipulador móvil realizando la tarea de Pick

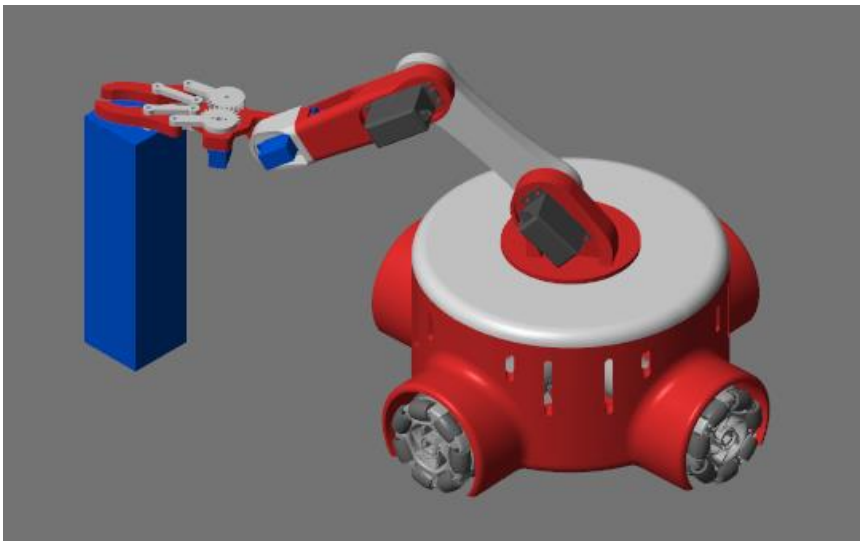


Ilustración 66: Manipulador móvil realizando la labor de Place

5 IMPLEMENTACIÓN REAL

Una vez finalizada de manera satisfactoria la fase de simulación y habiendo comprobado a través de esta que los sistemas en desarrollo son robustos y tendrán un correcto funcionamiento se procede a su implementación en el mundo real.

Como se ha explicado en 1 Introducción y objetivos la idea era realizar la implementación de ambos sistemas mecatrónicos. Por desgracia y ante la alteración del curso de los acontecimientos a nivel mundial debido al Coronavirus sólo ha sido posible implementar el Brazo Robot.

5.1 Materiales utilizados

5.1.1 Piezas de Impresión 3D

Los diferentes eslabones que conforman el Brazo Robot y que se presentan de manera detallada en 3.1 Brazo Robot se construyen haciendo uso de la impresión aditiva. Usando los archivos STL generados a partir del formato STEP utilizado para el proceso de simulación, se imprimen en el Laboratorio de Impresión 3D localizado en la ETSID. Así, análogamente a lo expuesto en el apartado mencionado, se muestran imágenes de los eslabones en cuestión.

Ilustración 67: Base giratoria Impresión 3D

Ilustración 68: Brazo 1 Impresión 3D

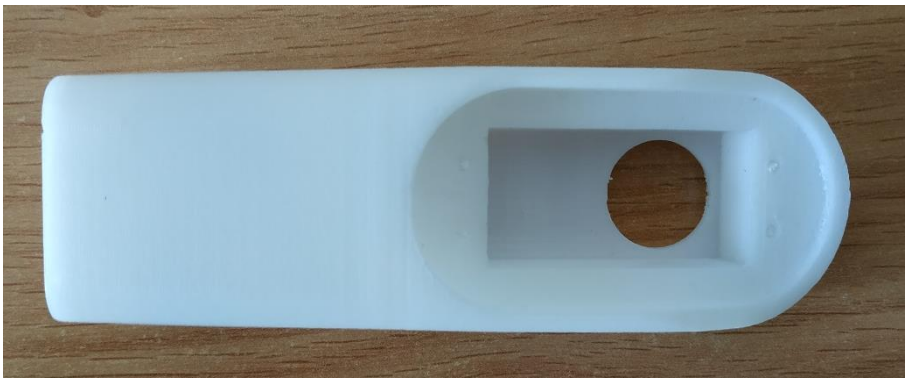


Ilustración 69: Brazo 2 Impresión 3D (1)

Ilustración 70: Brazo 2 Impresión 3D (2)

Ilustración 71: Engranajes del Gripper Impresión 3D

Ilustración 72: Brazo 3 Impresión 3D

Ilustración 73: Base del Gripper Impresión 3D

Ilustración 74: Enlaces y pinzas del Gripper Impresión 3D

5.1.2 Servomotores

Los servomotores utilizados son el MG996R para las tres primeras articulaciones y el SG90 para las tres últimas, como se ha mencionado en anteriores apartados. Los servomotores empleados para este tipo de proyectos tienen infinidad de fabricantes pero los modelos son limitados. Y en el caso de los servomotores utilizados, son dos de los más comunes a la hora de realizar proyectos mecatrónicos de esta escala. Por ello la elección se hizo, en una primera fase, haciendo una estimación en función de la experiencia tanto del autor de esta memoria como del tutor de este trabajo basándose en otros proyectos en los que se requería de la intervención de servomotores.

Una vez completado el modelo de simulación se llevaron a cabo varias simulaciones de lo que sería el funcionamiento real del robot y se midió, haciendo uso de las opciones de medición del bloque *Joint,* el par necesario en cada uno de los movimientos. Así, se comprobó que el par máximo que es capaz de proporcionar cada uno de los tipos de servos, no se veía excedido en ningún momento. De esta forma se concluyó, que los servos seleccionados eran apropiados para la aplicación en cuestión y se procedió a su adquisición.

La conexión con el 5.1.3 Servo Driver PWM PCA9685 se realiza uniendo cada uno de los tres cables que tiene cada servo (Naranja – Señal; Rojo – VCC; Negro - GND) en una de las salidas del PCA9685, que posee el mismo código de colores a excepción del naranja, para el cual usa el color amarillo.

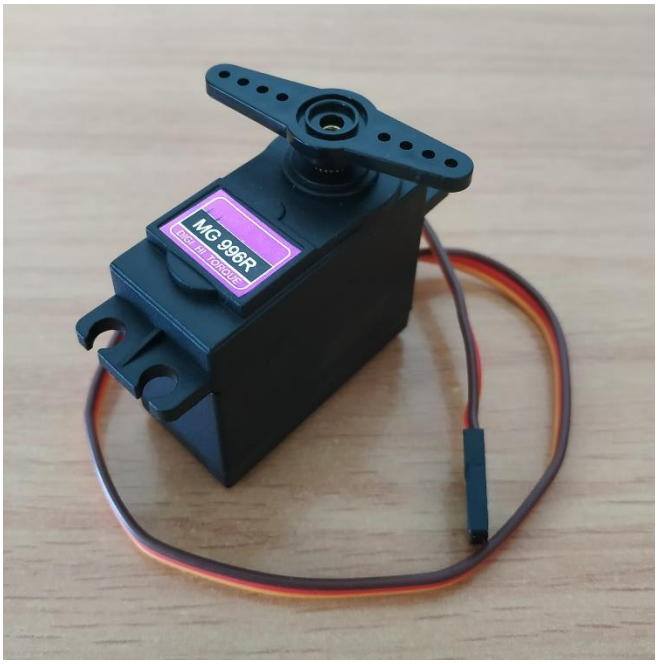


Ilustración 75: Servomotor MG996R

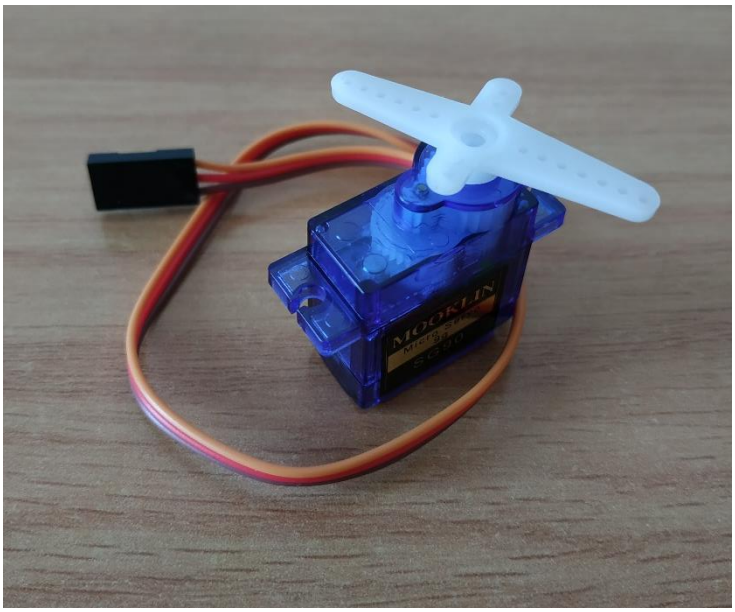


Ilustración 76: Servomotor SG90

5.1.3 Servo Driver PWM PCA9685

Debido al número de servos que se utilizan, 6, se decide utilizar un controlador de servos para ello. Concretamente el modelo escogido es capaz de gobernar hasta 16 salidas PWM con 12 bits de resolución, por lo que de cara a posibles ampliaciones del proyecto o incluso proyectos diferentes puede ser un componente muy útil. Esto junto a su alta valoración y fiabilidad unido a su precio económico lo convierten en una elección fundada y adecuada.

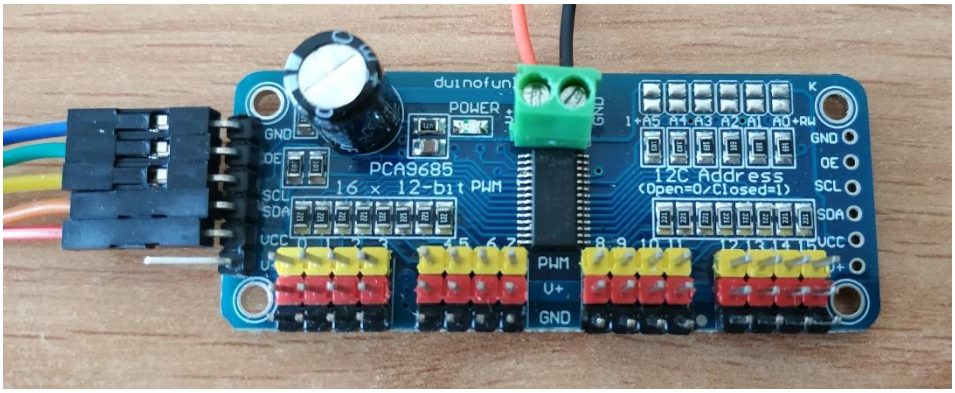


Ilustración 77: Servo Driver PCA9685

El módulo se alimenta a 5V y la comunicación con la placa Arduino se realiza usando el protocolo I2C. La alimentación mencionada es la correspondiente a la parte lógica del módulo, en lo que a los servos se refiere hay que seleccionar una fuente de tensión entre 4.5 y 6 V, que es el voltaje entre el que pueden trabajar nuestros servomotores, y que se conecta a la bornera verde. Para el manejo de este módulo con Arduino se utiliza la librería *Adafruit PWM Servo Driver Library*, disponible en el propio IDE de Arduino.

5.1.4 Fuente de alimentación de los servomotores

Como se acaba de mencionar, debe proporcionar entre 4.5 y 6V. Observando el consumo de los servomotores utilizados, varía entre 500-900 mA para el MG996R y entre 100-250 mA para el SG90. Sobredimensionando ligeramente, se considera que el consumo total será de 3*800+3*200 = 3000 mA.

La primera opción estudiada es una batería Li-Po que cumpla las especificaciones de tensión y con una capacidad de unos 6000 mAh, para poder otorgarnos unas 2 horas de funcionamiento sin necesidad de carga. Pero el precio es elevado y se decide buscar alternativas más económicas en el mercado. Así, se estudia la posibilidad de alimentar los servos usando pilas alcalinas de clase AA, denominadas LR6 por el IEC (International Electrotechnical Comission), en una disposición serie. Se trata de una opción más económica y cuenta con la ventaja de que este tipo de baterías tienen una elevada capacidad, concretamente entre 1700-3000 mAh. En contrapartida, estas pilas no son recargables, ya que si así se quisiera, el precio vuelve a subir de manera sustancial.

Es llegado a este punto donde se debe decidir qué opción es más conveniente. Tras sopesarlo, se considera que el proyecto tiene un enfoque académico y lectivo, pero en ningún caso se baraja la posibilidad de que entre en funcionamiento de manera regular dentro de algún proceso productivo. Por ello se entiende que la mayor parte del uso se dará durante la fase de pruebas y puesta a punto como en las sucesivas presentaciones. Para dicho uso, se estima que con 2-3 horas de uso continuado serán suficientes, por ello se opta por la segunda opción.

Se adquieren 4 pilas de 1,5 V que se dispondrán en serie para obtener el voltaje deseado como fuente de alimentación. Siendo conservadores en la estimación, ya que en las pilas adquiridas sólo se especifica que son de tipo LR6 y no la capacidad concreta, se tasa esta en 2000 mAh. Por lo que la fuente de alimentación tiene una capacidad de 8000mAh. Haciendo los cálculos pertinentes teniendo en mente que C=I*t, se obtiene que con la actual fuente de alimentación el sistema sería capaz de funcionar ininterrumpidamente durante 2h y 40 minutos, lo cual se adapta perfectamente a las intenciones de uso estipuladas.

Ilustración 78: Fuente de alimentación 6 V y 8000 mAh

5.1.5 Módulo Bluetooth HC-06

Para la comunicación entre el Arduino que alberga el programa de control y el sistema emisor de las órdenes, en principio será un teléfono móvil pero cualquier dispositivo que pueda enviar información por Bluetooth es apto, se utiliza el módulo HC-06. Sólo funciona como esclavo y se alimenta a 3.3 V, procedentes de la propia placa. Con esta última se conecta a través de los pines de comunicación serie (*Rx,Tx*). La información que recibe por Bluetooth la transmite al Arduino del pin *Tx* del módulo al *Rx* (0,1,2,3) del Arduino. En el caso de que se envíen datos de la placa al módulo la vía de comunicación deberá ser a la inversa, es decir, desde los pines *Tx* (0,1,2,3) de la placa conectados al *Rx* del módulo y una vez este los reciba los envía por Bluetooth al receptor deseado.

Ilustración 79: Módulo bluetooth HC-06

5.1.6 Microcontrolador Arduino DUE

Se trata del corazón del montaje, el encargado de gobernar todos los componentes electrónicos anteriores haciendo uso de diferentes algoritmos de control. El microprocesador que lleva incorporado es un Atmel SAM3X8E ARM Cortex-M3. Entre sus características principales resalta que al contrario que la mayoría de placas Arduino, el Due trabaja a 3.3 V (Mega, Nano, UNO, etc. trabajan a 5 V), siendo este el voltaje máximo que toleran sus pines de entrada/salida (I/O). En referencia a estas, cuenta con 54 pines I/O digitales de las cuales 12 pueden ser utilizadas como salidas PWM, 4 puertos serie UART (*Rx,Tx*) y 12 entradas analógicas. Destacar también que cuenta con un reloj de 84MHz de frecuencia y 2 DAC (conversión digital a analógico). El resto de sus características y funcionalidades, al igual que en el caso de los anteriores dispositivos, se pueden revisar con mayor detalle analizando su datasheet incluida en 12.3 Datasheets.

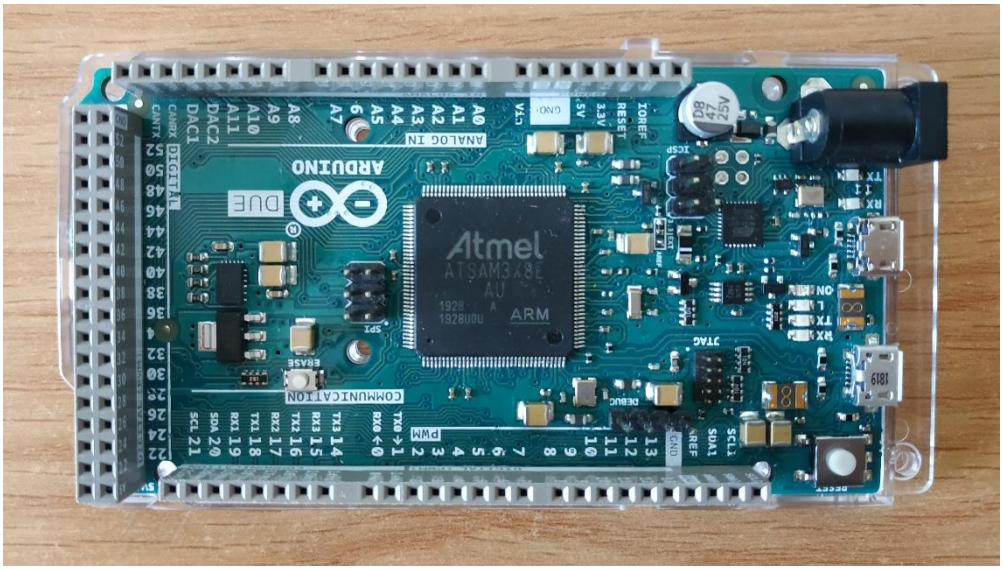


Ilustración 80: Placa Arduino DUE

5.1.7 Tornillería

Para el acople de los diferentes componentes del Brazo Robot se utilizan los accesorios y tornillos que vienen con los servomotores, como se explicará más en detalle en el apartado 6 Montaje y resultados reales. Pero para el caso del Gripper es necesario el uso de tornillos y tuercas. Concretamente se han usado tornillos Allen M3x20mm y tuercas autoblocantes M3. Estas últimas son indispensables ya que si se utilizan tuercas normales, debido al giro que se produce entre los componentes que unen, se puede producir un aflojamiento de la unión con relativa facilidad.

5.2 Desarrollo de algoritmos de control

El control del Brazo Robot estará distribuido en dos partes, la primera será la correspondiente al algoritmo que albergará la placa Arduino y que en función de la información que reciba realizará una u otra acción. La otra parte estará embebida en el dispositivo de mando que se utilice para gobernar el movimiento del Brazo Robot. Como ya se ha aludido anteriormente, dicho dispositivo puede ser cualquiera que disponga de la capacidad de enviar información a través de Bluetooth. A efectos de comodidad y práctica, unido a la motivación del autor de esta memoria por aprender acerca del mundo de las apps móviles, se decide que el programa de envío de órdenes estará albergado en una aplicación de Android creada por el alumno. Una vez explicada la metodología que se va a usar, se procede a explicar cada una de las etapas que la conforman.

5.2.1 Arduino

Para la elaboración del programa de control final previamente es necesario el testeo y puesta a punto de los diferentes componentes a utilizar, para una vez comprobado que funcionan de manera correcta por separado, hacerlos funcionar de manera conjunta bajo un algoritmo único. Se aplica así la metodología de convertir una tarea grande y complicada en varias simples, facilitando su realización. De esta forma también se facilita la detección de errores y malfuncionamiento de los componentes, pues hay menos elementos donde pueden estar localizados. Notar que para esta fase se ha utilizado el entorno de programación habitual para proyectos Arduino, Arduino IDE.

El primer paso de este proceso consiste en comprobar que el Servo Driver funciona de manera correcta así como en calibrar los servomotores para ver en qué rango PWM trabaja cada uno, ya que pese a que se usan sólo dos tipos, se pueden dar variaciones entre servos del mismo modelo. Para ello, se realizan las conexiones correspondientes explicadas anteriormente en 5.1.2 Servomotores y a través de un programa cuyo código se puede encontrar en 12.1.1 Calibración de servos se aumenta de manera cíclica el ancho de pulso PWM que se envía a cada servo para ver en qué valor de ancho de pulso empieza a moverse (límite inferior del rango de acción) y en cuál se para (límite superior del rango de acción). Una vez calibrados los servos, se sabe entre qué valores funcionan y se podrá definir cuál será su disposición óptima durante el montaje. Destacar el uso de las librerías *Adafruit_PWMServoDriver.h* para el manejo del Servo Driver y *Wire.h* para habilitar la comunicación I2C.

El siguiente paso consta de la puesta a punto del módulo Bluetooth HC-06 y la conexión de este con otro dispositivo Bluetooth y con Arduino. Por partes lo primero es realizar la conexión con Arduino para ser capaces de configurar el módulo según nuestras preferencias. Siguiendo los pasos explicados en 5.1.5 Módulo Bluetooth HC-06 se conectan ambos dispositivos y haciendo uso del programa cuyo código se

encuentra en 12.1.2 Configuración Bluetooth se configura el módulo HC-06 introduciendo a través del monitor serie los comandos AT explicados en su datasheet la cual se puede encontrar en 12.3 Datasheets. Se establecen así parámetros como el nombre del módulo, el PIN de emparejamiento o la velocidad de comunicación en baudios. Realizada la configuración, es el momento de emparejar el módulo HC-06 con un dispositivo Bluetooth y probar la conexión. Para esta prueba se descarga la aplicación Android *Serial Bluetooth Terminal* en un teléfono móvil y se conecta al módulo, el cual está disponible para la conexión siempre que no esté emparejado y reciba alimentación. Con esta aplicación se pueden enviar caracteres, los cuales recibe el HC-06 y se interpretan usando el programa cuyo código se puede encontrar en 12.1.3 Prueba Bluetooth. Este, en función del carácter recibido, ilumina un LED determinado o emite un mensaje de error. Habiendo comprobado el correcto envío, recepción e interpretación de la información y por tanto habiendo verificado el correcto funcionamiento tanto de los componentes utilizados como del código empleado para gobernarlos, se procede a aunarlo todo en el programa de control final, que irá cargado en el Arduino durante el funcionamiento del brazo.

Este algoritmo se basa en el de comunicación Bluetooth anterior con la diferencia de que ahora en función del carácter recibido no se encenderá un LED sino que se producirá el movimiento del servo seleccionado en el sentido de giro seleccionado. El código relativo a este programa se puede encontrar en 12.1.4 Programa final.

Finalizado esto, se confirma el funcionamiento correcto del programa; pero la app *Serial Bluetooth Terminal* no posee muchas más funcionalidades a parte del envío de caracteres por lo que el manejo del robot se puede hacer tedioso. Por ello, unido a los motivos expuestos con anterioridad se procede a la fase de concepción de una aplicación más versátil, con mayor atractivo visual y personalizada para el caso concreto de este proyecto.

5.2.2 Aplicación Móvil

Para el desarrollo de la aplicación se utiliza la plataforma del MIT (Massachussets Institute of Technology) denominada MIT App Inventor. Esta fue seleccionada por lo intuitivo que resulta su aprendizaje, basado en un lenguaje de programación orientado a bloques que facilita la elaboración de estructuras complejas. En propias palabras de su página web “El objetivo es democratizar el desarrollo de software empoderando a la gente, especialmente gente joven, para pasar del consumo de tecnología a la creación de tecnología”.

El entorno de programación consiste en 2 pantallas, siendo la primera de ellas la pantalla de *Diseño*. Aquí es donde se concibe el aspecto visual que presentará la aplicación.

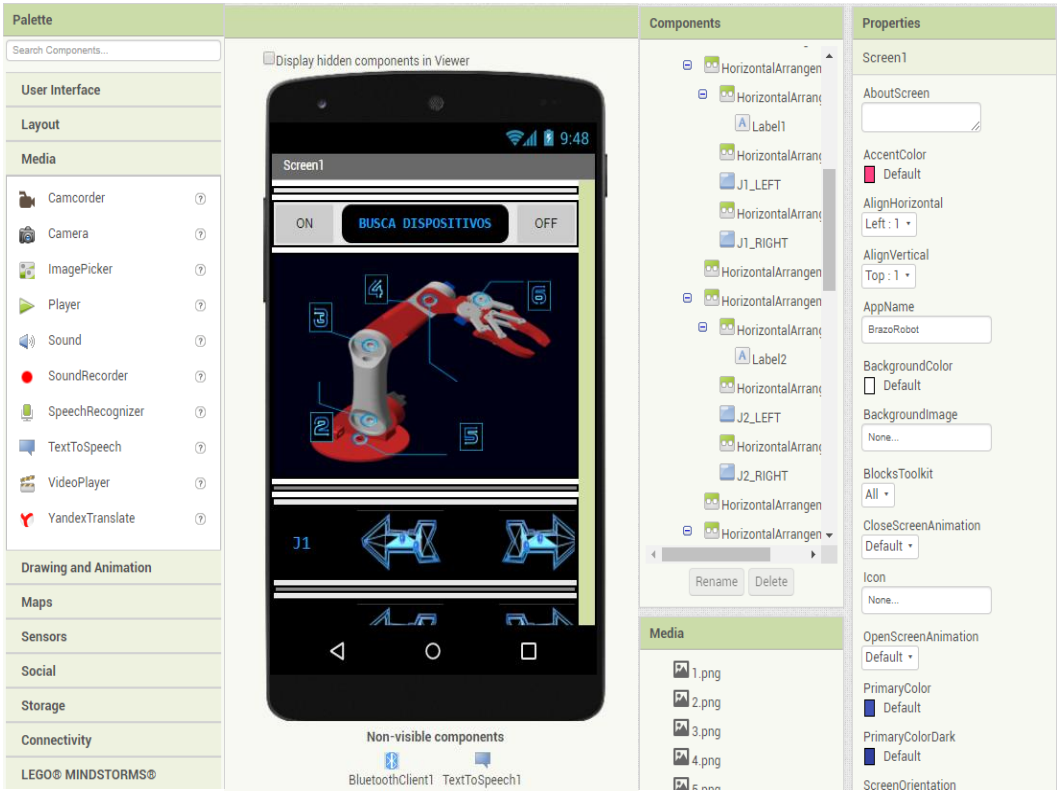


Ilustración 81: Pantalla de Diseño MIT App Inventor

De izquierda a derecha, la primera sección recibe el nombre de *Paleta* y es aquí donde se seleccionan los diferentes componentes que se quieren incluir en la aplicación. Hay una gran variedad de opciones desde elementos de interfaz de usuario como botones, switches, listas desplegables, etc. a sensores, mapas o elementos multimedia como pueden ser sonidos o llamadas a la cámara del dispositivo. A su derecha se encuentra el *Visualizador* que es donde se aprecia el aspecto visual que va tomando la aplicación. Más a la derecha se encuentra *Componentes*, que como indica su nombre, es la lista de componentes seleccionandos que conforman la app. Haciendo click en ellos, se abre la última columna *Propiedades* donde podemos editar al gusto las características de cada uno. Por último, justo debajo de la lista de componentes se encuentra la sección *Media* donde el usuario puede subir sus imágenes, sonidos o videos.

Una vez incluidos los componentes y habiendo personalizado al gusto la aplicación, es el turno de definir qué acciones realiza cada elemento y cómo interactúan entre ellos. Para esto, se utiliza la segunda pantalla denominada *Bloques.* Esta parte es la correspondiente a la programación de la app propiamente dicha.

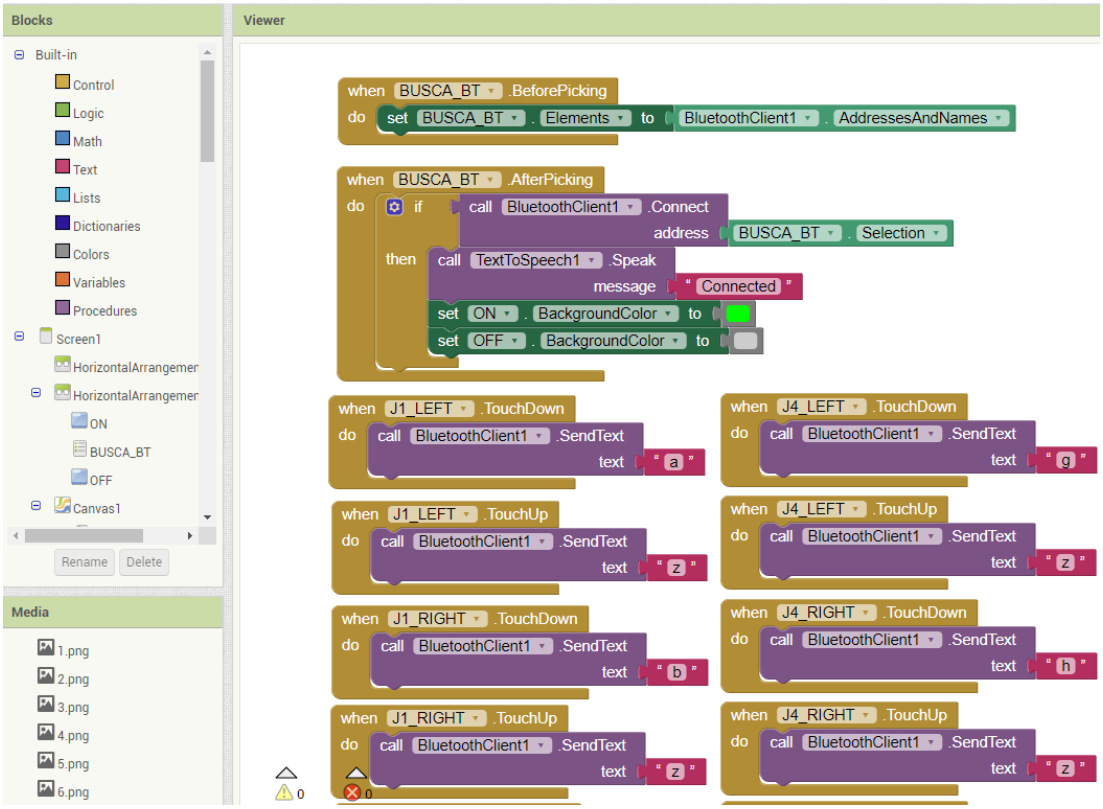


Ilustración 82: Pantalla de Bloques MIT App Inventor

La sección situada a la izquierda y que recibe el nombre de *Bloques* alberga los diferentes módulos de programación ordenados en diferentes categorías. En primer lugar se encuentran las categorías genéricas donde se incluyen todos los bloques de programación disponibles. Más abajo se encuentra la lista de elementos que conforman la app. Si se hace click sobre ellos se despliega una lista de los diferentes bloques compatibles con dicho elemento, lo que al juicio del autor, simplifica en gran medida el proceso de programación. La sección de *Visualizador* tiene el mismo concepto que en la pantalla anterior, pero en este caso lo que se visualiza son los bloques que se han ido incluyendo. Estos se unen entre sí como si fueran piezas de un puzzle, lo cual es muy intuitivo, siendo posible identificar qué uniones tienen sentido y qué puede faltar para realizar la unión requerida.

Presentada la interfaz se pasa a explicar cómo funciona la aplicación, de manera algo más detallada que el código de Arduino, el cual prácticamente se ha pasado por encima, ya que se considera que es más conocido para el lector que esta novedoso entorno de desarrollo.

En primer lugar, una vez se enciende la aplicación lo primero será emparejar el dispositivo en el que está funcionando la app con el módulo Bluetooth HC-06. Para ello se habilita el *ListPicker* "Busca Dispositivos" el cual despliega una lista de los dispositivos Bluetooth emparejados con el móvil en cuestión (por ello, es importante emparejar previamente el teléfono al dispositivo al cual se quiera conectar). En el momento de realizar la selección y si la conexión es satisfactoria, el indicador de ON se encenderá de color verde y se emitirá una voz diciendo "Connected". Cuando se quiera desconectar se pulsa el botón de OFF y este pasará a tener un color ROJO desactivándose el color del botón de ON, desconectando el dispositivo y emitiendo una voz de "Disconnected". Los elementos y bloques empleados se muestran a continuación.

Ilustración 83: Elementos para la conexión BT

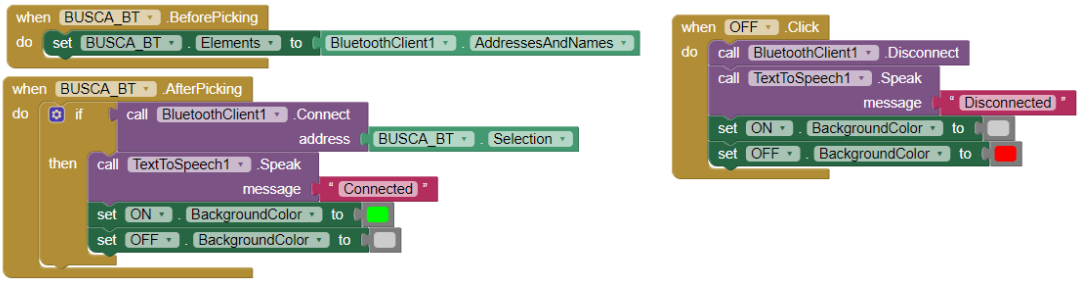


Ilustración 84: Bloques para la comunicación BT

Una vez llevada a cabo la conexión es el turno de enviar las órdenes, para ello se dispone de dos botones con aspecto de flecha, una por cada sentido de giro, para cada articulación. Así si se pulsa la flecha izquierda de la articulación X, esta girará en sentido antihoriario y si se hace lo propio con la derecha la articulación girará en sentido horario. Para una mayor comprensión del funcionamiento, se incluye una imagen del brazo con una leyenda para conocer qué articulación del robot corresponde con su denominación en la app. Añadir que la imagen de referencia mencionada es estática, mientras que la selección de la articulación a mover es deslizable, de manera que siempre es posible ver si la articulación que se quiere mover es la seleccionada. Esto se muestra en las dos imágenes siguientes, que son pantallazos del aspecto que presenta la aplicación en un dispositivo móvil y donde se puede apreciar la función deslizable del cuadro de mando.

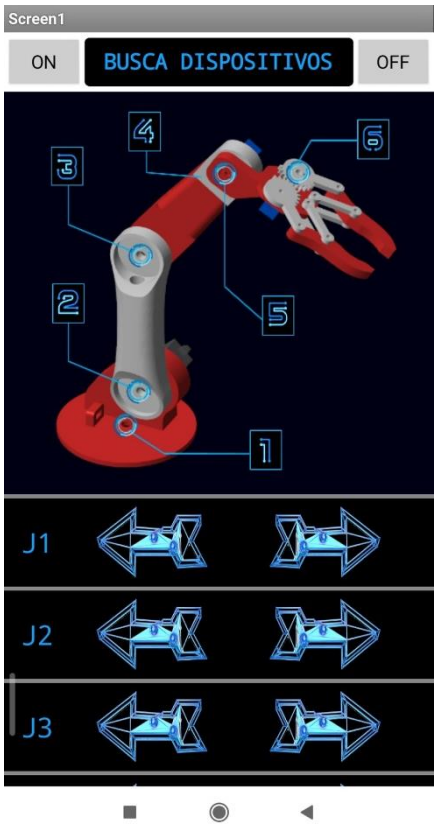


Ilustración 85: Aspecto de la app en un móvil (1)

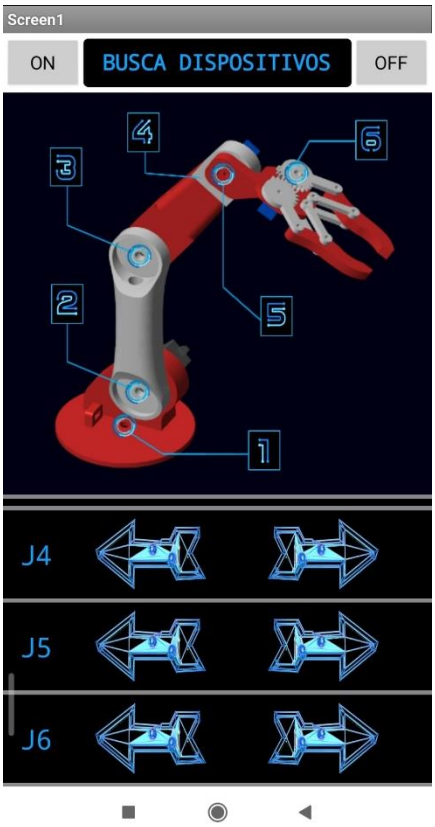


Ilustración 86: Aspecto de la app en un móvil (2)

Los bloques necesarios para habilitar las acciones arriba mencionadas se presentan a continuación.

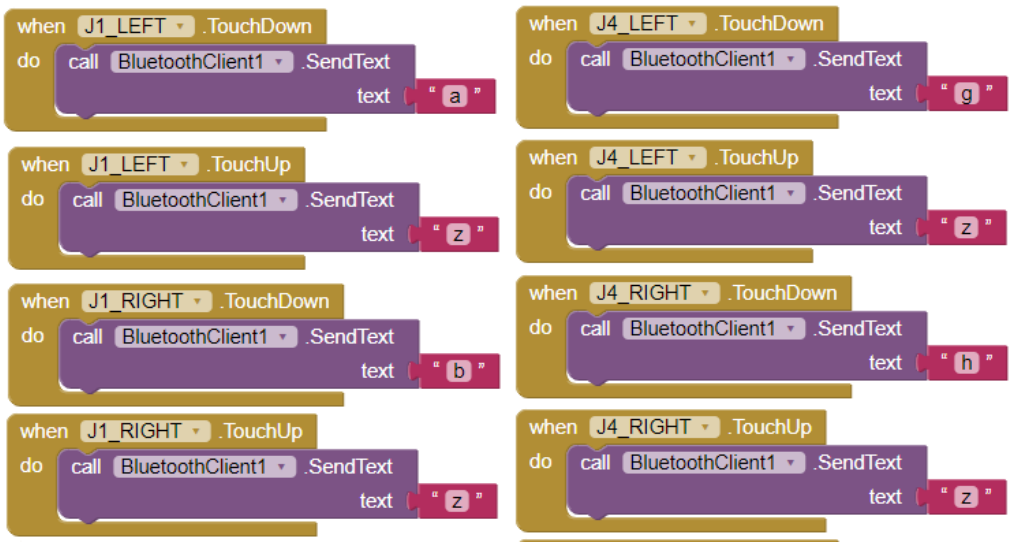


Ilustración 87: Bloques para el control de movimiento

Se han incluido sólo los bloques referentes a dos articulaciones porque la arquitectura es extensible a las demás. Cuando se pulsa, por ejemplo, el botón J1_LEFT (Articulación 1 hacia la izquierda) se envía el carácter “a”. Al recibir esa información el algoritmo alojado en el Arduino lo interpreta y realiza la acción correspondiente. Además, cómo se puede observar en 12.1.4 Programa final, mientras el botón esté pulsado se realiza el incremento de movimiento establecido con cada vuelta de ciclo y con un retardo de 15 ms, facilitando el control del brazo y evitando así tener que pulsar el botón cada vez que queramos incrementar la posición en un sentido dado. Una vez se deja de presionar el botón, se emite el carácter “z”, que el código de Arduino interpreta como un estado de stand-by, estando el robot listo para acometer el siguiente movimiento.

6 MONTAJE Y RESULTADOS REALES

Una vez más, debido a las circunstancias actuales mencionadas anteriormente sólo ha sido posible realizar el montaje el Brazo Robot. No sin dificultad, pues el encontrar los componentes y herramientas necesarios para su construcción se ha hecho mucho más laborioso que en condiciones normales. Esto es debido a que la situación actual supone pasar de trabajar en el laboratorio y taller del DISA, donde se cuentan con todas las herramientas y componentes necesarios, a trabajar desde un piso de estudiantes donde se carecen de los medios idóneos para esta fase del proyecto. Pese a ello, adquiriendo material básico de bricolaje y haciendo uso de la imaginación y el ingenio, se consigue llevar a cabo esta tarea.

Lo primero es conseguir un soporte que realice la función de la Plataforma pero sin la capacidad de moverse, es decir, que albergue los componentes electrónicos que posibilitan el funcionamiento del Brazo a la vez que hace las veces de soporte de este. Para ello se acondiciona una caja de zapatos, pues por el material y las dimensiones es lo que más se acerca a lo que se está buscando. Se hacen los taladros pertinentes y se realiza la sujeción, usando tornillos Allen M3x20mm y tuercas M3 autoblocantes, del servomotor que acciona la Base giratoria. Esta se monta encima y ambos componentes se acoplan haciendo uso de los tornillos que incluyen los servomotores al adquirirlos. Debido a que el grosor de la tapa del soporte es inferior a la tapa de la Plataforma diseñada, esto implica que el servomotor de la Base giratoria sobresale unos 10 mm más de lo calculado a priori, traduciéndose en una ligera inclinación del Brazo. Para corregir esto, se incluyen unos tapones de refresco que funcionan como guías y soportes para la Base giratoria y solucionan el problema sin afectar al movimiento de esta.

Ilustración 88: Montaje del Servomotor 1 y las guías-soporte de la Base giratoria

Ilustración 89: Montaje de la Base Giratoria

Sucesivamente se montan el Brazo 1 y el Brazo 2. Lo primero es fijar los servomotores correspondientes a cada articulación usando los tornillos incluidos con cada uno de ellos. La unión entre un eslabón y otro se realiza igual que en el caso de la Base giratoria, en primer lugar, fijando el acople del servo a la articulación que se quiere mover con este y posteriormente uniendo el acople con el eje dicho servo. Esto es así para todos los eslabones excepto en el caso del Brazo 2 que se mueve por la acción del motor que tiene fijado él mismo, ya que el Brazo 1 no alberga ningún servo

Ilustración 90: Montaje de los Brazos 1 y 2

A partir del Brazo 3, este incluido, los eslabones se mueven usando los servos SG90 pero como se ha explicado la metodología de montaje es la misma. A continuación se muestran unas imágenes detallando la unión entre eslabones por si no hubiera quedado claro.

Ilustración 91: Detalle del servo alojado en Brazo 2 que moverá Brazo 3

Ilustración 92: Detalle del acople fijado a Brazo 3 para ser movido por el servo alojado en Brazo 2

La unión entre el acople y el servo se hace haciendo uso de uno de los tornillos incluidos con el servo, diseñado específicamente para ello.

Cabe destacar que a medida que se monta un nuevo eslabón se ajusta su rango de movimiento en base a los valores máximo y mínimo establecidos durante el proceso de calibración (cuyo código se puede encontrar en 12.1.1 Calibración de servos) con el fin de optimizarlo al máximo antes de fijarlo de manera definitiva.

El montaje del Gripper se realiza utilizando nuevamente tornillos Allen M3x20mm y tuercas autoblocantes M3 para la unión de sus diferentes componentes prestando especial atención al apriete de las diferentes uniones, pues debe ser lo suficientemente fuerte como para garantizar su sujeción a la vez que permite el movimiento relativo entre las partes. Para ello se van ajustando las tuercas a la vez que se hace funcionar el servo, hasta dar con la solución de compromiso idónea.

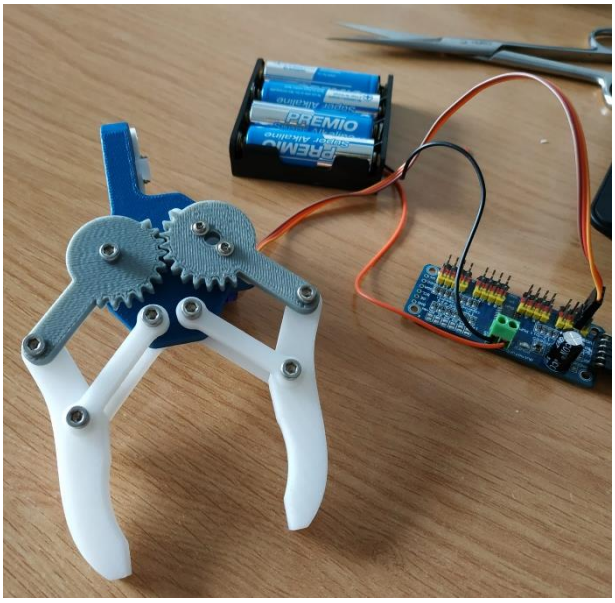


Ilustración 93: Aspecto final del Gripper

Una vez unidos todos los eslabones, se muestra la apariencia que luce el Brazo Robot.

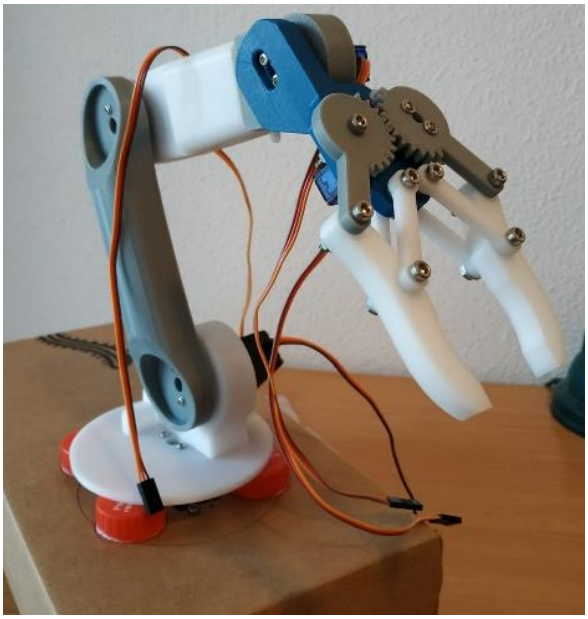


Ilustración 94: Aspecto del brazo con todos sus eslabones 1

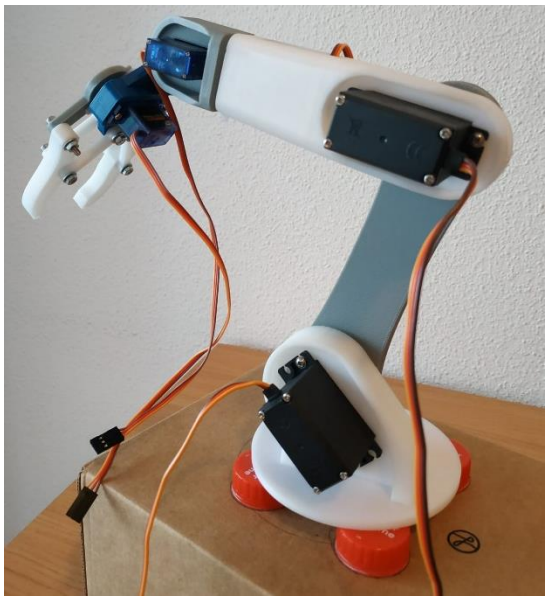


Ilustración 95: Aspecto del Brazo con todos sus eslabones 2

Habiendo montado todos los eslabones del Brazo Robot, el siguiente paso es la introducción del Microcontrolador Arduino DUE, el Servo Driver PWM PCA9685, la Fuente de alimentación de los servomotores y el Módulo Bluetooth HC-06 dentro de la caja soporte.

Antes de realizar las conexiones de los servomotores con el Servo Driver, conviene ordenar los cables, pues como se puede observar en las imágenes superiores, se encuentran desperdigados libremente. Para ello, primero se alargan los cables procedentes de cada servo usando cables macho-hembra. La unión entre estos se asegura usando cinta de carrocero en sus extremos como se muestra a continuación.

Ilustración 96: Detalle de la unión entre cables Dupont

La extensión de cable precisada se evalúa viendo cuál es la configuración del robot más desfavorable para cada uno de los motores, esto es, la que más longitud de cable necesita.

Habiendo hecho esto, se introducen por el conducto diseñado específicamente para albergarlos del Brazo 1, prestando especial atención a la holgura que necesita el cable desde el servo hasta el orificio superior del conducto citado, asegurando así su integridad durante el funcionamiento. Para que los cables penetren en el interior de la caja y proceder a su conexión con el Servo Driver se realiza un agujero en la tapa de la caja soporte.

Ilustración 97: Detalle de la inserción de los cables en el conducto de Brazo 1

Ilustración 98: Detalle del agrupamiento de cables y su inserción en la caja soporte

Para finalizar se realizan las conexiones de cada servo con su puerto correspondiente del Servo Driver y se realiza un agujero en la parte trasera de la caja para introducir el cable de alimentación del microcontrolador Arduino DUE.

Comentar que no ha sido posible incluir una foto de la conexión en el interior de la caja ya que esta se abre como un libro y para sacar una imagen mínimamente decente se debería abrir demasiado poniendo en peligro la integridad de las conexiones. Se ha considerado cortar la tapa, pero al no garantizarse la estabilidad del soporte se ha descartado esa opción.

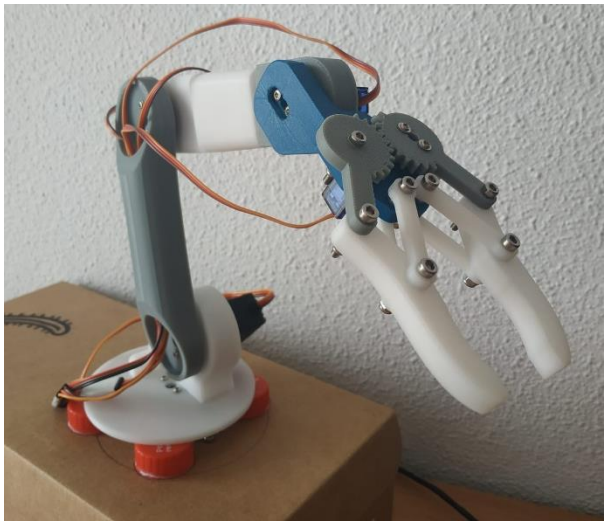


Ilustración 99: Aspecto final del Brazo Robot

Como punto final a este capítulo se muestran unas imágenes del sistema realizando labores de Pick & Place para las cuales fue concebido. Como parte de los archivos a entregar se incluye una demostración del proceso completo.

Ilustración 100: Captura proceso Pick and Place 1

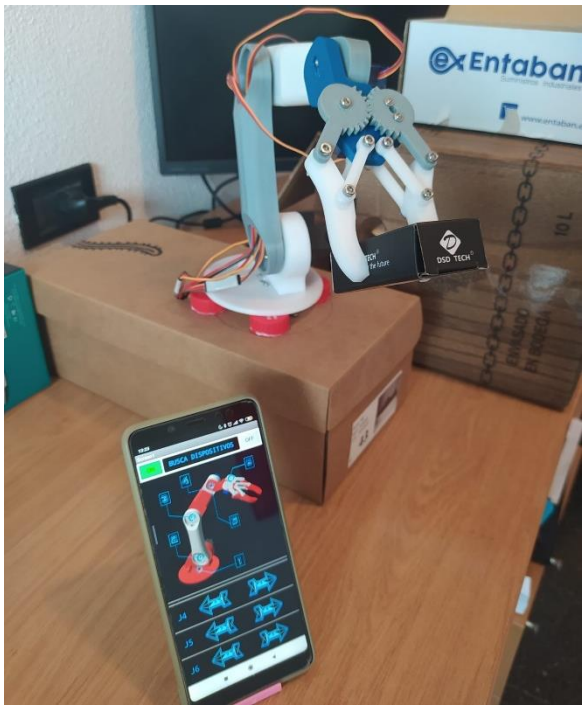


Ilustración 101: Captura proceso Pick and Place 2

Ilustración 102: Captura proceso Pick and Place 3

7 PROPUESTAS DE MEJORA

7.1 Motores de CC para Brazo Robot

Vistos los diferentes casos de estudio para la simulación del Brazo Robot y su funcionamiento real, se puede afirmar que la elección de servomotores como elementos motrices de la estructura está justificada dado su buen funcionamiento. Pero en lugar de estos, se podrían haber empleado motores de corriente continua y el resultado podría ser igualmente efectivo. Por ello, se considera oportuno un estudio alternativo usando este tipo de motores, en el cual el control que asegura la correcta posición, que en el caso de los servomotores está integrado en ellos mismos, sea sustituido por un bucle de control diseñado por el alumno con el fin de analizar la viabilidad de la alternativa propuesta.

Los motores de corriente continua a emplear, con objeto de no complicar el proceso, serán los mismos que para la plataforma omnidireccional (Ilustración 16: Motor CC Pololu 37D-70).

Al tratarse de una propuesta de mejora a ser desarrollada, no se realizará un diseño exhaustivo del modelo (no se cambiarán en el diseño del Brazo Robot los servos por motores de CC pues implicaría cambiar completamente el diseño del brazo), sino que se hará únicamente desde el punto de vista del control. Se eliminarán los subisistemas de entrada y salida de las articulaciones utilizados en los ensayos con servos y se añadirán nuevos subistemas análogos.

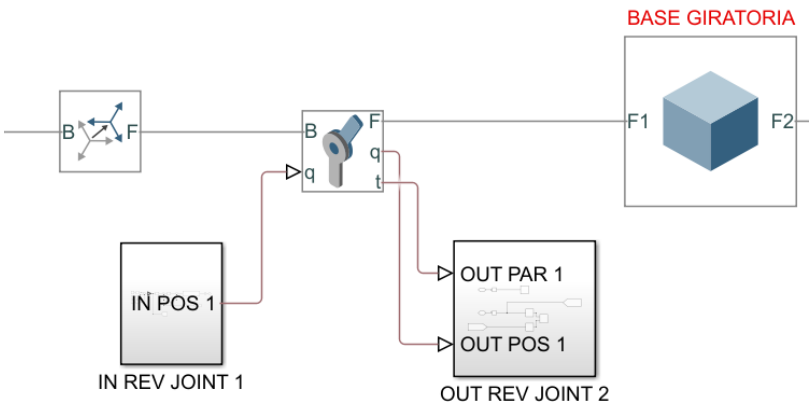


Ilustración 103: Subsistemas de entrada y salida de la Articulación 1

El bloque de entrada de cada articulación tendrá la estructura que se muestra en la imagen a continuación.

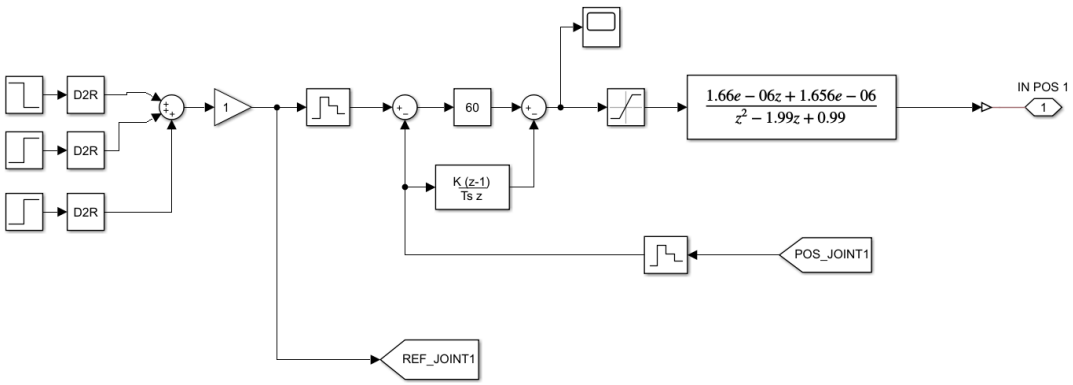


Ilustración 104: Subsistema de entrada a la Articulación 1

En ella se representa el motor empleado usando su función de transferencia discretizada, generada a partir de la función de transferencia continua típica de este tipo de motores junto a los valores específicos del modelo sacados de su datasheet.

$$\theta(s) = \frac{K_v}{(\tau_1 s + 1)s} U(s)$$

Ilustración 105: Función de transferencia para control de posición Motor CC

La estructura de control es en bucle cerrado y se utiliza un regulador PD discreto. Tanto la entrada de referencia como la realimentación se discretizan con un bloque *Zero Order Hold* con el objetivo de trabajar siempre con el mismo periodo de muestreo. La acción de control que entra en la función de transferencia es limitada a al valor máximo de tensión que admite el motor y que proporciona una velocidad máxima de giro de 12 rad/s. Esto se consigue usando un bloque *Saturation*.

El subsistema de salida está conectado por dos puertos a las salidas de par y posición de la articulación en cuestión. La primera simplemente se grafica mientras que la segunda hace lo propio, pero se compara con la referencia con el fin de observar el seguimiento. La salida de posición también realimenta al bucle de control de posición.

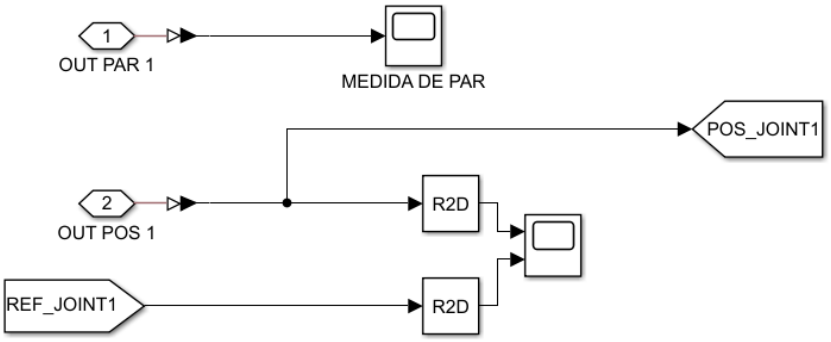


Ilustración 106: Subsistema de salida de la Articulación 1

Una vez presentados los nuevos elementos del modelo de simulación se procede a realizar una serie de ensayos para definir los valores de la ganancia proporcional y derivativa del regulador. Tras una serie de simulaciones en articulaciones trabajando de manera aislada y conjunta, se concluye que los valores más adecuados para las intenciones de uso de este proyecto son Kd=5 y Kp=60. A la hora de la implementación real habría que ajustar dichos valores en caso de que tal la velocidad de movimiento suponga un riesgo para la integridad de los componentes.

Estableciendo una serie de referencias de posición para cada articulación en unos instantes determinados, se obtienen los siguientes resultados.

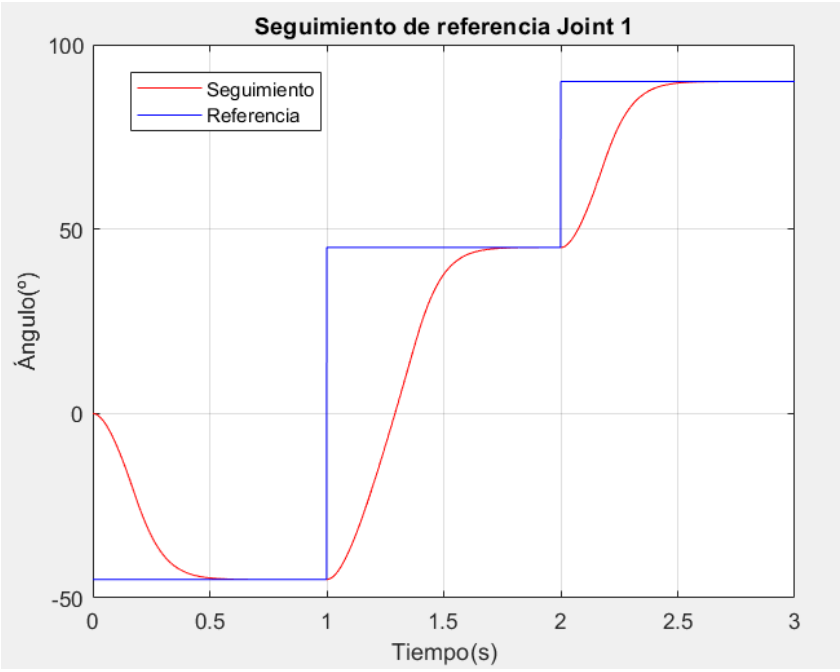


Ilustración 107: Seguimiento de referencia Joint 1 (Ensayo MotorCC)

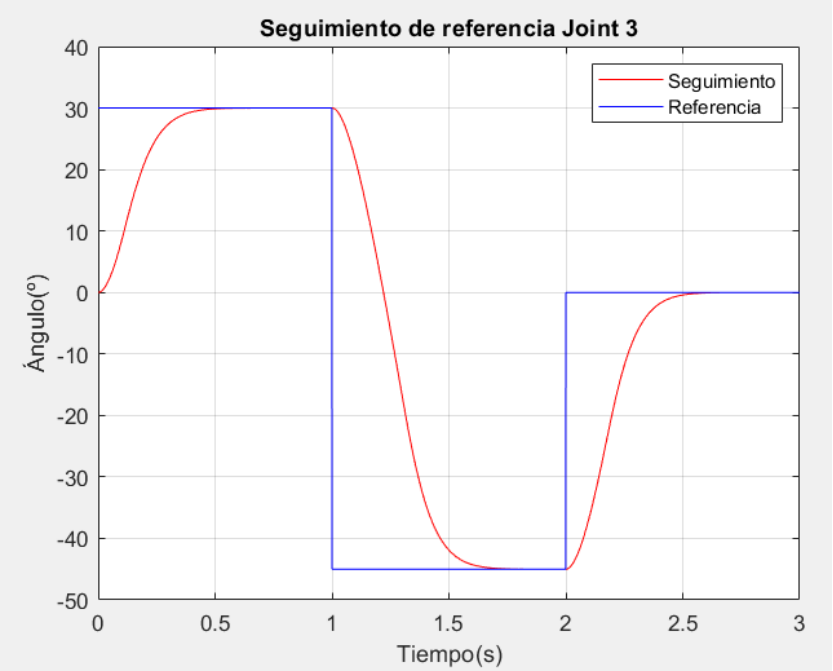


Ilustración 108: Seguimiento de referencia Joint 3 (Ensayo MotorCC)

Se muestran los resultados de las Articulaciones 1 y 3 por ser junto a la Articulación 2 las que más exigidas están a nivel de par. Se advierte que el seguimiento es adecuado, tardando medio segundo aproximadamente en alcanzar la referencia de posición y con error nulo.

Por tanto, siendo favorables los resultados de la simulación, quedaría pendiente para futuras ampliaciones o pruebas con el proyecto, llevar a la realidad el uso de motores de corriente continua y su control embebido en un microcontrolador.

7.2 Implementación real de la plataforma omnidireccional

Como se ha explicado ya en el apartado final de 1 Introducción y objetivos, por circunstancias externas y fuera del alcance del autor de este proyecto, no ha sido posible el montaje e implementación real de la Plataforma Omnidireccional. Por ello, es esta la mejora más inmediata y relevante que se puede hacer, trasladando los conceptos explicados tanto en 2 Estudio teórico como en 4.2.2 Simulación de trayectorias y aplicándolos en el diseño de un sistema de control análogo en una plataforma como Arduino o Raspberry Pi. La obtención de los elementos estructurales a través de la fabricación aditiva así como la adquisición de los componentes necesarios se puede hacer de manera directa y sencilla en condiciones normales y no debe suponer un contratiempo.

Uno de los puntos que más interés suscitaban y cuya consecución se hubiera abordado en caso de poderse llevar a la realidad la plataforma es la inclusión y desarrollo de un sistema de balizamiento para conocer la posición en tiempo real de la plataforma con el fin de utilizar dichos datos como realimentación del control de posición. Esto supone una solución al problema que conlleva el uso de los datos de los encoders de los motores de las ruedas, pues un control basado en esa información no tiene en cuenta factores como el deslizamiento, que son claves a la hora de controlar de forma precisa la posición de la Plataforma. También en el caso de querer controlar el efecto del deslizamiento con el fin de aumentar más aún la precisión del sistema, es recomendable la lectura de *Omnidirectional Control* de Raúl Rojas [1], donde se trata dicho tema.

8 CONCLUSIONES

Una vez acabado el Trabajo de Fin de Máster y echando la vista atrás, se puede afirmar que se han cumplido la mayoría de tareas y objetivos propuestos al empezar.

Por un lado se ha podido poner en práctica los conocimientos de control automático y programación adquiridos a lo largo de este último año y ampliarlos de manera notable, ganando mucha práctica en el diseño de diferentes sistemas de control y por ende comprendiendo en mayor medida el funcionamiento y la correlación entre sus partes. El hecho de combinar esto con la labor de diseño de productos usando herramientas CAD/CAM, la cual siempre ha sido del agrado del autor, para la concepción de entornos y modelos de simulación ha jugado un papel importante en el grado de involucración en el proyecto, ya que salvo en algunas ocasiones en las que dar con una solución se hace cuesta arriba, en líneas generales este ha resultado entretenido de llevar a cabo.

En el caso del Brazo Robótico, se ha conseguido el montaje e implementación real del prototipo planteada al inicio del trabajo con resultados satisfactorios, por lo que se considera que en lo relativo a él se han cumplido los objetivos con exactitud. En cuanto a la Plataforma Omnidireccional, la implantación de la solución al problema cinemático que supone el seguimiento de la trayectoria ha arrojado resultados muy positivos durante las sucesivas simulaciones, lo cual invita a ser moderadamente optimistas en lo que habría sido su implementación real. Es esto último, para lástima del autor, lo único que no se ha podido cumplir de la propuesta inicial, pero el haber sentado las bases para hacerlo y haber hecho todo lo posible por desarrollarlo al máximo hace que los resultados obtenidos se consideren favorables.

Se ha comprendido la importancia de la metodología del mínimo producto viable a partir del cual ir desarrollando mejoras y puliendo aspectos anteriores. Una metodología que se considera aplicable al mundo laboral y otros aspectos de la vida donde sea necesario cumplir con unos requerimientos básicos y en base a ellos tratar de ir más allá y mejorar de forma continua. Junto a ella, se destaca otra metodología de trabajo implementada a lo largo de todo este proyecto, la cual consiste en desgranar una tarea complicada en muchos simples, facilitando su realización así como la evaluación del progreso realizado.

Por último, el proceso de montaje ha demostrado ser el más problemático, aparte de por las circunstancias actuales ya mencionadas anteriormente, porque es la fase en la que se debe hacer la transición de la teoría a la práctica, del mundo simulado al real. Y por tanto, surgen una serie de incidencias que no se habían tenido en cuenta, como puede ser la difícil retirada de los soportes de agujeros o taladros en piezas de impresión 3D o el cambio de algunos componentes por otros que en las circunstancias de uso intencionadas ofrezcan un mejor resultado, como las tuercas autoblocantes. Esto sin duda, ha ayudado a entender la vasta cantidad de cosas a tener en cuenta a la hora de fabricar un prototipo desde cero y ha dotado al autor de más experiencia y preparación para futuros desafíos.

9 PRESUPUESTO

Para el cálculo de este presupuesto, se considera que es llevado enteramente acabo por un único trabajador, el ingeniero autor de esta memoria (sin tener en cuenta al tutor universitario pese a su colaboración, en mayor o menor medida, en las diferentes partes de este). Se divide el presupuesto, en diferentes categorías en función de la naturaleza de cada una, con el fin de aportar una mayor transparencia y facilitar el entendimiento de este.

9.1 Coste material

Es el derivado de todos aquellos componentes que fue necesario adquirir para la realización de este proyecto.

Material	Coste unitario (€)	Unidades	Coste (€)
Servomotor MG996R	5,99	4	23,96
Servomotor SG90	3,50	4	14,00
Portapilas 4 pilas AA	5,80	1	5,80
Pila AA (LR6)	0,25	4	1,00
Servo Driver PWM PCA9685	8,99	1	8,99
Módulo Bluetooth	9,99	1	9,99
Microcontrolador Arduino DUE	35,00	1	35,00
Cables Dupont	0,08	20	1,60
Tornillo Allen M3x20mm	0,03	10	0,30
Tuerca hex. autoblocante M3	0,02	10	0,20
Coste total			100,84

9.2 Coste relativo a licencias

Es el derivado de adquirir las licencias necesarias para la utilización del software empleado a lo largo del desarrollo de este proyecto.

Licencia	Coste (€)
Catia V5R20	11.200,00
Matlab	2.000,00
	13.200,00

9.3 Coste personal

Es el coste derivado del sueldo percibido por las personas que participan en este proyecto, en este caso se considera que el único personal retribuido es el autor. Para la estimación del sueldo de este, en base al *XIX Convenio colectivo del sector de*

empresas de ingeniería y oficinas de estudio técnicos [4], se establece que sea de 20€/hora.

Actividad	Coste por hora (€/h)	Horas (h)	Coste (€)
Concepción de la solución	20,00	8,00	160,00
Diseño de piezas	20,00	4,00	80,00
Simulación del prototipo	20,00	100,00	2.000,00
Diseño de control	20,00	70,00	1.400,00
Montaje y puesta a punto	20,00	16,00	320,00
Coste total (sin IVA)			3.960,00
Coste total (IVA incluido)			4.791,60

9.4 Coste total

Una vez estipulados los diferentes costes del proyecto, se muestran de manera conjunta en la tabla siguiente, en la que se puede observar el coste final del proyecto a la empresa que quisiera adquirirlo.

Área	Coste (€)
Material	100,84
Personal	4.791,60
Licencias	13.200,00
	18.092,44

Notar que estos presupuestos están sujetos a modificaciones que se puedan acordar entre las partes involucradas como el uso de una licencia ya en propiedad o la utilización de material disponible.

10 PLIEGO DE CONDICIONES

10.1 Definición y alcance del pliego

El alcance de este proyecto se extiende a la ejecución de todas las prescripciones técnicas y trabajos que lo integran y que han sido expuestos de manera detallada a lo largo de la memoria.

El objeto de este documento es establecer las condiciones técnicas mínimas que debe cumplir el desarrollo e implementación de un brazo robótico de 6 grados de libertad para la elaboración de tareas de Pick & Place. Al tratarse de un proyecto académico, con fines educativos, este apartado no será tan extenso ni incluirá especificaciones respecto a las diferentes condiciones necesarias para su puesta en marcha en, por ejemplo, una línea de producción real dentro de un proceso productivo. Por ello se centra principalmente en las fases de modelado, programación, montaje y pruebas de funcionamiento real.

10.2 Condiciones generales

Son de obligado cumplimiento todas las normas y requerimientos que con carácter general se observan durante proyectos de la misma índole. En concreto, con el fin de velar por la seguridad tanto del proyectista, a lo largo del proceso de diseño e implementación, así como de aquellos que puedan llevar acabo modificaciones futuras o puestas en marcha del proyecto; es mandatorio especificar el obligado cumplimiento de lo establecido en el Reglamento Electrotécnico de Baja Tensión por parte de todos los componentes de carácter electrónico o eléctrico. Es este el marco de referencia sobre el cual se establece la normativa a aplicar por el hecho de trabajar siempre con un suministro eléctrico <1000 V de CA y <1500 V de CC.

10.3 Condiciones particulares

10.3.1 Condiciones facultativas

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

- Conocer la normativa aplicable.
- Conocer el proyecto en todas sus partes.
- Presencia o localización de los responsables durante la ejecución de las diferentes fases del proyecto.
- Obligación de disponer un documento donde se reflejen las indicaciones, aclaraciones o modificaciones del proyecto.
- Disposición de cuantos medios auxiliares fuese necesario para garantizar el correcto desarrollo del proyecto (en caso de que no fuera posible acceder al material proporcionado por la universidad, el contratista adquirirá lo que precise siendo incluido en el coste de material).

- Derecho a recibir los pagos comprometidos (en caso de que los hubiera) en las fechas pactadas.
- Derecho a percibir compensación económica por los trabajos realizados no especificados en los documentos del proyecto y necesarios para la correcta ejecución.

En lo que a la dirección facultativa se refiere (Tutor de la Universidad), se destacan las siguientes obligaciones y facultades:

- Supervisión de aquellos aspectos del proyecto que puedan afectar a la fiabilidad, calidad y seguridad durante la ejecución del proyecto.
- Encontrarse presente en los momentos del desarrollo del proyecto que se convenga.
- Involucrarse en la aportación de soluciones técnicas a problemas no previstos durante la ejecución.
- Cursar las ampliaciones de proyecto necesarias en función de las modificaciones introducidas sobre las soluciones iniciales.

10.3.2 Condiciones técnicas

En lo relativo al uso de todos aquellos aparatos electrónicos y eléctricos supeditados al REBT mencionado anteriormente, deberá ser complementado por la información técnica especificada por cada fabricante en cuestión. Siguiendo en todo momento aquellas recomendaciones de uso especificadas en las hojas técnicas o datasheets de los componentes.

Todas las piezas cuya realización se basa en fabricación 3D o impresión aditiva, deben seguir las normas establecidas para la aplicación de dichas tecnologías y su impresión se debe realizar utilizando material apropiado y siempre bajo la supervisión de personal cualificado.

Los requisitos de hardware son:

- Sistema operativo Windows 10 o superior
- Matlab 2018b o superior
- Catia V5R20 o superior
- Arduino IDE

En lo referente al hardware, para poder realizar cada una de las diferentes fases del proyecto de manera fluida, se recomienda contar al menos con las siguientes especificaciones (es posible que sin cumplir todas ellas se puedan realizar las diferentes actividades requeridas, pero es posible que el proceso no sea todo lo óptimo que debería).

- Procesador Intel Core i5 o superior.
- Memoria RAM 8GB o superior.
- Almacenamiento libre mínimo 30 GB.

- Tarjeta gráfica GeForce GTX 750 o superior.

10.3.3 Condiciones económicas y legales

Dada la naturaleza docente del proyecto, que como se ha especificado en varias ocasiones a lo largo de la memoria tiene como único objeto el desarrollo de las capacidades ingenieriles del contratista dentro del marco de la realización del Trabajo de Fin de Máster, así como el de aquellos que en el futuro decidan utilizar algún procedimiento explicado en este estudio o una modificación de este, se considera irrelevante el establecimiento de unas condiciones económicas o legales.

11 BIBLIOGRAFÍA

[1] Raúl Rojas. *Omnidirectional Control*. Freie Universität Berlin 18 May 2005.

[2] Antonio Barrientos, Luis Felipe Penín, Carlos Balaguer y Rafael Aracil. *Fundamentos de Robótica. 2ª Edición.* (McGraw Hill). (2007). España.

[3] L. Huang, Y. S. Lim, David Li and Christopher E. L. Teoh. *Design and Analysis of a Four-wheel Omnidirectional Mobile Robot*. 2nd International Conference on Autonomous Robots and Agents. December 13-15, 2004 Palmerston North, New Zealand.

[4] BOE. *Resolución de 7 de octubre de 2019, de la Dirección General de Trabajo, por la que se registra y publica el XIX Convenio colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos.*

12 ANEXO

12.1 Código de programación

12.1.1 Calibración de servos

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x40);

unsigned int pos0=60; //ancho de pulso en cuentas para pocicion 0°
unsigned int pos180=600; //ancho de pulso en cuentas para la pocicion 180°

void setup(){
  Serial.begin(9600);
  servos.begin();
  servos.setPWMFreq(50); //Frecuecia PWM de 50Hz o T=20ms
}

void loop() {
  for (int duty = pos0; duty < pos180; duty=duty+5){
//     for (int n=0; n<6; n=n+1){ //Descomentar si se quieren calibrar todos a la vez (+ rápido - precisión)
       servos.setPWM(3,0,duty); // Los argumentos son (Canal PWM, Flanco Subida, Flanco Bajada)
       Serial.println(duty);  //Visualización de los valores por el Monitor Serie
       delay(250);
//     }
  }
  delay(1000);
  for (int duty = pos180; duty > pos0; duty=duty-5) {
//    for (int n=0; n<6; n=n+1){
     servos.setPWM(3,0,duty);
     Serial.println(duty);
     delay(250);
//  }
  }
  delay(1000);
}
```

12.1.2 Configuración Bluetooth

La configuración Bluetooth se ha realizado de dos maneras diferentes, pero igualmente válidas, debido a unos problemas iniciales con el módulo HC-06. Con este primer código, basta con elegir valores de los comandos AT deseados y subirlo a Arduino con el módulo conectado. Cuando el LED conectado al pin 13 luzca, quiere decir que ya se ha cargado la configuración al módulo Bluetooth y este está listo para su uso.

```
const int LED = 13;
const int BTPWR = 12; //Se conecta la alimentación del módulo a este pin para que sólo se active una vez se ha cargado el programa, dejando libre la comunicación serie (Rx,TX) para ello mientras tanto

char nombreBT[12] = "BrazoRobot";
char velocidad ='4';//9600
char pin [5]= "0000";
char role = 'S';

void setup(){
  pinMode(LED, OUTPUT);
  pinMode(BTPWR, OUTPUT);

  digitalWrite(LED, LOW);
  digitalWrite(BTPWR, HIGH);

  Serial.begin(9600);

  Serial.print("AT");
  delay(1500);

  Serial.print("AT+NAME");
  Serial.print(nombreBT);
  delay(1500);

  Serial.print("AT+BAUD");
  Serial.print(velocidad);
  delay(1500);

  Serial.print("AT+ROLE");
  Serial.print(role);
  delay(1500);
```

```
  Serial.print("AT+PIN");
  Serial.print(pin);
  delay(1500);

  digitalWrite(LED, HIGH);
}

void loop(){
  }
```

La otra alternativa es usar el siguiente código, el cual se basa en un eco entre dos canales serie (estando el módulo conectado a los canales Rx1 y Tx1 del Arduino Due). Se escribe a través del monitor serie los diferentes comandos AT que se deseen, siendo posible observar la respuesta del módulo a cada uno de ellos por el mismo canal.

```
void setup(){
  Serial.begin(9600);
  Serial1.begin(9600);
}

void loop() {
  if (Serial1.available())
  {
    Serial.write(Serial1.read());
  }
  if (Serial.available())
  {
    Serial1.write(Serial.read());
  }
}
```

12.1.3 Prueba Bluetooth

```
char dato;

int ledpin0 = 4;
int ledpin1 = 5;
int ledpin2 = 6;
int ledpin3 = 7;

void setup() {
  Serial1.begin(9600);
  Serial.begin(9600);
  pinMode (ledpin0, OUTPUT);
  pinMode (ledpin1, OUTPUT);
  pinMode (ledpin2, OUTPUT);
  pinMode (ledpin3, OUTPUT);
}

void loop() {
  if (Serial1.available()>0) {
    dato=Serial1.read();
    Serial.print(dato);

    switch (dato){
      case 'a':
         digitalWrite(ledpin0, HIGH);
         digitalWrite(ledpin1, LOW);
         digitalWrite(ledpin2, LOW);
         digitalWrite(ledpin3, LOW);
         break;

      case 'b':
         digitalWrite(ledpin0, LOW);
         digitalWrite(ledpin1, HIGH);
         digitalWrite(ledpin2, LOW);
         digitalWrite(ledpin3, LOW);
         break;

      case 'c':
         digitalWrite(ledpin0, LOW);
         digitalWrite(ledpin1, LOW);
         digitalWrite(ledpin2, HIGH);
         digitalWrite(ledpin3, LOW);
```

```
        break;

    case 'd':
        digitalWrite(ledpin0, LOW);
        digitalWrite(ledpin1, LOW);
        digitalWrite(ledpin2, LOW);
        digitalWrite(ledpin3, HIGH);
        break;
   }
  }
}
```

12.1.4 Programa final

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver ServoDriver = Adafruit_PWMServoDriver(0x40);

char dato;

unsigned int incremento = 5;
unsigned int espera=15;
unsigned int duty[6]={300,300,300,300,300,400}; //Una vez ajustados los servos
sabremos su posición inicial y se cambia el valor
unsigned int maxvalue[6]={515,515,515,560,560,515};
unsigned int minvalue[6]={75,75,75,80,80,325};

int ledpin0 = 4;
int ledpin1 = 5;
int ledpin2 = 6;
int ledpin3 = 7;

void setup() {
  Serial1.begin(9600);
  Serial.begin(9600);
  ServoDriver.begin();
  ServoDriver.setPWMFreq(50);
  ServoDriver.setPWM(0,0,duty[0]);
  ServoDriver.setPWM(1,0,duty[1]);
  ServoDriver.setPWM(2,0,duty[2]);
  ServoDriver.setPWM(3,0,duty[3]);
  ServoDriver.setPWM(4,0,duty[4]);
```

```
 ServoDriver.setPWM(5,0,duty[5]);

pinMode (ledpin0, OUTPUT);
pinMode (ledpin1, OUTPUT);
pinMode (ledpin2, OUTPUT);
pinMode (ledpin3, OUTPUT);
}

void loop() {
 if (Serial1.available()>0) {
   dato=Serial1.read();
   Serial.print(dato);

   switch (dato){
    case 'a':
       digitalWrite(ledpin0, HIGH);
       digitalWrite(ledpin1, LOW);
       digitalWrite(ledpin2, LOW);
       digitalWrite(ledpin3, LOW);
       if (duty[0] < maxvalue[0]){
         duty[0] = duty[0] + incremento;
         }else{}
       ServoDriver.setPWM(0,0,duty[0]);
       delay(espera);
       break;

    case 'b':
       digitalWrite(ledpin0, LOW);
       digitalWrite(ledpin1, HIGH);
       digitalWrite(ledpin2, LOW);
       digitalWrite(ledpin3, LOW);
       if (duty[0] > minvalue[0]){
         duty[0] = duty[0] - incremento;
         }else{}
       ServoDriver.setPWM(0,0,duty[0]);
       delay(espera);
       break;
    .
    .
    .
    case 'l':
       digitalWrite(ledpin0, LOW);
       digitalWrite(ledpin1, LOW);
```

```
        digitalWrite(ledpin2, LOW);
        digitalWrite(ledpin3, HIGH);
        if (duty[5] > minvalue[5]){
          duty [5] = duty[5] - incremento;
          }else{}
        ServoDriver.setPWM(5,0,duty[5]);
        delay(espera);
        break;

    case 'z':
        digitalWrite(ledpin0, LOW);
        digitalWrite(ledpin1, LOW);
        digitalWrite(ledpin2, LOW);
        digitalWrite(ledpin3, LOW);
        delay(espera);
        break;
    }

} else {
  switch (dato){
    case 'a':
        digitalWrite(ledpin0, HIGH);
        digitalWrite(ledpin1, LOW);
        digitalWrite(ledpin2, LOW);
        digitalWrite(ledpin3, LOW);
        if (duty[0] < maxvalue[0]){
          duty[0] = duty[0] + incremento;
          }else{}
        ServoDriver.setPWM(0,0,duty[0]);
        delay(espera);
        break;

    case 'b':
        digitalWrite(ledpin0, LOW);
        digitalWrite(ledpin1, HIGH);
        digitalWrite(ledpin2, LOW);
        digitalWrite(ledpin3, LOW);
        if (duty[0] > minvalue[0]){
          duty[0] = duty[0] - incremento;
          }else{}
        ServoDriver.setPWM(0,0,duty[0]);
        delay(espera);
        break;
```

```
    .
    .
    .

    case 'l':
      digitalWrite(ledpin0, LOW);
      digitalWrite(ledpin1, LOW);
      digitalWrite(ledpin2, LOW);
      digitalWrite(ledpin3, HIGH);
      if (duty[5] > minvalue[5]){
       duty [5] = duty[5] - incremento;
       }else{}
      ServoDriver.setPWM(5,0,duty[5]);
      delay(espera);
      break;

    case 'z':
      digitalWrite(ledpin0, LOW);
      digitalWrite(ledpin1, LOW);
      digitalWrite(ledpin2, LOW);
      digitalWrite(ledpin3, LOW);
      delay(espera);
      break;
    }
  }
}
```

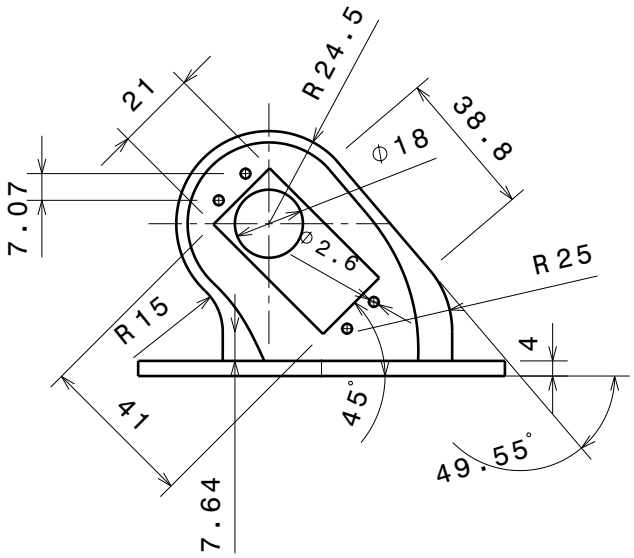
12.2 Planos

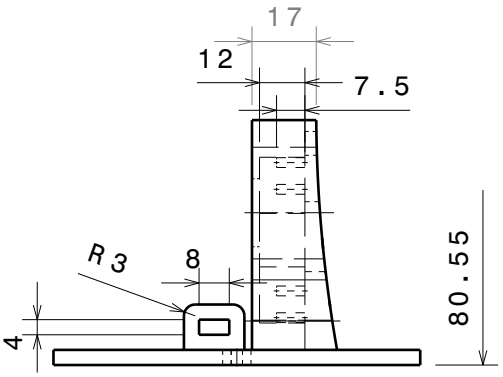
Al tratarse de documentos en formato PDF generados a partir de un archivo CATDrawing de CATIA, se adjuntan en dicho formato al término de esta memoria con el fin de no perder resolución de imagen.

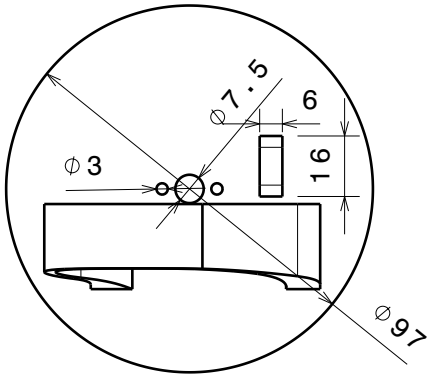
12.3 Datasheets

Al tratarse de documentos PDF descargados de las páginas web de los diferentes proveedores, se adjuntan en dicho formato al final de esta memoria con el fin de no perder resolución de imagen.

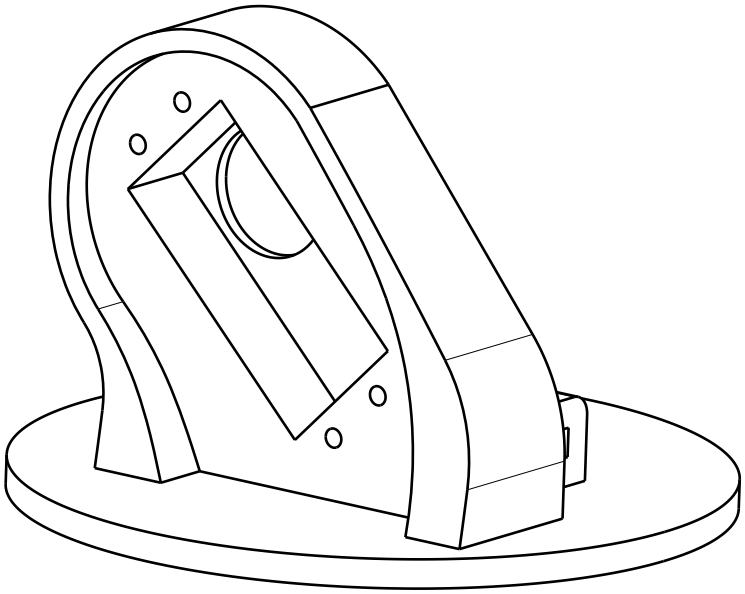
PLANOS





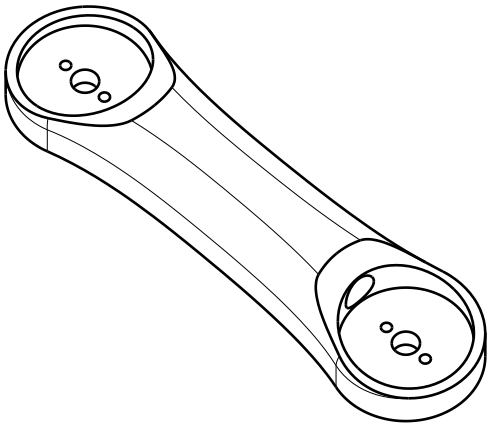


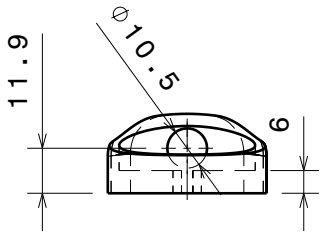
Isometric View 1:1

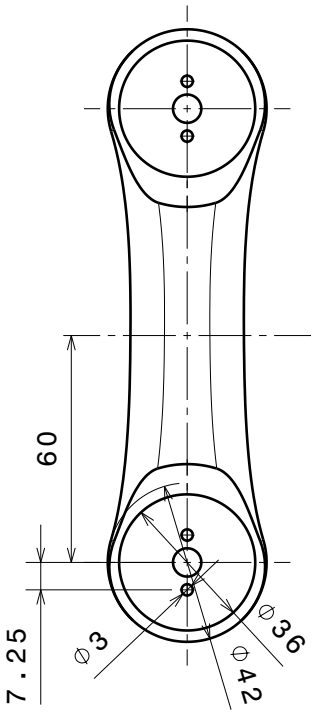


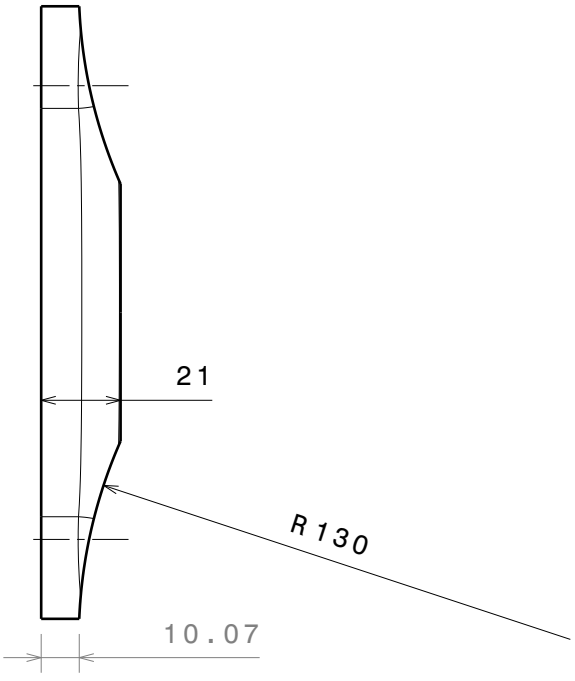
DESIGNED BY: Adrian	UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Escuela Técnica Superior de Ingeniería del Diseño	I _
DATE: 12/05/2020			H _
CHECKED BY: XXX			G _
DATE: XXX			F _
SIZE A4	BASE GIRATORIA		E _
			D _
			C _
			B _
			A _

SCALE	WEIGHT (kg)	Author:	SHEET
1:2	XXX	Adrián Tham Ochoa	1/1

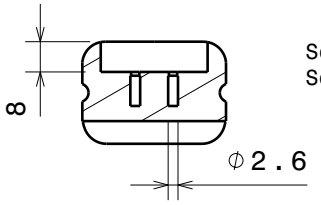




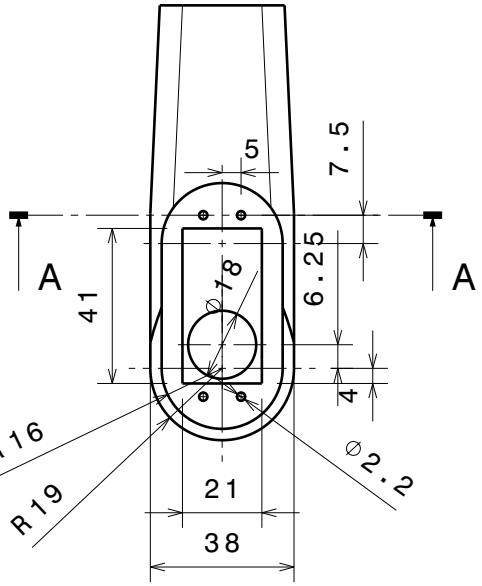


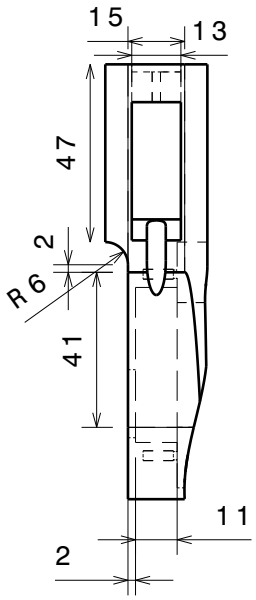


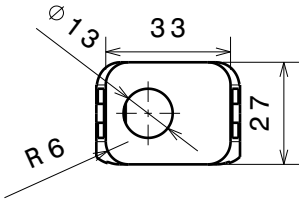
DESIGNED BY: Adrian DATE: 12/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4	BRAZO 1		E _
			D _
			C _
SCALE 1:2	WEIGHT (kg) XXX	Author: Adrián Tham Ochoa	SHEET 1/1
			B _
			A _



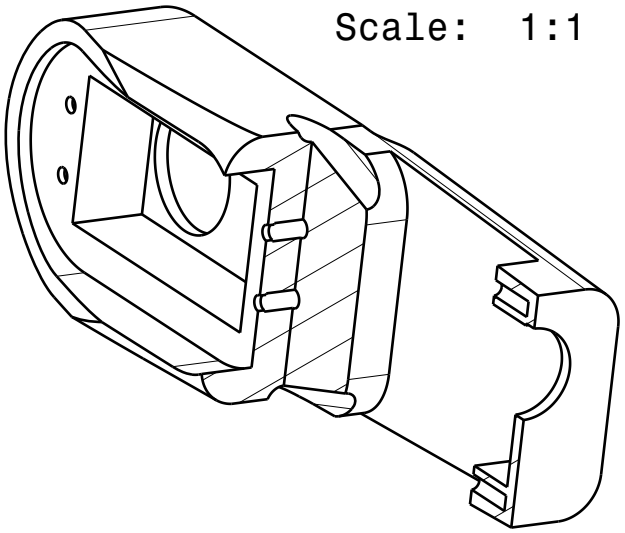
Section view A-A
Scale: 1:2



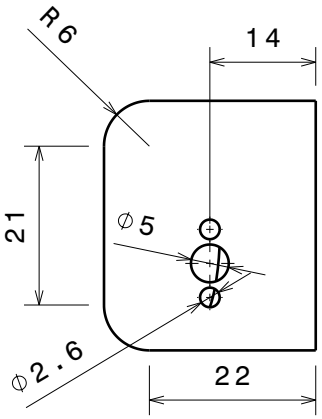


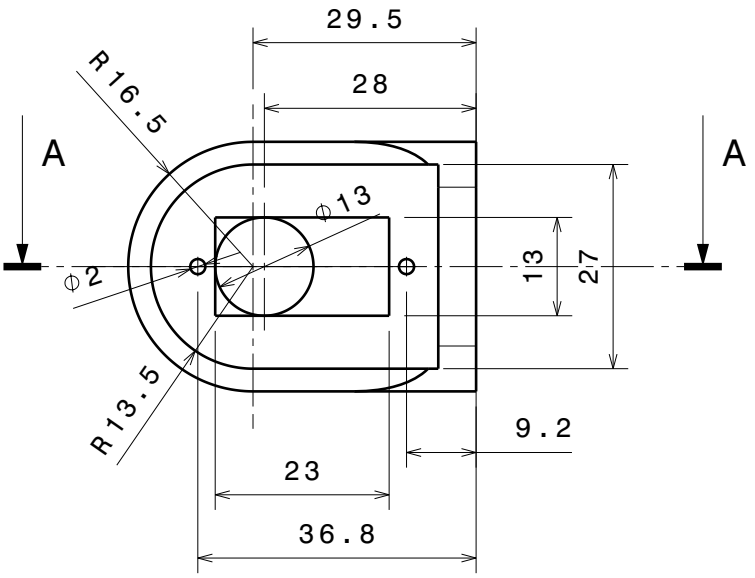


Isometric view
Scale: 1:1

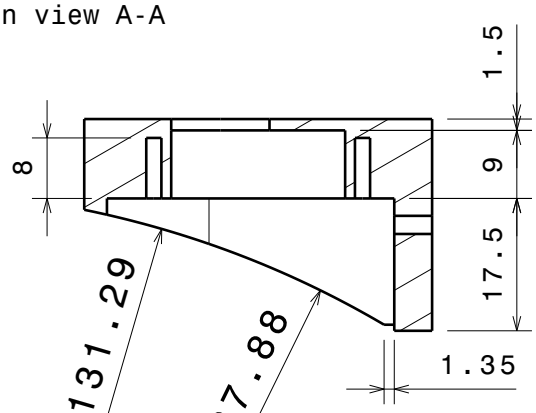


DESIGNED BY: Adrian DATE: 12/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4	BRAZO 2		E _
			D _
SCALE 1:2	WEIGHT (kg) XXX	SHEET 1/1	C _
	Author: Adrián Tham Ochoa		B _
			A _

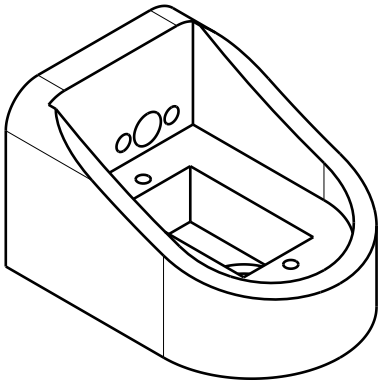




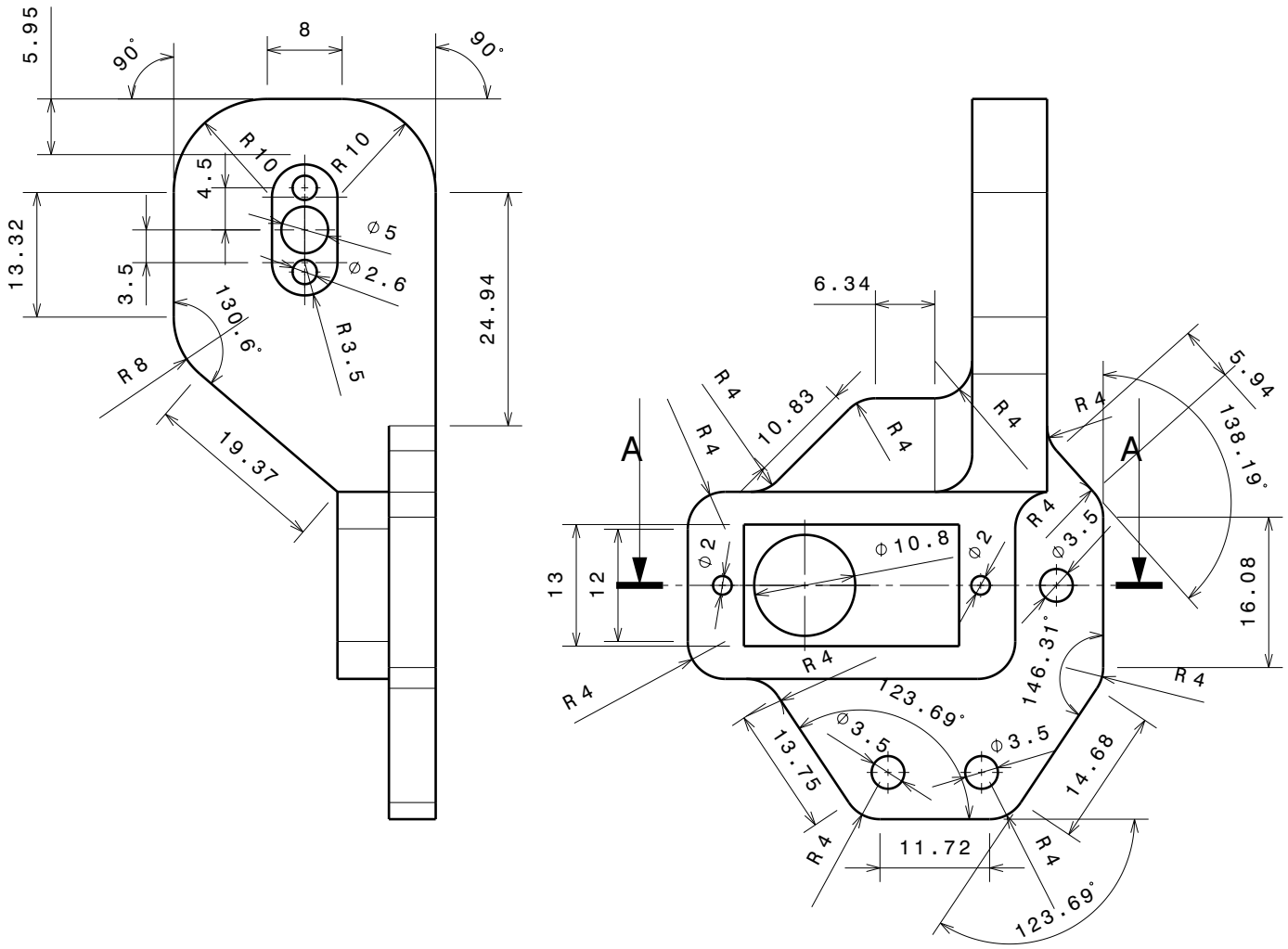
Section view A-A



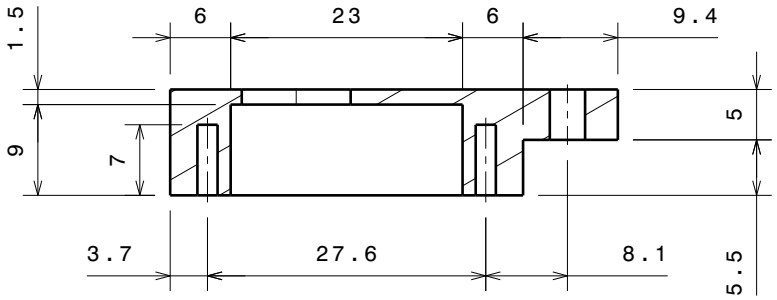
Isometric view



DESIGNED BY: Adrian DATE: 12/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4	BRAZO 3		E _
			D _
SCALE 1:1	WEIGHT (kg) XXX	Author: Adrián Tham Ochoa	SHEET 1/1
			C _
			B _
			A _

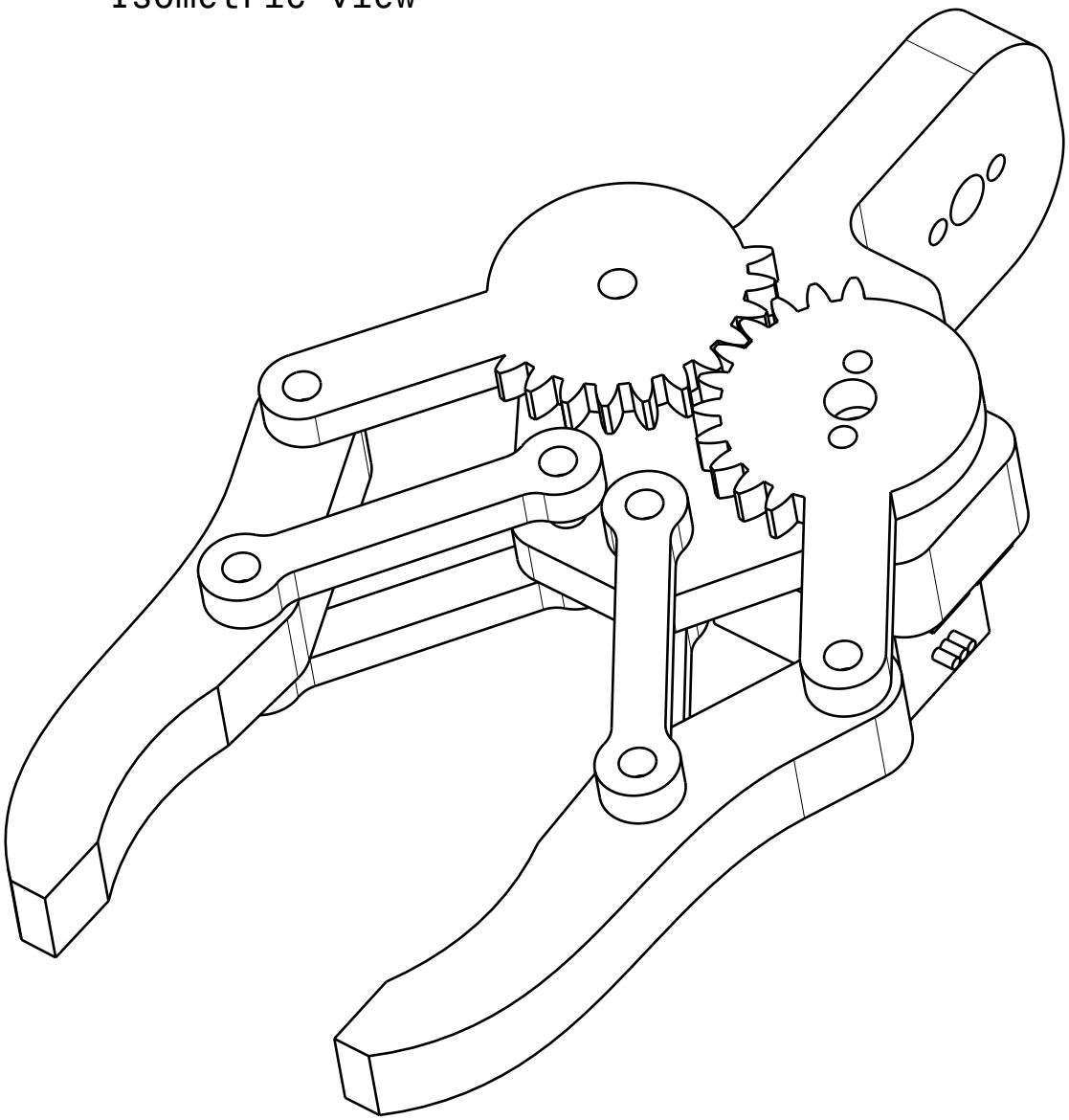


Section View A-A

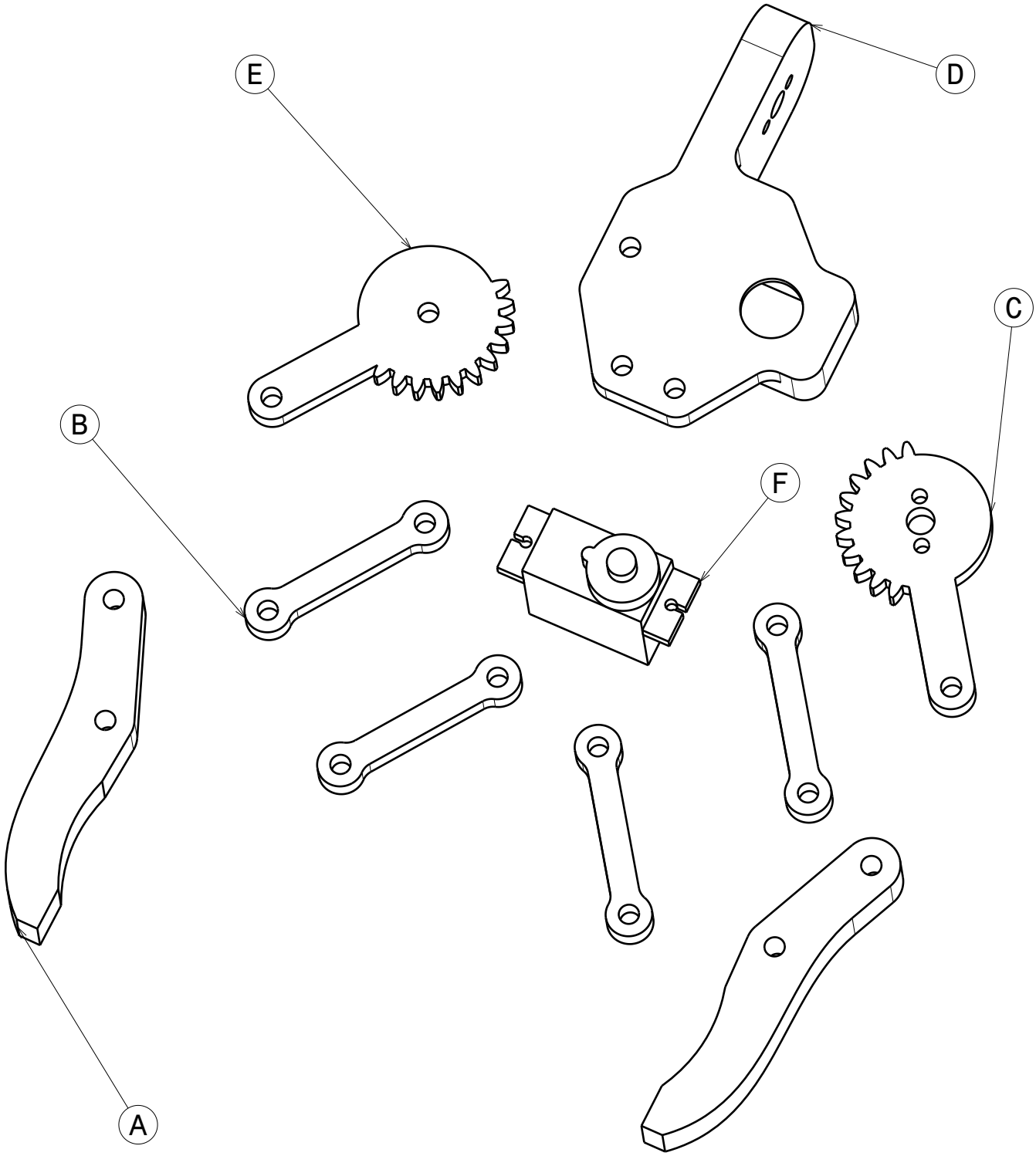


DESIGNED BY: Adrian DATE: 12/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4	BASE GRIPPER		E _
			D _
SCALE 4:3	WEIGHT (kg) XXX	Author: Adrián Tham Ochoa	SHEET 1/1
			C _
			B _
			A _

Isometric view



DESIGNED BY: Adrian DATE: 13/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _ H _ G _ F _ E _ D _ C _ B _ A _
CHECKED BY: XXX DATE: XXX			
SIZE A4	VISTA ISOMÉTRICA GRIPPER		
SCALE 3:2	WEIGHT (kg) XXX	Author: Adrián Tham Ochoa	SHEET 1/1

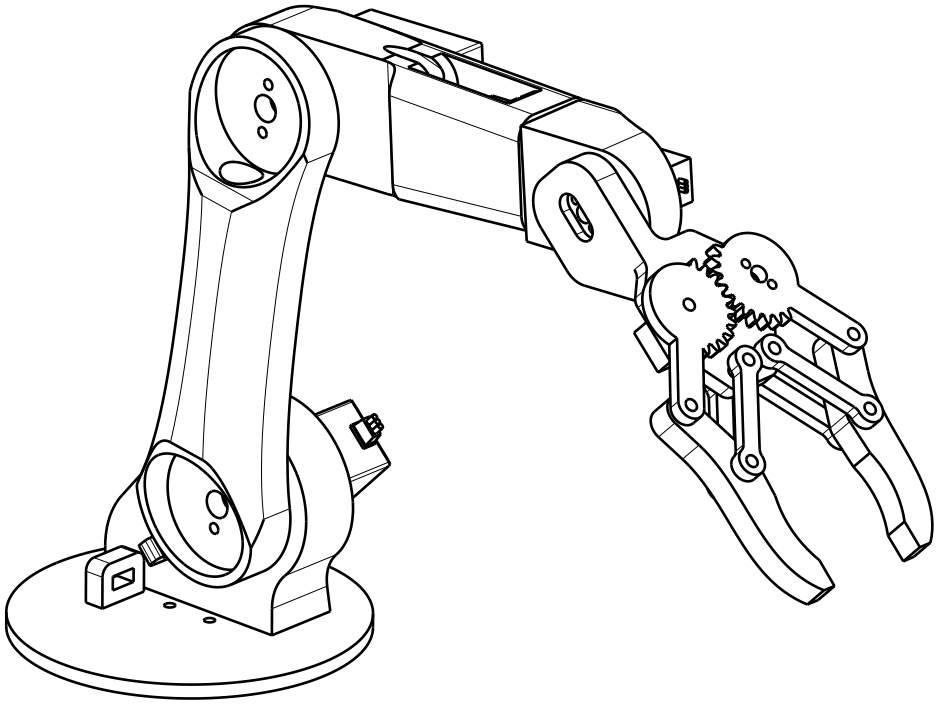


DESIGNED BY: Adrian	
DATE: 13/05/2020	
CHECKED BY: XXX	
DATE: XXX	
SIZE A4	EXPLOSIONADO GRIPPER
SCALE 1:1	WEIGHT (kg) XXX
Author: Adrián Tham Ochoa	SHEET 1/1

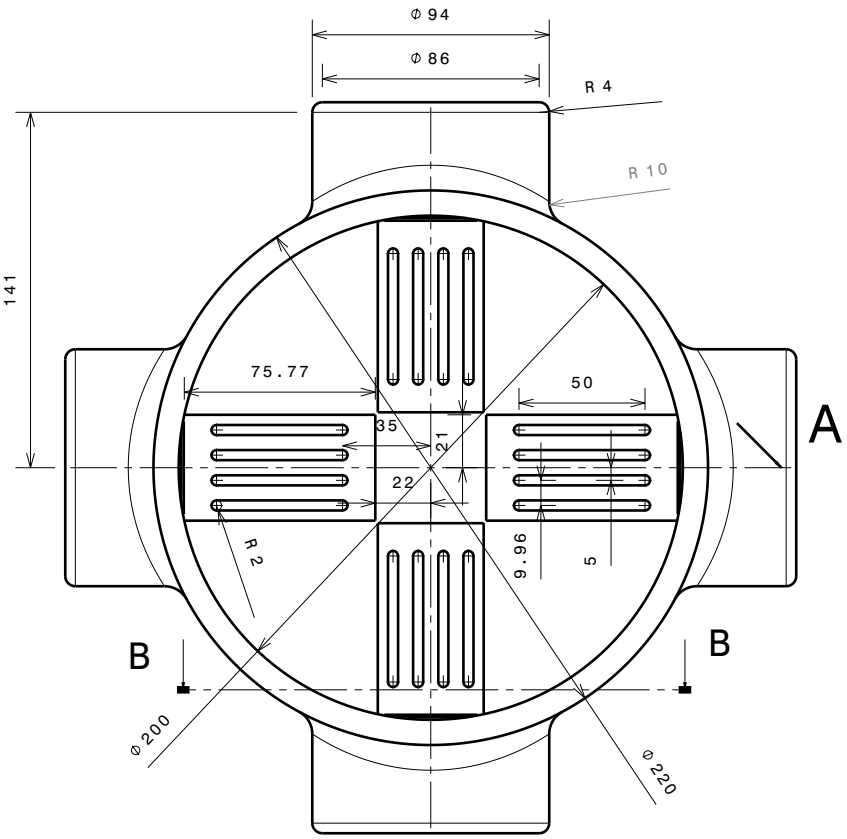
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

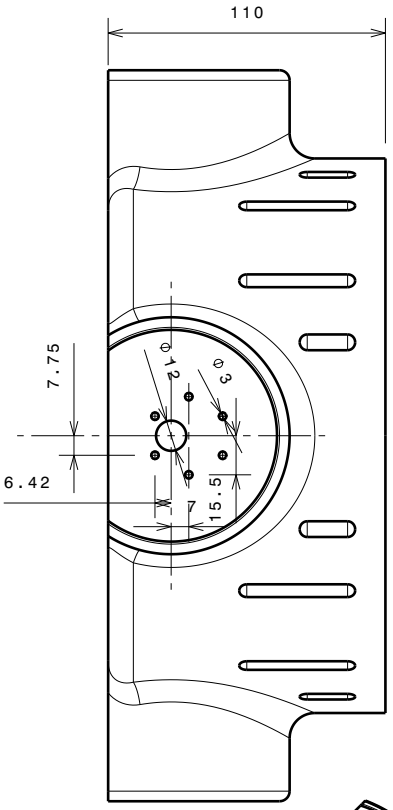
Escuela Técnica Superior de Ingeniería del Diseño

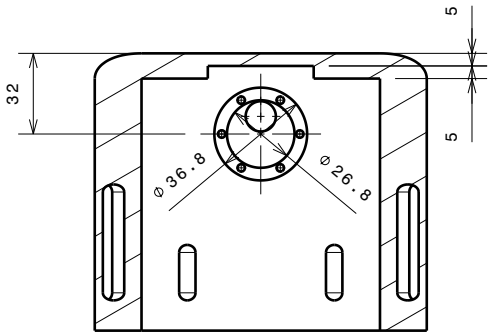
I	_
H	_
G	_
F	_
E	ENGRANAJE 2
D	BASE GRIPPER
C	ENGRANAJE 1
B	LINKER
A	PINZA



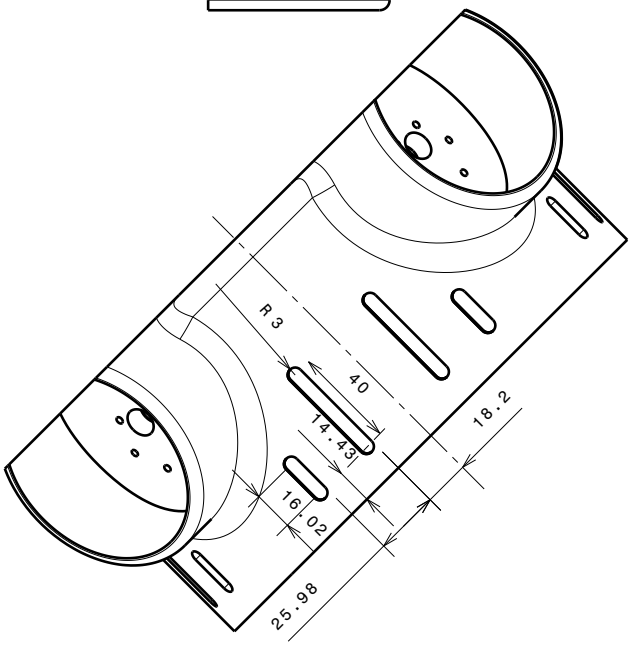
DESIGNED BY: Adrian DATE: 13/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4			E _
	VISTA ISOMÉTRICA BRAZO ROBOT		D _
SCALE 1:2	WEIGHT (kg) XXX	Author: Adrián Tham Ochoa	SHEET 1/1
			C _
			B _
			A _





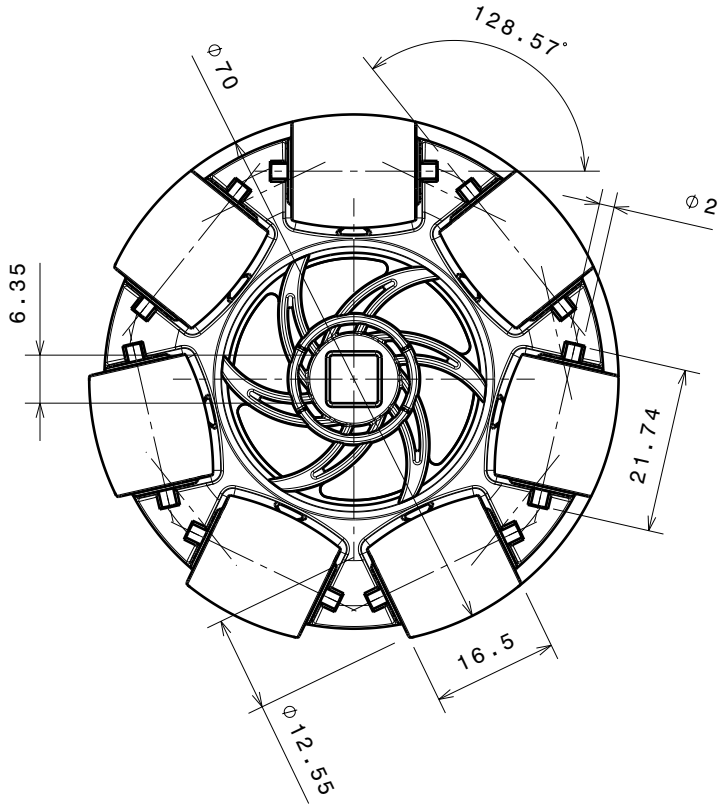


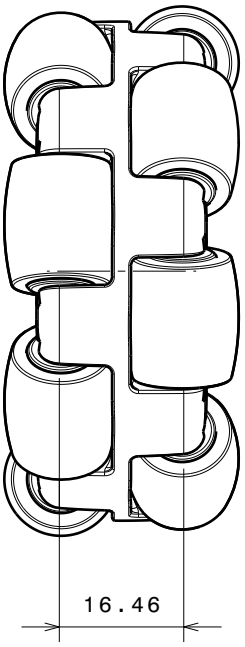
Section view B-B

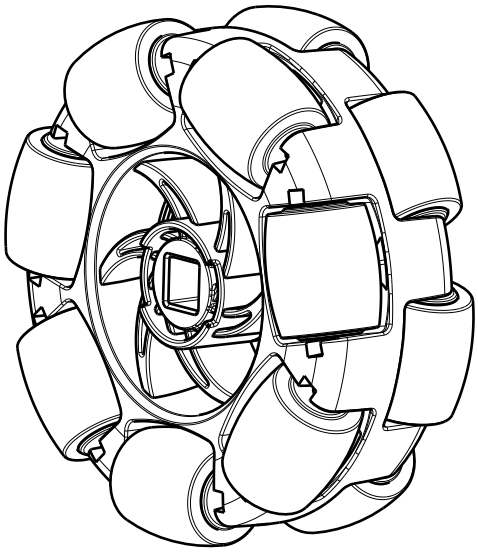


Auxiliary view A

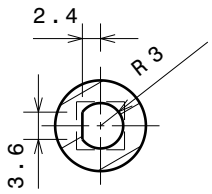
DESIGNED BY: Adrian DATE: 13/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4	CHASIS PLATAFORMA OMNIDIRECCIONAL		E _
			D _
SCALE 1:3	WEIGHT (kg) XXX	SHEET 1/1	C _
Author: Adrián Tham Ochoa			B _
This drawing is our property; it can't be reproduced or communicated without our written agreement.			A _



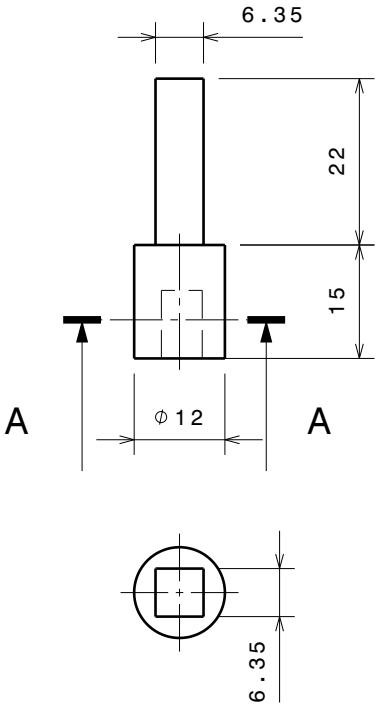


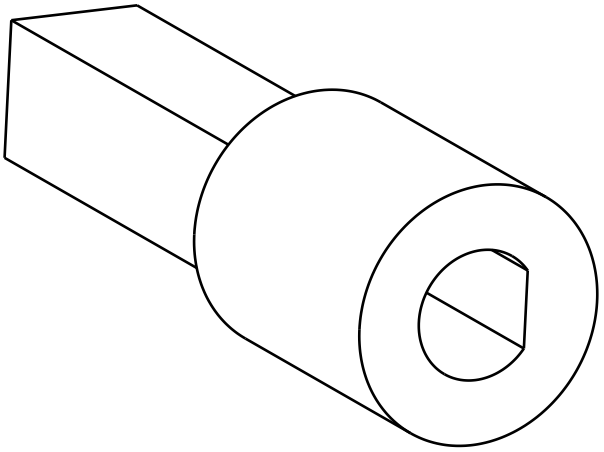


DESIGNED BY: Adrian DATE: 19/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4		RUEDA VEX OMNIDIRECCIONAL 2.75''	E _
			D _
SCALE 1:1	WEIGHT (kg) XXX	Author: Adrián Tham Ochoa	SHEET 1/1
			C _
			B _
This drawing is our property; it can't be reproduced or communicated without our written agreement.			A _



Section view A-A
Scale: 1:1

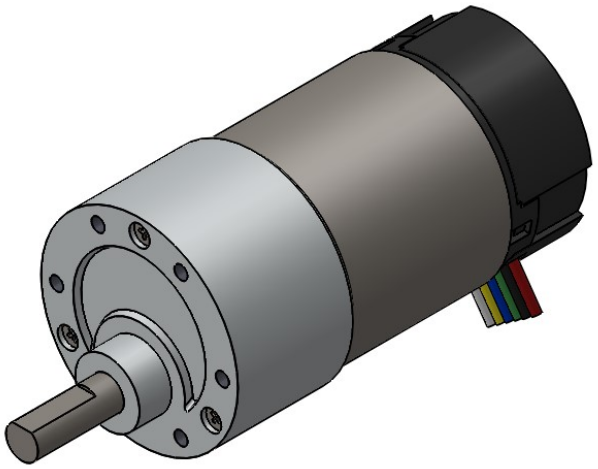


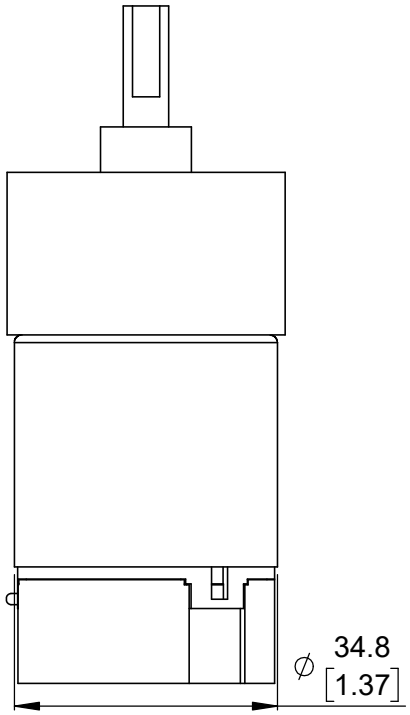


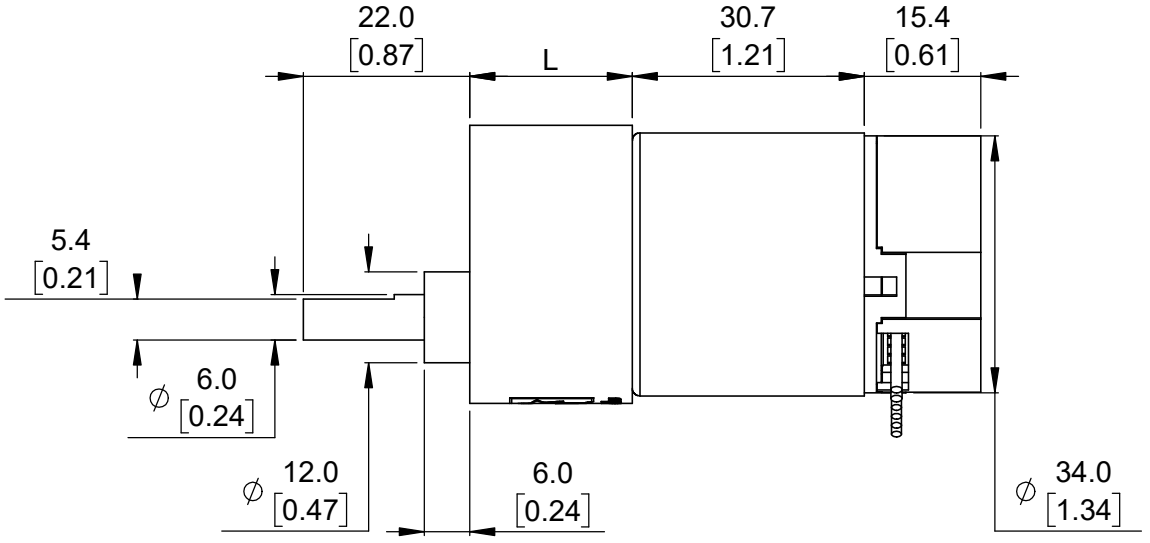
Isometric view
Scale: 3:1

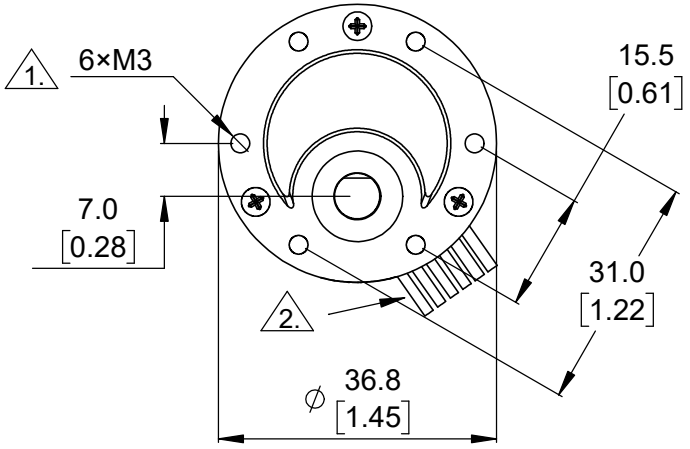
DESIGNED BY: Adrian DATE: 19/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4	ACOPLE MOTOR-RUEDA		E _
			D _
SCALE 1:1	WEIGHT (kg) XXX	Author: Adrián Tham Ochoa	SHEET 1/1
			C _
			B _
			A _

Gear ratio	L
6.3:1, 10:1	20.0 mm [0.79 in]
19:1, 30:1	21.5 mm [0.87 in]
50:1, 70:1	24.0 mm [0.94 in]
100:1, 131:1, 150:1	26.5 mm [1.04 in]









1. Threaded to a depth of 3.0 mm [0.12 in]; exceeding this depth can damage gears in the gearbox.

2. Leads are approximately 200 mm [8 in] long and are terminated by a 1×6 female header with a 2.54 mm [0.1 in] pitch.

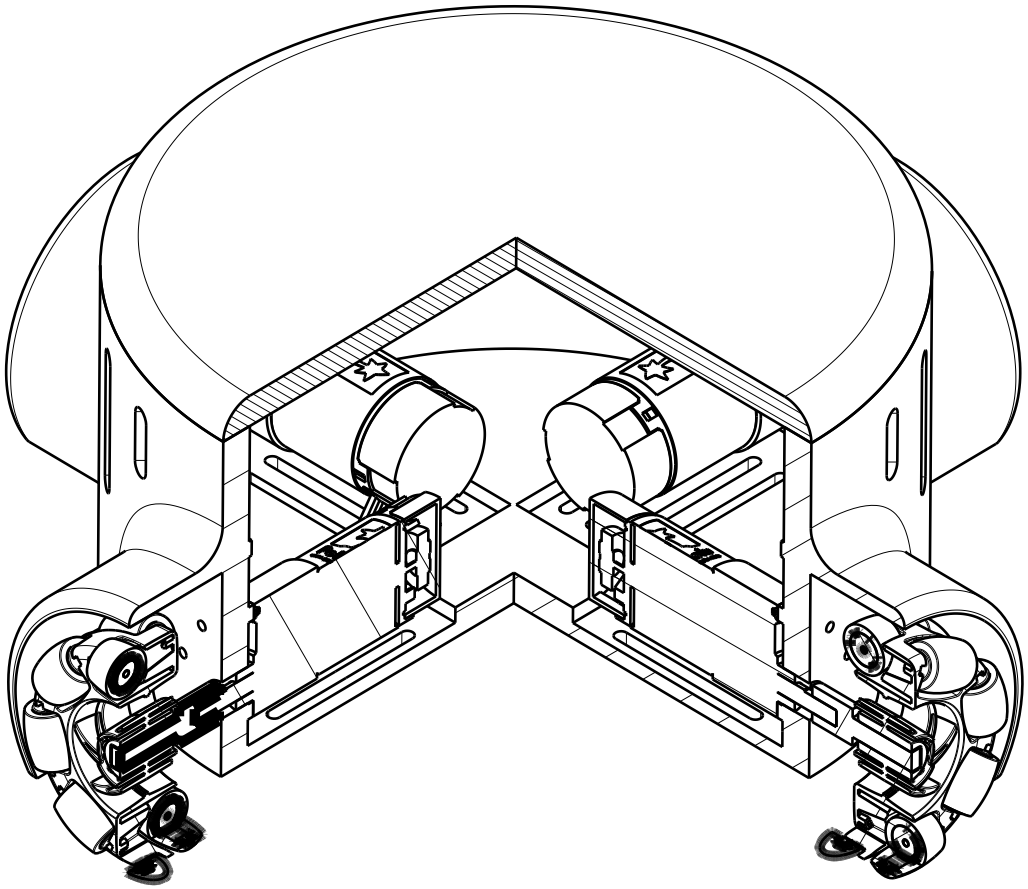
3. To get the specified scale, select 100% in print settings.

Scale: 1:1

https://www.pololu.com/category/116/37d-mm-gearmotors

Name: 37D mm Metal Gearmotors with 64 CPR Encoder	Item number: 2828, 4691-4699, 4751-4758
Drawing date: 10 January 2020	Dev code: —
Units: mm [in]	Material: Mixed

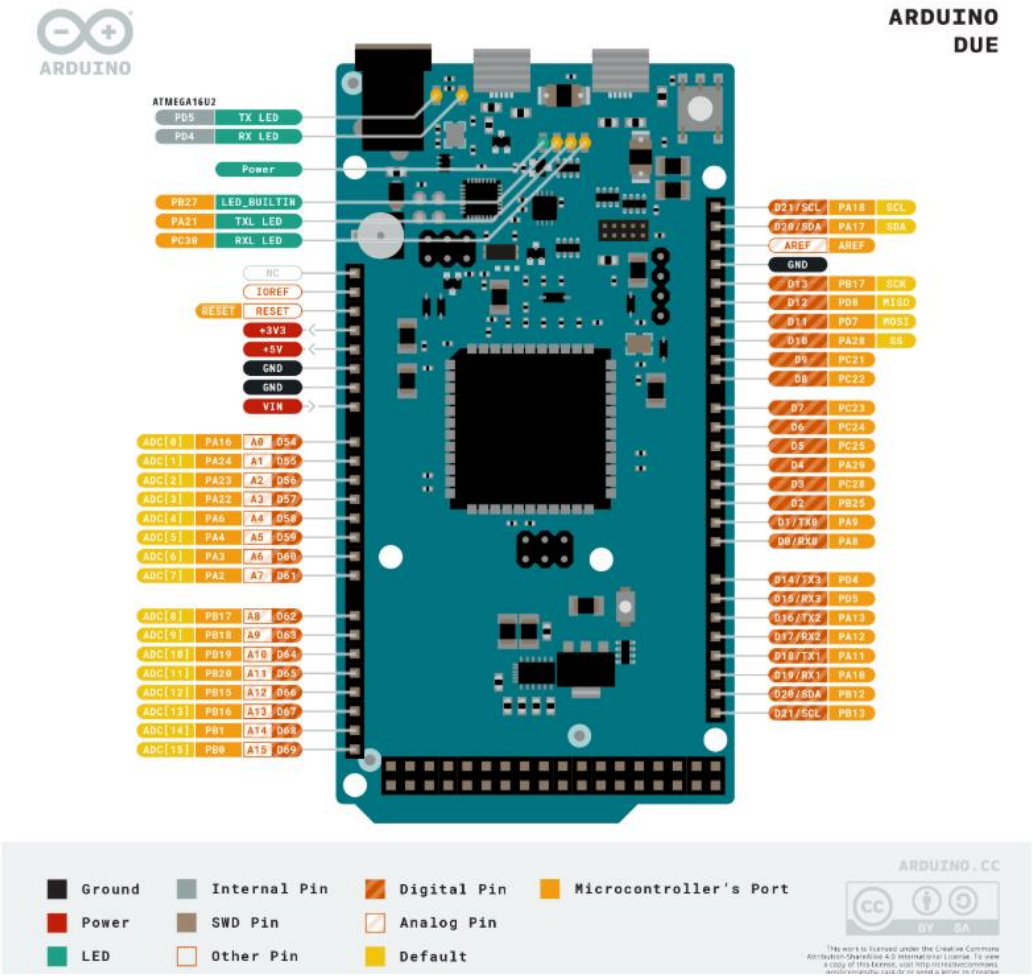
Pololu Robotics & Electronics



DESIGNED BY: Adrian	UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Escuela Técnica Superior de Ingeniería del Diseño	I _
DATE: 20/05/2020			H _
CHECKED BY: XXX			G _
DATE: XXX			F _
SIZE A4	PLATAFORMA OMNIDIRECCIONAL		E _
			D _
SCALE 1:2	WEIGHT (kg) XXX	Author: Adrián Tham Ochoa	SHEET 1/1
			C _
			B _
			A _

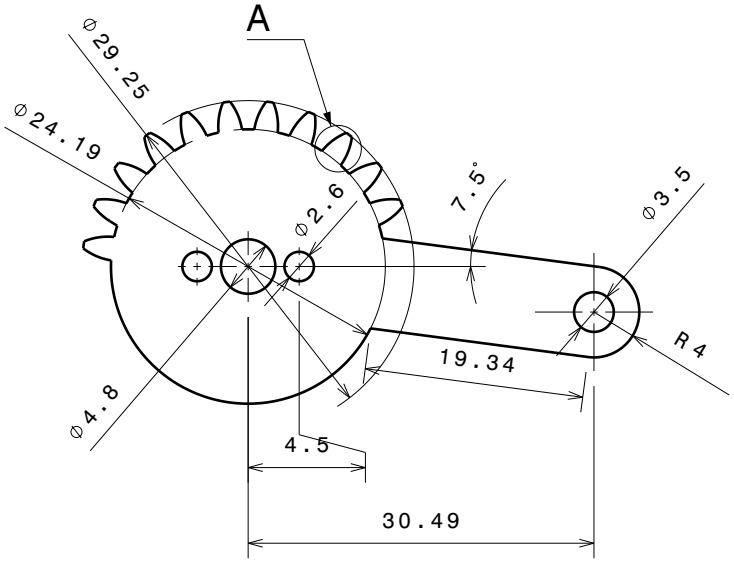
DATASHEETS

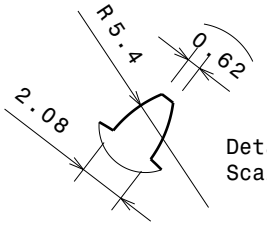
Pinout Diagram



Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

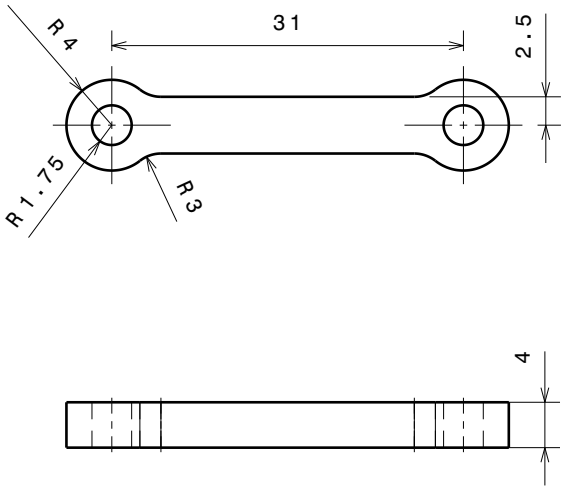
Engranaje
Scale: 3:2



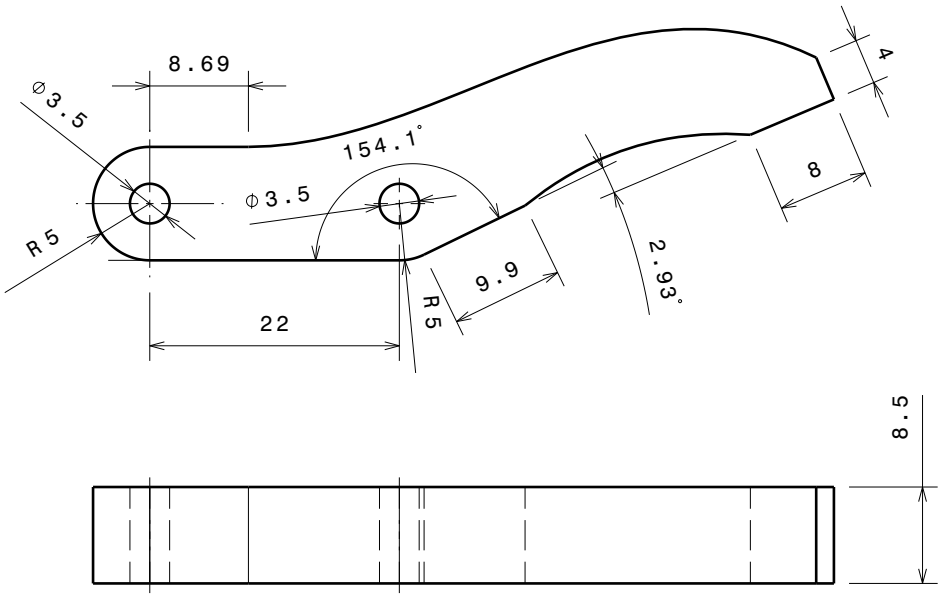


Detail A
Scale: 3:1

Linker
Scale: 3:2



Pinza
Scale: 3:2



DESIGNED BY: Adrian DATE: 13/05/2020	UNIVERSITAT POLITÈCNICA DE VALÈNCIA Escuela Técnica Superior de Ingeniería del Diseño		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
			F _
SIZE A4	COMPONENTES DEL GRIPPER		E _
			D _
SCALE 3:2	WEIGHT (kg) XXX	SHEET 1/1	C _
	Author: Adrián Tham Ochoa		B _
This drawing is our property; it can't be reproduced or communicated without our written agreement.			A _

HC-06 Data Sheet

Bluetooth to Serial Port Module

1、Overview

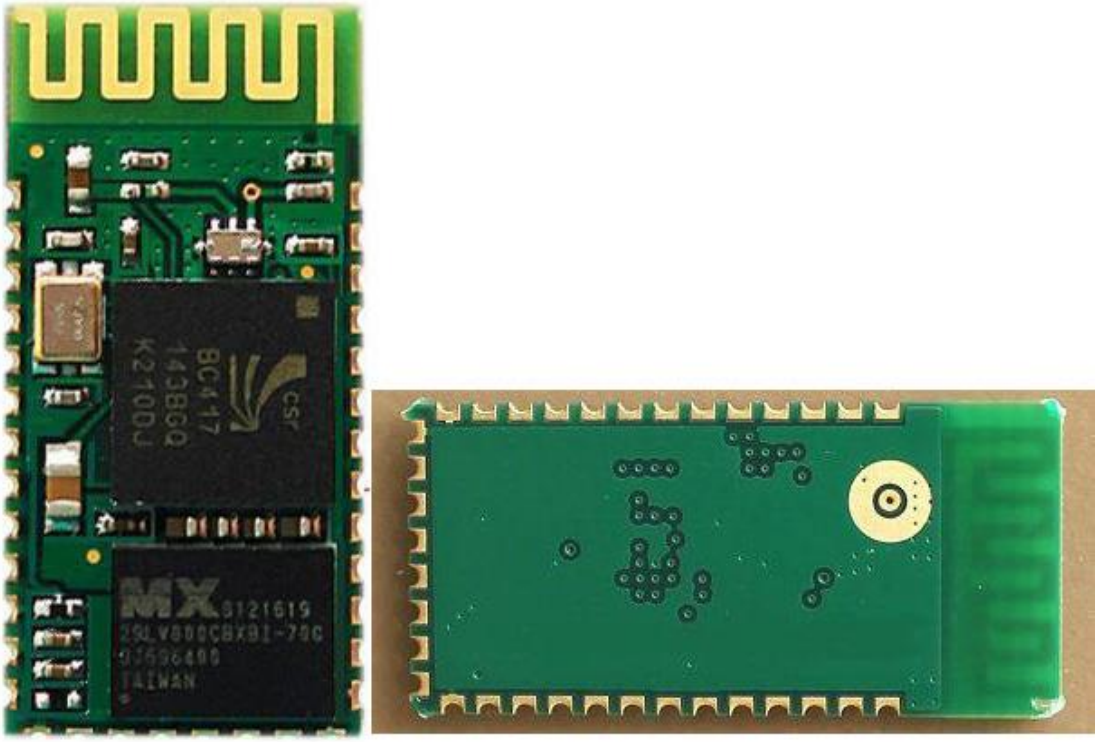
HC-06 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.

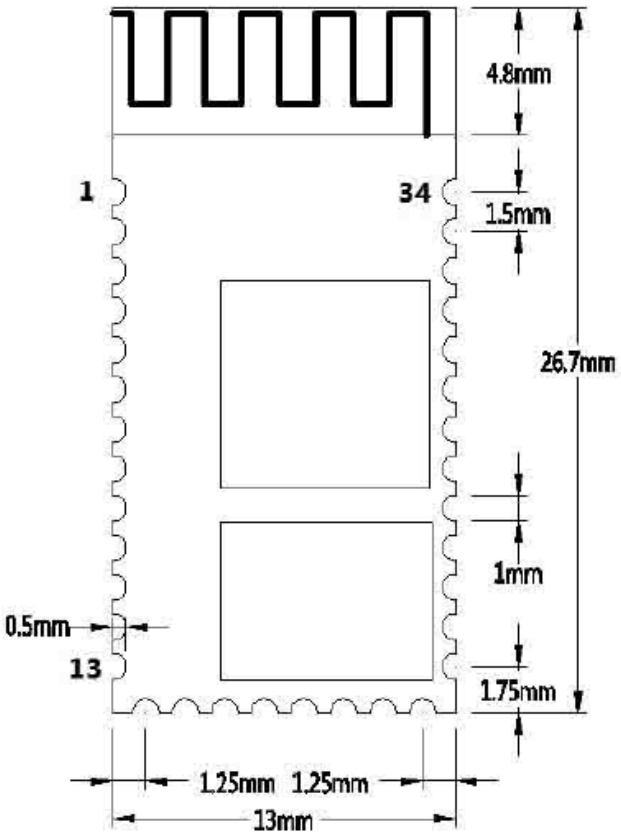
Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore BC417143 chip. It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

2、Feature

- Sensitivity (Bit error rate) can reach -80dBm,The change range of output's power: -4 - +6dBm.
- Has an EDR module; and the change range of modulation depth: 2Mbps - 3Mbps.
- Has a build-in 2.4GHz antenna; user needn't test antenna.
- Has the external 8Mbit FLASH
- Can work at the low voltage (3.1V~4.2V). The current in pairing is in the range of 30～40mA.
- The current in communication is 8mA.
- Standard HCI Port (UART or USB)
- USB Protocol: Full Speed USB1.1, Compliant With 2.0
- This module can be used in the SMD.
- It's made through RoHS process.
- The board PIN is half hole size.
- Has a 2.4GHz digital wireless transceiver.
- Bases at CSR BC04 Bluetooth technology.
- Has the function of adaptive frequency hopping.
- Small (27mm×13mm×2mm)
- Peripherals circuit is simple.
- It's at the Bluetooth class 2 power level.
- Storage temperature range: -40 ℃ - 85℃， work temperature range: -25 ℃ - +75℃
- Any wave inter Interference: 2.4MHz， the power of emitting: 3 dBm.
- Bit error rate: 0. Only the signal decays at the transmission link, bit error may be produced. For example, when RS232 or TTL is being processed, some signals may decay.

3、Product's picture

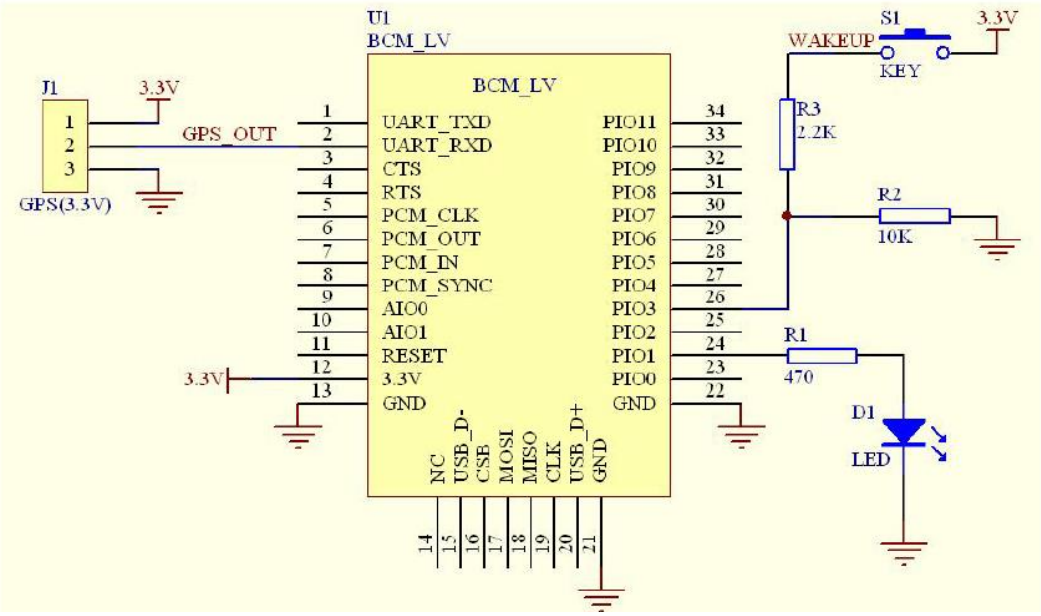


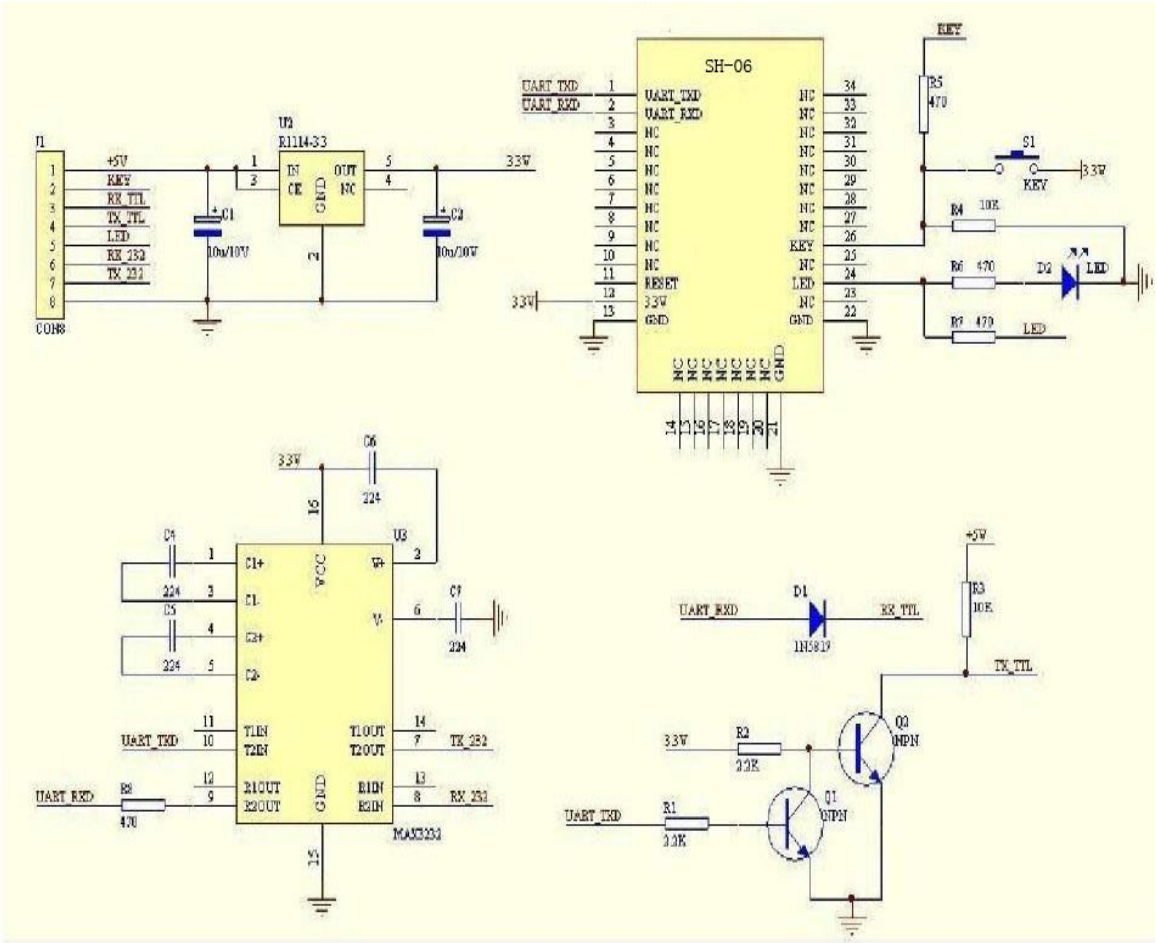


4、Application fields

- Bluetooth Car Handsfree Device
- Bluetooth GPS
- Bluetooth PCMCIA , USB Dongle
- Bluetooth Data Transfer

5、Block diagram

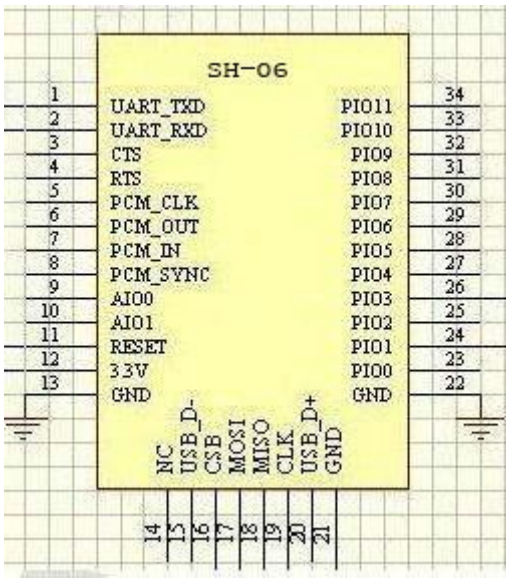




HC-06 master device has a function of remembering the last paired slave device. As a master device, it will search the last paired salve device until the connection is built. But if the WAKEUP bottom is pressed, HC-06 will lose the memory and research the new

slave device.

6、 PINs description



PIN Name	PIN	Pad type	Description	Note
UART_TX	1	CMOS output, Tri-stable with weak internal pull-up	UART Data output	
UART_RX	2	CMOS input with weak internal pull-down	UART Data input	
UART_CTS	3	CMOS input with weak internal pull-down	UART clear to send, active low	
UART_RTS	4	CMOS output, tri-stable with weak internal pull-up	UART r qu st to send, active low	
PCM_CLK	5	Bi-Directional		

PCM_OUT	6	CMOS output		
PCM_IN	7	CMOS Input		
PCM_SYNC	8	Bi-Directional		
AIO0	9	Bi-Directional		
AIO1	10	Bi-Directional		
RESETB	11	CMOS Input with RESETB 11 weak intemal pull-down		
VCC	12	3.3V		
GND	13	VSS	Ground pot	
1V8	14	VDD	Integrated 1.8V (+) supply with On-chip linear regulator output within 1.7-1.9V	
USB_-	15	Bi-Directional		
SPI_CSB	16	CMOS input with weak internal pull-up	Chip select for serial peripheral interface, active low	
SPI_MOSI	17	CMOS input with weak internal pull-down	Serial peripheral interface data input	
SPI_MISO	18	CMOS input with weak internal pull-down	Serial peripheral interface data Output	
SPI_CLK	19	CMOS input with weak internal	Serial peripheral interface clock	
USB_+	20	Bi-Directional		
GND	21	VSS	Ground pot	
GND	22	VSS	Ground pot	
PIO0	23	Bi-Directional RX EN	Programmable input/output line, control output for LNA(if fitted)	

PIO1	24	Bi-Directional TX EN	Programmable input/output line, control output for PA(if fitted)	LED
PIO2	25	Bi-Directional	Programmable input/output line	
PIO3	26	Bi-Directional	Programmable input/output line	KEY
PIO4	27	Bi-Directional	Programmable input/output line	
PIO5	28	Bi-Directional	Programmable input/output line	
PIO6	29	Bi-Directional	Programmable input/output line	CLK_REQ
PIO7	30	Bi-Directional	Programmable input/output line	CLK_OUT
PIO8	31	Bi-Directional	Programmable input/output line	
PIO9	32	Bi-Directional	Programmable input/output line	
PIO10	33	Bi-Directional	Programmable input/output line	
PIO11	34	Bi-Directional	Programmable input/output line	

7、AT Command

The way to the AT command mode: supply power to the module, it will enter to the AT mode if it needn't pair. The interval of command is about 1 second.

Default parameter: Baud rate:9600N81, Password:1234

1. Test communication

Send: AT (please send it every second)

Back: OK

2. Reset the Bluetooth serial baud rate
Send: AT+BAUD1
Back: OK1200
Send: AT+BAUD2
Back: OK2400
……
1---------1200
2---------2400
3---------4800
4---------9600 (Default)
5---------19200
6---------38400
7---------57600
8---------115200
9---------230400
A---------460800
B---------921600
C---------1382400
PC can't support the baud rate lager than 115200. The solution is: make the MCU have higher baud rate (lager than 115200) through programming, and reset the baud rate to low level through the AT command.
The baud rate reset by the AT command can be kept for the next time even though the power is cut off.

3. Reset the Bluetooth name
Send: AT+NAMEname
Back: OKname

Parameter name: Name needed to be set (20 characters limited)
Example:
Send: AT+NAMEbill_gates
Back: OKname
Now, the Bluetooth name is reset to be "bill_gates"
The parameter can be kept even though the power is cut off. User can see the new Bluetooth name
in PDA refresh service. (Note: The name is limited in 20 characters.)

4. change the Bluetooth pair password

Send: AT+PINxxxx
Back:OKsetpin

Parameter xxxx: The pair password needed to be set, is a 4-bits number. This command can be usedin the master and slave module. At some occasions, the master module may be asked to enter the password when the master module tries to connect the slave module (adapter or cell-phone). Only if the password is entered, the successful connection can be built. At the other occasions, the pair can be finish automatically if the master module can search the proper slave module and the password is correct.
Besides the paired slave module, the master can connect the other devices who have slave module, such as Bluetooth digital camera, Bluetooth GPS, Bluetooth serial printer etc.
Example:
Send: AT+PIN8888
Back: OKsetpin
Then the password is changed to be 8888, while the default is 1234.
This parameter can be kept even though the power is cut off.

5. No parity check (The version, higher than V1.5, can use this command)
Send: AT+PN (This is the default value)
Back: OK NONE
6. Set odd parity check (The version, higher than V1.5, can use this command)
Send: AT+PO
Back: OK ODD
7. Set even parity check(The version, higher than V1.5, can use this command)
Send: AT+PE
Back: OK EVEN

8. Set Master / Slave mode

Send: AT+ROLE=[mode]

Back: OK+ROLE:[mode]

[Mode] : S or M

AT+ROLE=M Setting to master mode

AT+ROLE=S Setting to slave mode

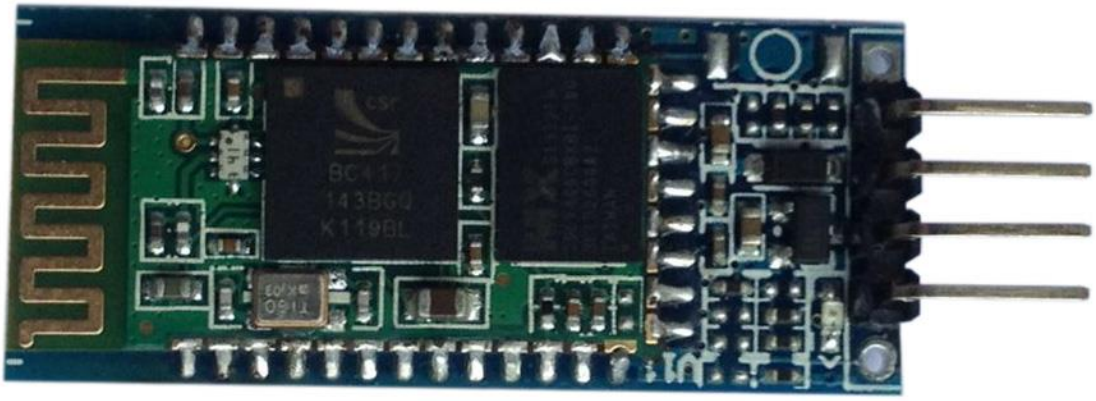
If a master HC-06 module is to be paired with a slave hc-06 module, note the following:

1, PIN is the same.

2, The master module will remember the paired slave module. If you want to pair the second slave module, make sure that the master module clears paired slave module.(give a hign level to PIN26 , master mode will clear paired module)

Any question ,please contact us: info@sihaicorp.com

8、HC-06 with base board 4PIN



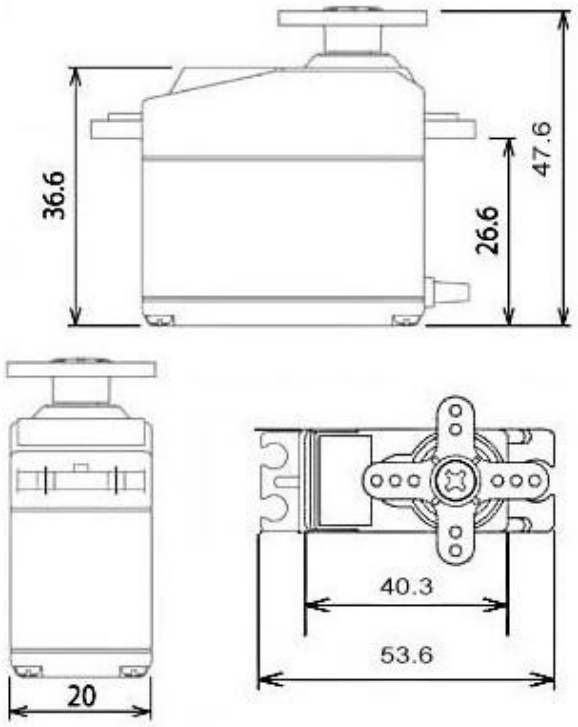
1,Core module uses HC-06, leads from the module interface includes VCC, GND,

TXD, RXD, reserve LED status output pin, the microcontroller can be judged by the foot state Bluetooth has connected KEY pin slave invalid.
2, led indicate Bluetooth connection status, flashing Bluetooth connectivity, lit the Bluetooth connection and open a port Backplane
3, 3.3V LDO input voltage 3.6 ~ 6V, current is about 30mA unpaired, paired about 10mA, the input voltage to prohibit more than 7V!
4, the interface level 3.3V, can be directly connected the various SCM (51, AVR, PIC, ARM, MSP430, etc.), the 5V MCU also can be connected directly, without MAX232 can not go through the MAX232!
5, open to the effective distance of 10 meters, over 10 meters is also possible, but not of this the quality of the connection of the distance do to ensure
6, after the pair when full-duplex serial port to use, do not need to know anything about the Bluetooth protocol, but only supports 8 data bits, 1 stop bit, no parity communication format, which is the most commonly used communication format does not support other formats .
7, support for Bluetooth connection is not established by the AT command set the baud rate, name, passkey, set parameters are saved after. Bluetooth connection is automatically switched to the pass-through mode
8, compact (3.57cm * 1.52cm), the factory chip production to ensure the placement quality. And sets of transparent heat shrink tubing, dust and beautiful, and anti-static.
9, the link from the machine, computer, Bluetooth with Bluetooth function from the functio n with a variety of hosts, the majority with a Bluetooth-enabled cell phone, PDA, PSP and other intelligent terminal pairing from not pairing between the machine.

Any question ,please contact us:

info@sihaicorp.com

MG996R High Torque
Metal Gear Dual Ball Bearing Servo



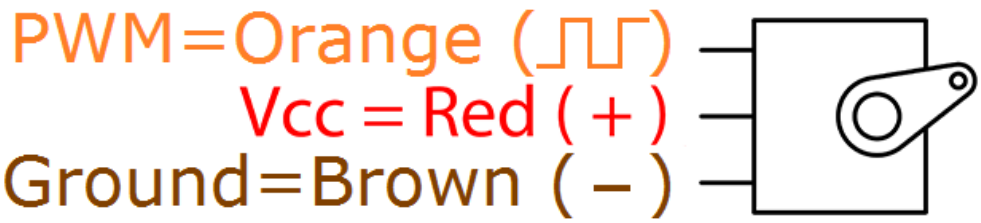
This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

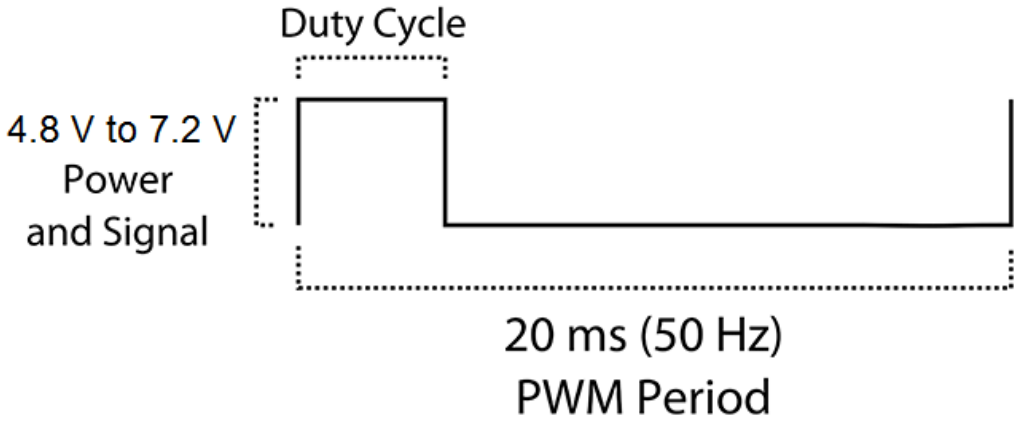
This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

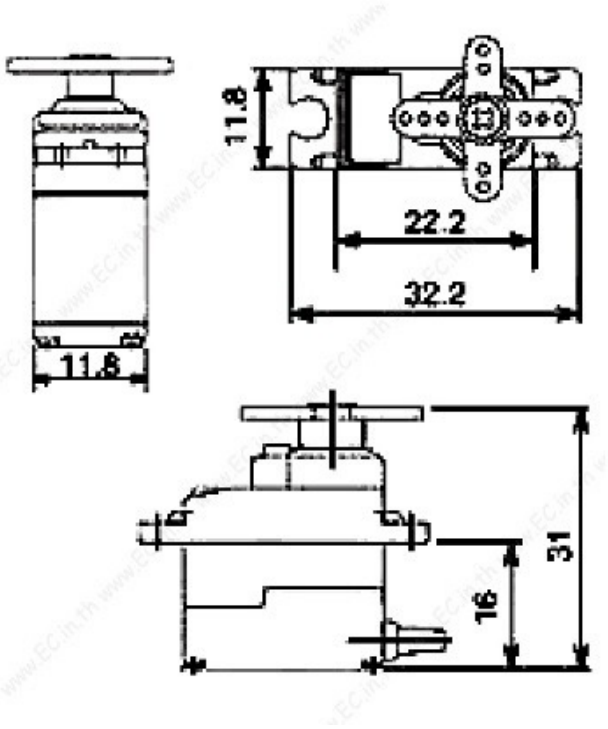
- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC





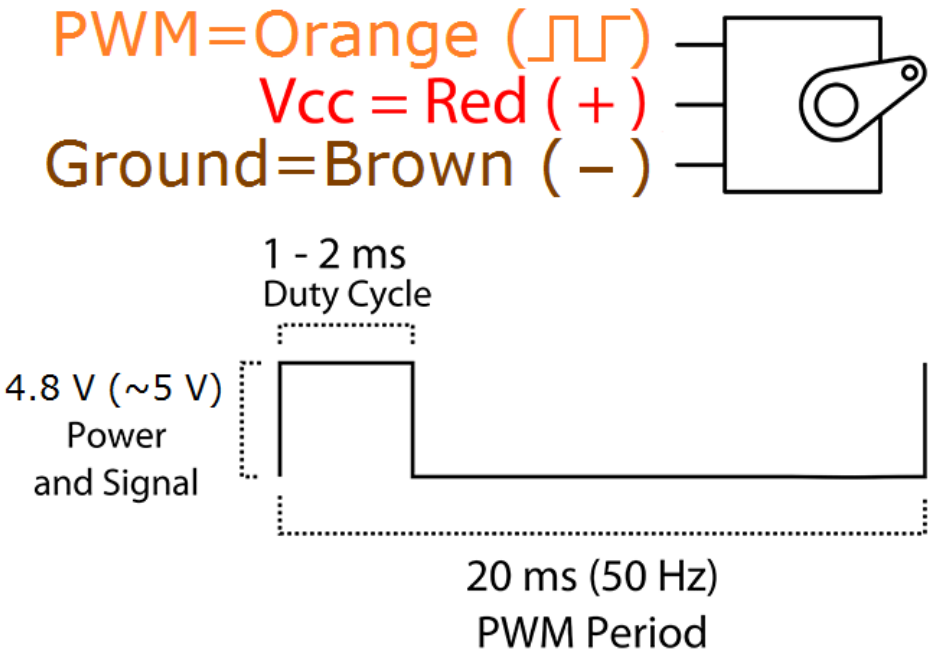
SG90 9 g Micro Servo



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but *smaller*. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

Specifications

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 µs
- Temperature range: 0 ºC – 55 ºC



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.

37D Metal Gearmotors

Pololu 37D Metal Gearmotors are powerful brushed DC motors paired with 37mm-diameter gearboxes. There are nine different gearbox options available, ranging from 6.3:1 to 150:1, and two different motor options: 12 V and 24 V. The 24 V versions offer approximately the same speed and torque at 24 V as their 12 V counterparts do at 12 V, with approximately half the current draw. This datasheet includes two sets of performance graphs for each version, one at its nominal voltage and one at half of its nominal voltage. Each version is available with an integrated 64 CPR quadrature encoder on the motor shaft.

Note: The original versions of these gearmotors had gearboxes with all spur gears. In August 2019, these were replaced by functionally identical “Helical Pinion” versions that feature helical gears for the first stage of the gearbox, which reduces noise and vibration and improves efficiency. The picture on the right shows the helical pinion gear and first mating gear.

Performance summary and table of contents

Rated Voltage	Pololu Item #	Gear Ratio	No Load		At Maximum Efficiency				Max Power	Stall Extrapolation[2]		Graph Pages
			Speed	Current	Speed	Torque	Current	Output		Torque	Current	
		:1	RPM	A	RPM	kg·mm	A	W	W	kg·mm	A	
12 V	4750 [1]	1	10,000	0.2						5	5.5	
	4747, 4757	6.25	1600		1300	4.9	1.2	6.4	12	30		5, 6
	4748, 4758	10	1000		850	6.6	0.91	5.7	12	49		7, 8
	4741,4751	18.75	530		470	10	0.76	5.0	12	85		9, 10
	4742, 4752	30	330		280	18	0.78	5.1	12	140		11, 12
	4743, 4753	50	200		180	22	0.66	4.0	10	210		13, 14
	4744, 4754	70	150		130	32	0.68	4.2	10 [3]	270		15, 16
	4745, 4755	102.08	100		87	42	0.72	3.8	8 [3]	340		17, 18
	4746, 4756	131.25	76		66	60	0.74	4.1	6 [3]	450		19, 20
	2828, 2829	150	67		58	65	0.72	3.8	6 [3]	490		21, 22
24 V	4690 [1]	1	10,000	0.1						5.5	3.0	
	4688, 4698	6.25	1600		1300	5.5	0.58	7.4	14	35		23, 24
	4689, 4699	10	1000		850	7.5	0.49	6.6	14	55		25, 26
	4681,4691	18.75	530		450	13	0.49	6.1	13	95		27, 28
	4682, 4692	30	330		280	19	0.46	5.5	13	150		29, 30
	4683, 4693	50	200		170	27	0.41	4.9	12	230		31, 32
	4684, 4694	70	140		120	39	0.42	5.0	10 [3]	310		33, 34
	4685, 4695	102.08	100		86	51	0.42	4.5	8 [3]	390		35, 36
	4686, 4696	131.25	79		68	63	0.40	4.4	6 [3]	470		37, 38
	4687, 4697	150	68		59	73	0.41	4.4	6 [3]	560		39, 40

Notes:

(1) Max efficiency data and performance graphs currently unavailable for the motors without gearboxes (items #4750 and #4690).

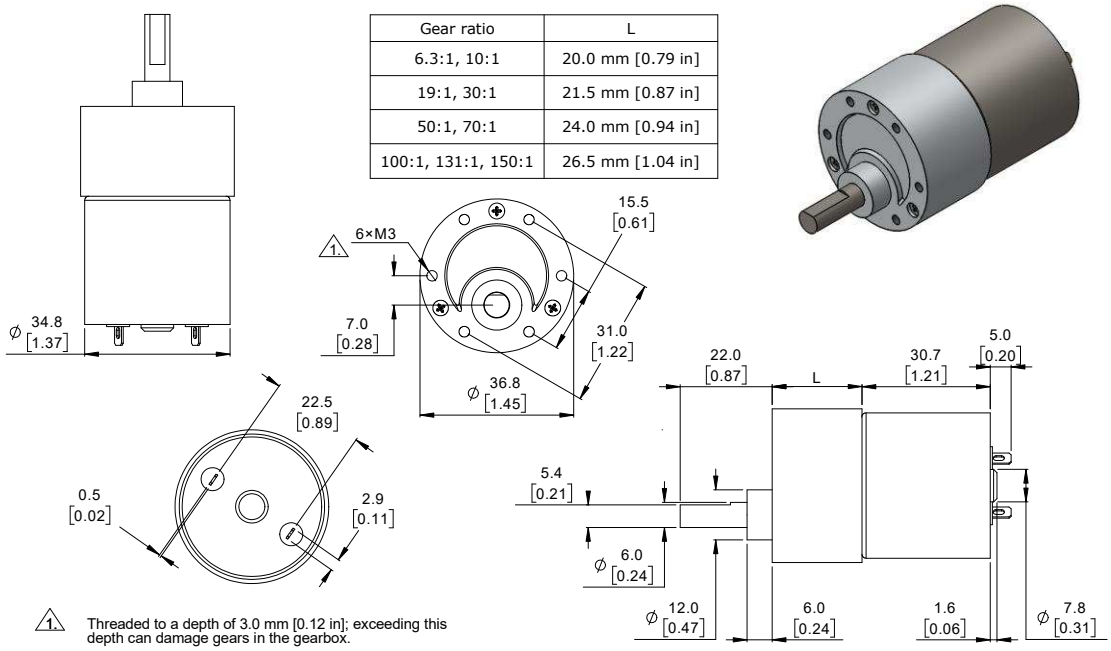
(2) Listed stall torques and currents are theoretical extrapolations; units will typically stall well before these points as the motors heat up. Stalling or overloading gearmotors can greatly decrease their lifetimes and even result in immediate damage. The recommended upper limit for continuously applied loads is 100 kg·mm, and the recommended upper limit for instantaneous torque is 250 kg·mm. Stalls can also result in rapid (potentially on the order of seconds) thermal damage to the motor windings and brushes; a general recommendation for brushed DC motor operation is 25% or less of the stall current.

(3) Output power for these units is constrained by gearbox load limits; spec provided is output power at max recommended load of 100 kg·mm.

Dimensions (units: mm over [inches])

Gearmotor versions without encoders (items #2829, 4681–4689, 4741–4748) weight: 175 g to 195 g

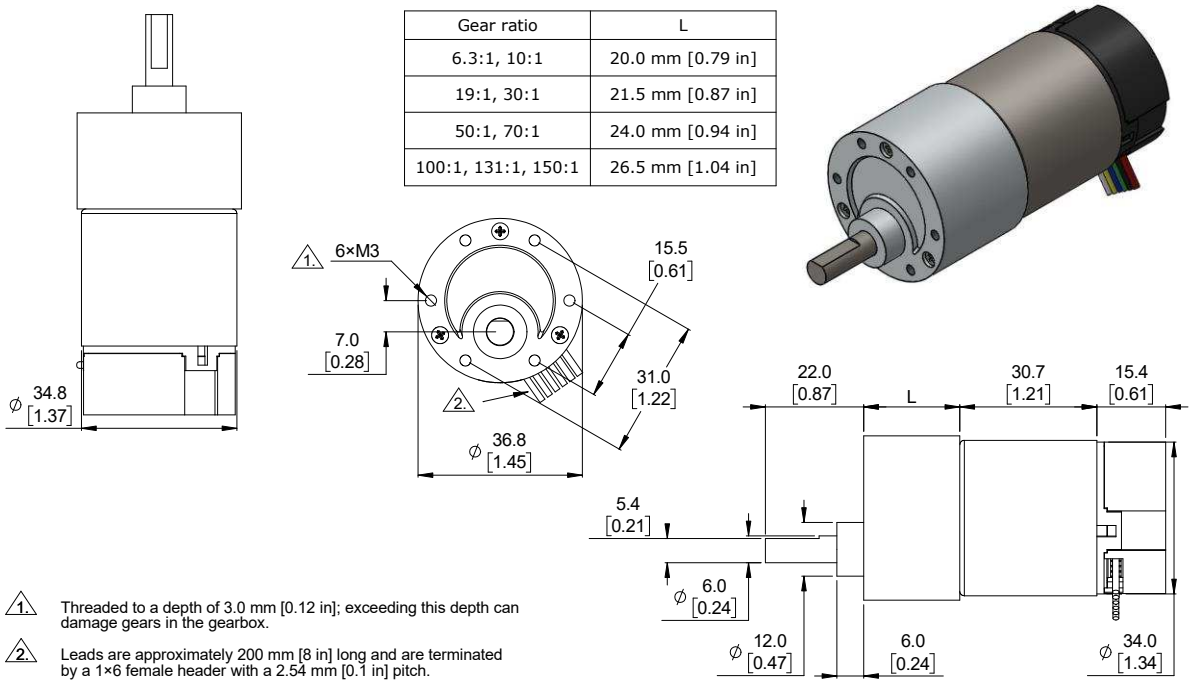
Gear ratio	L
6.3:1, 10:1	20.0 mm [0.79 in]
19:1, 30:1	21.5 mm [0.87 in]
50:1, 70:1	24.0 mm [0.94 in]
100:1, 131:1, 150:1	26.5 mm [1.04 in]



1. Threaded to a depth of 3.0 mm [0.12 in]; exceeding this depth can damage gears in the gearbox.

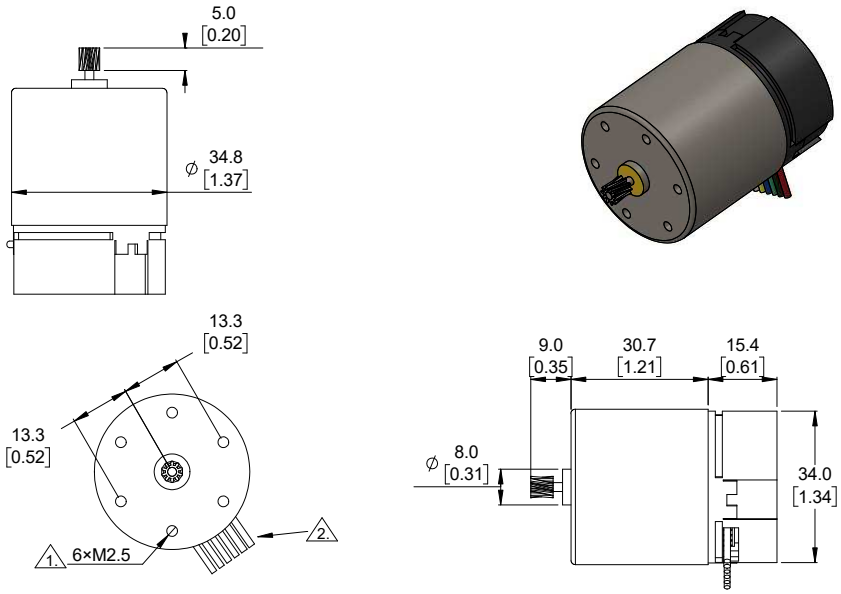
Gearmotor versions with encoders (items #2828, 4691–4699, 4751–4758) weight: 190 g to 210 g

Gear ratio	L
6.3:1, 10:1	20.0 mm [0.79 in]
19:1, 30:1	21.5 mm [0.87 in]
50:1, 70:1	24.0 mm [0.94 in]
100:1, 131:1, 150:1	26.5 mm [1.04 in]



1. Threaded to a depth of 3.0 mm [0.12 in]; exceeding this depth can damage gears in the gearbox.
2. Leads are approximately 200 mm [8 in] long and are terminated by a 1×6 female header with a 2.54 mm [0.1 in] pitch.

Motor with encoder and no gearbox (items #4690, 4750) weight: 110 g



1. Threaded to a depth of 3.5 mm [0.14 in]; exceeding this depth can damage the motor.
2. Leads are approximately 200 mm [8 in] long and are terminated by a 1×6 female header with a 2.54 mm [0.1 in] pitch.

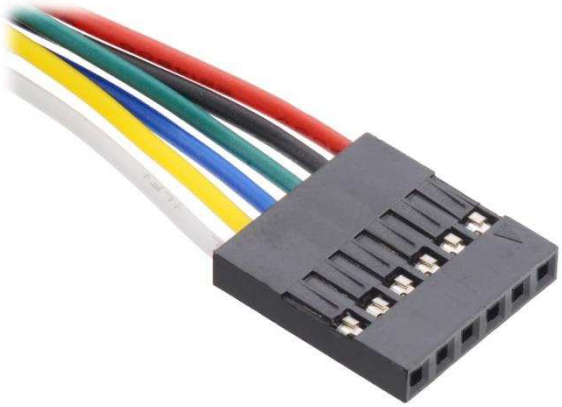
Using the encoder

Versions with encoders have additional electronics mounted on the rear of the motor. Two Hall-effect sensors are used to sense the rotation of a magnetic disc on a rear protrusion of the motor shaft. The encoder electronics and magnetic disc are enclosed by a removable plastic end cap. The following pictures show what the encoder portion looks like with the end cap removed:

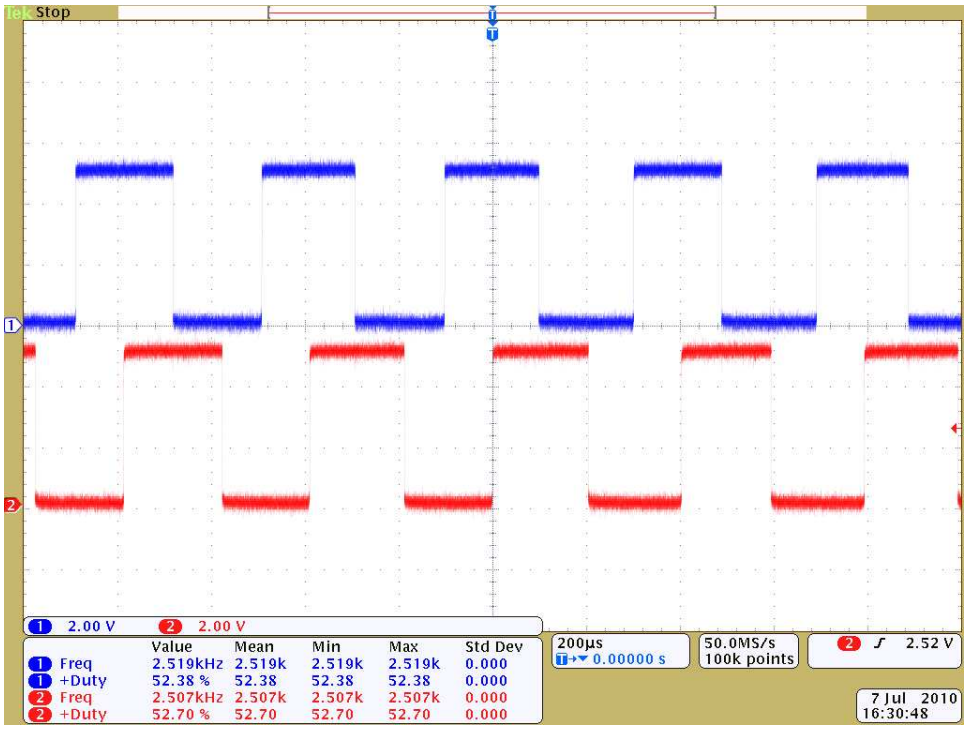
The quadrature encoder provides a resolution of 64 counts per revolution (CPR) of the motor shaft when counting both edges of both channels. To compute the counts per revolution of the gearbox output, multiply the gear ratio by 64.

The motor/encoder has six color-coded, 20 cm (8″) leads terminated by a 1×6 female connector with a 2.54 mm (0.1″) pitch. This connector works with standard 0.1″ male breakaway headers and Pololu male premium jumper and precrimped wires. If this header is not convenient, the crimped wires can be pulled out of the 1×6 housing and used with different crimp connector housings instead (e.g. 1×2 for the motor power and 1×1 housings for the other four leads), or the connectors can be cut off entirely.

Lead Color	Function
Red	Motor power
Black	Motor power
Green	Encoder ground
Blue	Encoder Vcc (3.5 V to 20 V)
Yellow	Encoder A output
White	Encoder B output

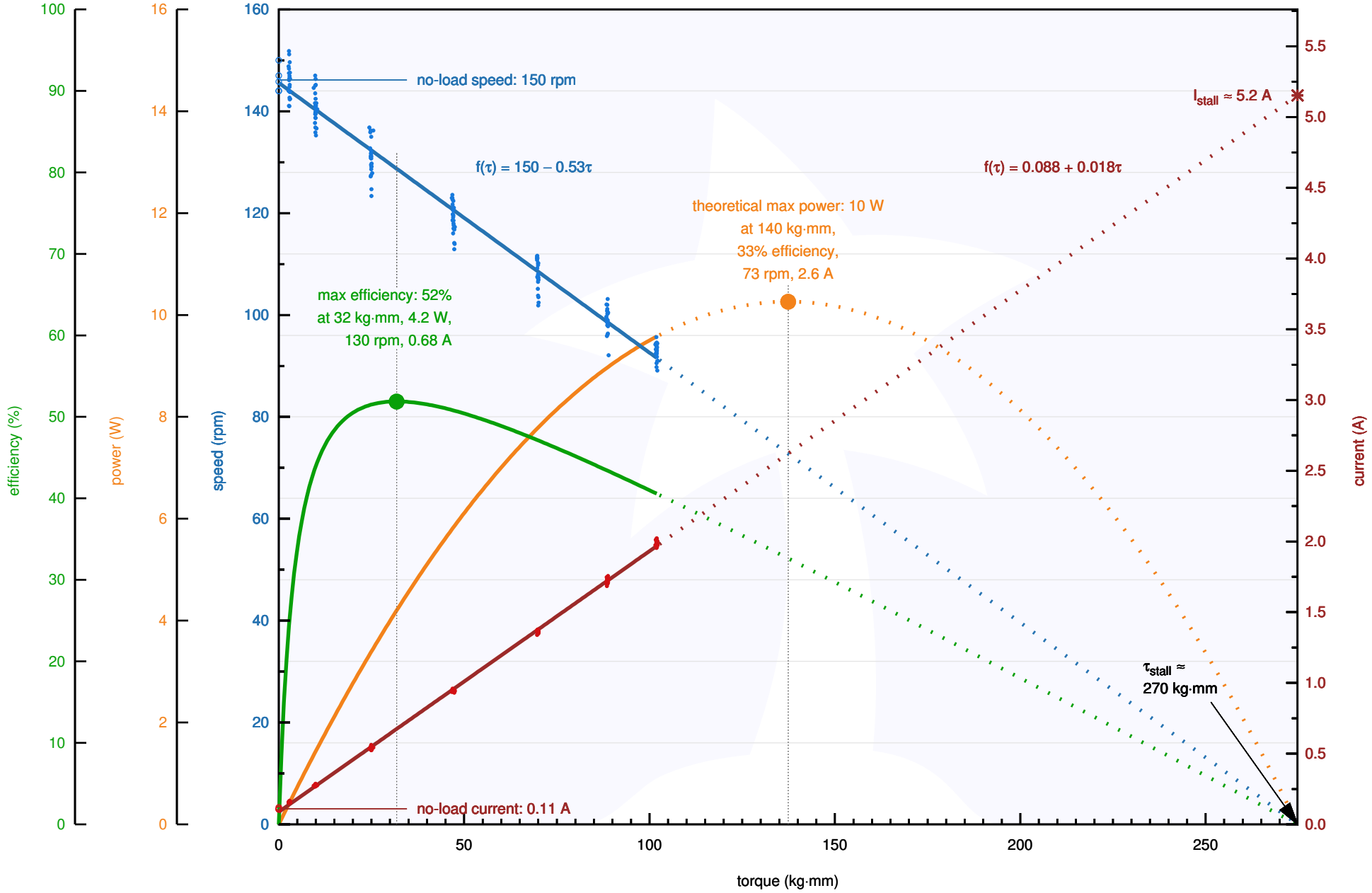


The Hall sensors require an input voltage, Vcc, between 3.5 V and 20 V and draw a maximum of 10 mA. The A and B outputs are square waves from 0 V to Vcc approximately 90° out of phase. The speed of the motor can be determined from the frequency, and the direction of rotation can be determined from the order of the transitions. The following oscilloscope capture shows the A and B (yellow and white) encoder outputs using a 12 V motor at 12 V and a Hall sensor Vcc of 5 V:



Counting both the rising and falling edges of both the A and B outputs results in 64 counts per revolution of the motor shaft. Using just a single edge of one channel results in 16 counts per revolution of the motor shaft, so the frequency of the A output in the above oscilloscope capture is 16 times the motor rotation frequency.

Pololu Items #4744, #4754 (70:1 Metal Gearmotor 37D 12V) Performance at 12 V



Pololu Items #4744, #4754 (70:1 Metal Gearmotor 37D 12V) Performance at 6 V