UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

TRABAJO DE FIN DE GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos.

AUTOR: Vicente Ferrando Santamaría

TUTOR: Leopoldo Armesto Ángel

Curso Académico: 2019/2020

Gracias a mi familia, amigos y pareja
por todo el apoyo incondicional dado
durante tanto tiempo

Resumen:

El objetivo de este TFG es proporcionar a alumnos de cualquier titulación/asignatura relacionada con la robótica los planos, el diseño y el código de un brazo robótico con 3 grados de libertad para que puedan crear su propio robot a un precio económico.

Para la realización del proyecto se deberá de resolver una serie de problemas:

En primer lugar, se deben identificar los elementos con los que se compondrá el robot, tales como motores, fuente de alimentación, controlador de los motores…

Los elementos que se adquieran deberán ser lo más económicos posibles siempre y cuando cumplan con las especificaciones necesarias para mover el robot sin problema.

Una vez se tengan identificados, se procederá al diseño del brazo robótico utilizando un programa de diseño y ensamblaje de piezas. No se buscará un diseño complejo, sino más bien simple para agilizar y simplificar el proceso de fabricación y ensamblaje de piezas por parte de los alumnos que monten el robot.

Al finalizar el diseño, se importarán las piezas a un programa de simulación para comprobar el correcto funcionamiento dinámico del robot y se procederá a construir el robot.

En cuanto al funcionamiento del robot, se programará para que funcione en cinemática inversa y tenga dos modos de actuación, el manual y el automático. En el manual moveremos el robot sobre los ejes x, y, z mediante sliders y en el modo automático crearemos una trayectoria que seguirá el robot. Una vez que los modos de funcionamiento se ejecuten correctamente en el simulador se procederá a transmitir las instrucciones por puerto serie a los motores.

Palabras clave: ‘brazo robótico’, ‘3 grados de libertad’, ‘control’, ‘cinemática inversa’.

Resum:

L'objectiu d'aquest TFG és proporcionar als alumnes de qualsevol titulació / assignatura relacionada amb la robòtica els plànols, el disseny i el codi d'un braç robòtic amb 3 graus de llibertat perquè puguin crear el seu propi robot a un preu econòmic.

Per a la realització del projecte s'haurà de resoldre una sèrie de problemes:

En primer lloc, s'han d'identificar els elements amb els quals es compondrà el robot, com ara motors, font d'alimentació, controlador dels motors ...

Els elements que s'adquireixin hauran de ser el més econòmics possibles sempre que compleixin amb les especificacions necessàries per moure el robot sense problema.

Una vegada s'hagin identificats, es procedirà al disseny del braç robòtic utilitzant un programa de disseny i assemblatge de peces. No es buscarà un disseny complex, sinó més aviat simple per agilitar i simplificar el procés de fabricació i acoblament de peces per part dels alumnes que muntin el robot.

A l'acabar el disseny, s'importaran les peces a un programa de simulació per comprovar el correcte funcionament dinàmic del robot i es procedirà a construir el robot.

Pel que fa al funcionament del robot, es programarà perquè funcioni en cinemàtica inversa i tingui dues maneres d'actuació, el manual i l'automàtic. Al manual mourem el robot sobre els eixos x, i, z mitjançant 'sliders' i en la manera automàtica crearem una trajectòria que seguirà el robot. Una vegada que els modes de funcionament s'executin correctament en el simulador es procedirà a transmetre les instruccions per port sèrie als motors.

Paraules clau: 'braç robòtic', '3 graus de llibertat', 'control', 'cinemàtica inversa'.

Abstract:

The objective of this Final Project is to give the students of any titulation or subject related to robotics the plans, design and the code of a robotic arm with three grades of freedom so they can create their own robot in an inexpensive way.

To conduct this project is necessary to solve a series of problems:

First of all, the elements that will form the robot such as the driving forces, power supply and motor controller need to be identified.

All the elements acquired have to be as economical as possible as long as they achieve the specifications needed to move the robot without problem.

Once they are identified, the design of the robotic arm using a design and assembly program will proceed. The objective is not to get a complex design but to accelerate and simplify the production and assembly process by the students making the robot.

When the design is ended, all the pieces will be introduced to a simulation program to check the correct dynamic operation of the robot and the construction of the robot will start.

For the operation of the robot, it will be programmed to work in reverse kinematics and have two modes of action, manual and automatic. In the manual we will move the robot on the x, y and z axes using sliders and with the automatic mode we will create a path to be followed by the robot. Once the operating modes are correctly executed in the simulator, the instructions will be transmitted via serial port to the motors.

Key words: 'robotic arm', '3 degrees of freedom', 'control' 'inverse kinematics'

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos.

I. MEMORIA

AUTOR: Vicente Ferrando Santamaría

TUTOR: Leopoldo Armesto Ángel

Curso Académico: 2019/2020

Índice memoria:

Índice figuras de la memoria:

1. Objeto

El trabajo se basará en la creación de un brazo robótico con tres grados de libertad con fines docentes, para ello se procederá a la realización de varios procesos, tales como el diseño y ensamblaje del brazo, la programación, simulación y la puesta en marcha del brazo robótico.

Debido a que el proyecto tiene una finalidad educativa, el diseño del robot será lo más sencillo posible para reducir los tiempos montaje y ensamblaje de piezas. Además, no se pretende que la dificultad del proyecto se base en esta parte, sino en la implementación y simulación.

Los campos docentes que abarca el proyecto son todos los relacionados con la robótica y el control robótico, aunque el grado de dificultad de los conceptos que se puedan explicar y desarrollar dependerán de la asignatura en la que se utilice y del propio profesor. Al tratarse de un diseño simple es fácilmente ampliable y adaptable a nuevo temario. Como base se ha planteado el proyecto para el aprendizaje de la cinemática inversa, estudio de especificaciones de componentes electrónicos, adaptación de los componentes electrónicos entre sí, programación y simulación del robot, y comunicación por puerto serie. Además, se ha planteado como un proyecto alternativo a la mayoría de los proyectos educativos en los cuales, el controlador es de la marca Arduino. En este caso, se ha desarrollado mediante un controlador de servomotores y un programa de simulación robótico.

Al tratarse de un proyecto académico y para un amplio número de alumnos se ha diseñado teniendo en cuenta el coste de fabricación, siendo el mínimo posible manteniendo las especificaciones mínimas para el movimiento del brazo y con la posibilidad de añadir grados de libertad o de añadir nuevas funcionalidades al brazo robótico, como, por ejemplo, una pinza.

2. Antecedentes

2.1. Definición de brazo robótico

Antes de empezar, se debe definir el concepto de brazo robótico para tener la idea clara de lo que se va a realizar. Se trata de un brazo mecánico cuyas partes están interconectadas mediante articulaciones capaces de realizar movimientos de rotación, de traslación, o lineales. Actualmente, la mayoría de los brazos son programables y muy semejantes a un brazo humano.

Suelen formar parte de una herramienta mayor y tienen un gran número de funcionalidades, desde usos cotidianos hasta prótesis para personas con desmembramiento pasando por el uso industrial para la automatización de procesos.

2.2. Historia de los brazos robóticos

Existen brazos robóticos desde hace siglos, en la antigua Grecia ya se pudieron ver máquinas cuyo aspecto y funcionamiento se asemeja mucho a los brazos robóticos actuales. En la Edad Media se continuó avanzando en este campo junto con Alberto Magno, creador de un hombre de hierro que le servía como mayordomo realizando algunas tareas tales como andar, abrir las puertas y saludar a los invitados. No fue hasta 1948 cuando George Devol incorporó los robots a la industria abriendo un camino de nuevas posibilidades. En 1960 instaló el primer brazo robótico en una fábrica de Nueva Jersey, cuyo trabajo era levantar y apilar piezas de metal caliente.

En la actualidad, la robótica y la tecnología están creciendo de forma exponencial mejorando las condiciones de vida de las personas y facilitando la misma. Surgen nuevos avances muy deprisa, y aunque las nuevas generaciones no tienen problema en adaptarse ya que han crecido con estos avances, la sociedad en general debe cambiar sus hábitos, ya que, aunque son para mejorar la calidad de vida de las personas, también deja sin trabajo a un gran número de ellas.

En estos últimos años, se ha empezado a ver de forma más generalizada y simple la inteligencia artificial y el *machine learning* en un amplio número de ámbitos. Uno de esos ámbitos ha sido el de los brazos robóticos. La combinación y perfeccionamiento de estas dos materias puede generar otro gran avance en nuestra sociedad y quien sabe lo que nos espera en un futuro no tan lejano.

2.3. Usos de los brazos robóticos

Los brazos robóticos tienen una gran variedad de usos, éstos se pueden dividir en tres grandes categorías: el uso cotidiano, el uso industrial, y el uso en el ámbito de la medicina.

En el uso cotidiano podemos encontrar todo tipo de brazos robóticos, tales como atrapa objetos, que mediante una cámara calculan la trayectoria de un objeto que se dirige hacia el brazo y lo atrapa en el aire, o brazos cuyos movimientos calculan la fuerza y la dirección exacta para jugar a los bolos o al billar de forma perfecta o incluso hay que son capaces de realizar recetas de cocina identificando y cogiendo ellos mismos los alimentos para cocinarlos.

En el uso industrial, se combina la rapidez, la efectividad y la ausencia de errores en los procesos. Cuanto mayor sea la rapidez más productos se podrán fabricar y mayor será el beneficio. Se pueden aplicar a casi cualquier proceso de fabricación, desde la industria automovilística hasta la fabricación de juguetes pasando por la industria alimentaria.

En el ámbito de la medicina y la biotecnología, estos brazos robóticos se pueden utilizar tanto como prótesis de desmembramientos como para la realización de cirugías y operaciones cuya precisión sea milimétrica e incluso micrométrica, como por ejemplo las operaciones oculares.

Las posibilidades en cualquiera de los ámbitos son infinitas.

2.4. Motivación del proyecto

El proyecto nace de la necesidad del tutor de mejorar la enseñanza en su propia asignatura de robótica. La mayoría del temario de la asignatura se podría apoyar en el brazo robótico diseñado. De esta forma, se impartirá la asignatura de forma mucho más dinámica tanto para el profesor como para los alumnos. Además, los conceptos teóricos se asimilarán mejor y se podrán comprobar los cálculos realizados no sólo con los programas informáticos, sino de forma visual, observando los movimientos del robot.

Como ya se ha comentado, la materia que se puede aprender utilizando este robot como base, dependerá de cada profesor y/o asignatura, pero simplemente con la construcción y configuración del brazo se podrán presentar y entender las técnicas de cinemática directa, cinemática inversa, una pequeña introducción al programa de simulación y a la programación en el mismo. El alumno será capaz de simular e implementar cualquier robot siempre que no tenga una configuración demasiado compleja y también podrá aprender los conceptos básicos de la comunicación por puerto serie y como configurarla.

Debido al diseño simple del robot se podrá fabricar fácilmente sin tener que dedicarle demasiado tiempo y esfuerzo. Otra de las ventajas del diseño será el fácil mantenimiento ante cualquier rotura, separación o imperfección que pueda generarse. Se podrá identificar rápidamente el problema y se podrá sustituir o arreglar la pieza en cuestión sin tener que modificar otras partes del robot.

Se ha pretendido no limitar el robot, es decir, está preparado para ser fácilmente ampliable si el profesor o la asignatura lo requiere. Los motores son lo suficientemente potentes como para incrementar los grados de libertad añadiendo nuevos servomotores, además, el controlador de motores admite hasta 16 servomotores.

2.5. Marco teórico

En este apartado se verán las hipótesis y cálculos necesarios para determinar de forma teórica la posición y los movimientos del robot. Con estas hipótesis y cálculos se podrá comprobar el correcto funcionamiento del robot. Se determinará la cinemática directa y la cinemática inversa. Las hipótesis se han generado teniendo en cuenta un brazo robótico con 3 grados de libertad, siendo los 3 de revolución.

La cinemática se define como la parte de la mecánica que trata del movimiento en un espacio tiempo determinado que no tiene en cuenta las causas de dicho movimiento.

2.5.1. Cinemática directa

La cinemática directa es una técnica usada para calcular la posición de las partes de una estructura articulada a partir de sus componentes fijas. Utiliza ecuaciones cinemáticas para calcular la posición de su actuador final a partir de valores específicos.

Hay distintas formas de calcular la cinemática directa de un brazo robótico, estos métodos son:

- Transformación de matrices
- Geometría
- Transformación de coordenadas

En esta ocasión se utilizará la transformación de matrices ya que se trata de una técnica sistemática y no está limitada por los grados de libertad, ya que el método de la geometría está limitado a un pequeño número de grados de libertad. El método de transformación de matrices homogéneas o matrices Denavit-Hartemberg (D-H) parte de la hipótesis de considerar al

robot como una cadena cinemática formada por eslabones unidos entre sí. A cada eslabón de la cadena se le asocia un sistema de referencia solidario. El sistema de referencia de cada eslabón se coloca al final de este, en el extremo de la articulación a la cual está conectado el eslabón siguiente. En la base del robot se coloca un sistema de referencia fijo que servirá como origen de coordenadas. La representación de la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot se harán mediante la matriz de transformación ${}^{i-1}A_i$. Para la representación total o parcial de la cadena de cinemática del robot se realizará encadenando las transformaciones:

$$T = {}^{0}A_3 = {}^{0}A_1 \cdot {}^{1}A_2 \cdot {}^{2}A_3$$

En nuestro caso tan solo tenemos 3 grados de libertad, por tanto, solo se encadenarán 3 matrices de transformación.

Cada sistema de coordenadas sc_i se define a partir del sistema de coordenadas anterior sc_{i-1} y de los ejes de articulación i e i+1. Este sistema permite el paso de un eslabón al siguiente mediante 4 parámetros que dependen exclusivamente de las características del robot. Estos cuatro parámetros son:

- Rotación θ_i : Ángulo de X_{i-1} a X_i medido sobre el eje Z_{i-1} (Regla de la mano derecha)
- Traslación d_i : Distancia de X_{i-1} a X_i medida a lo largo del eje Z_{i-1}
- Traslación a_i : Distancia de Z_{i-1} a Z_i medida a lo largo del eje X_i
- Rotación α_i : Ángulo de Z_{i-1} a Z_i medido sobre el eje X_i (Regla de la mano derecha)

La ecuación de transformación final de un eslabón al siguiente quedaría tal que así:

$${}^{i-1}A_i = T(z,\theta_i)T(0,0,d_i)T(a_i,0,0)T(x,\alpha_i)$$

Para convertir la ecuación a la matriz transformada, quedaría de la siguiente manera, siendo C el coseno, y S el seno:

$${}^{i-1}\mathbf{A}_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 1: Matriz de transformación Denavit-Hartemberg (D-H). Fuente [3]

Para obtener los parámetros del robot se han de seguir estos 9 pasos de forma sistemática:

1. Identificar los eslabones y los ejes de las articulaciones trazando líneas imaginarias a lo largo de ellos. Los eslabones se numeran empezando por el uno, aunque la base fija se numera como eslabón 0. Las articulaciones se numeran desde 1 hasta n.
2. El origen O_i se sitúa entre los ejes i+1 e i(Z_{i-1} y Z_i). Si no existe intersección, se identificará la perpendicular común entre ejes consecutivos y el origen O_i se colocará en la intersección del eje i+1 con la normal común entre los ejes i e i+1.
3. Se coloca el eje Z_i sobre el eje de la articulación i+1
4. Se coloca el eje X_i sobre la perpendicular común. Si lo ejes intersecan, se colocará sobre la normal al plano que forman los ejes Z_{i-1} y Z_i (Producto vectorial de Z_{i-1} x Z_i)
5. Se coloca el eje Y_i siguiendo la regla de la mano derecha
6. Para i de 0 a n-1 se situará el eje z sobre el eje de la articulación i+1. El eje X_i se situará con la normal común entre Z_i y Z_{i-1}
7. El primer sistema de coordenadas asociado a la base del robot, SC_0 se puede situar en cualquier punto a lo largo del eje Z_0.
8. En el resto de los sistemas de coordenadas debe cumplirse que el origen esté en la intersección entre Z_i y x_i. El eje Y_i se define para que SC_i sea un sistema dextrógiro.
9. El sistema de coordenadas SC_n asociado al último elemento podrá tener su origen en cualquier parte de éste, siempre que la dirección de Z_n sea equivalente a la de Z_{n-1}.

2.5.2. Cinemática inversa

La cinemática inversa es la técnica que permite determinar el movimiento de una cadena de articulaciones para lograr que el actuador final se ubique en una posición concreta. Normalmente la solución de esta técnica no es única. Dependiendo de la configuración del robot habrá múltiples soluciones.

Hay distintos métodos de resolver el problema de la cinemática inversa, algunos ellos son:

- Métodos geométricos
- A partir de matrices de transformación homogénea
- Desacoplamiento cinemático
- Cuaterniones duales

En este caso se resolverá por el método geométrico ya que es adecuado para pocos grados de libertad. Este procedimiento se basa en encontrar suficientes relaciones geométricas dependientes de las coordenadas del robot.

En primer lugar, planteamos una posición aleatoria de un robot con 3 grados de libertad:

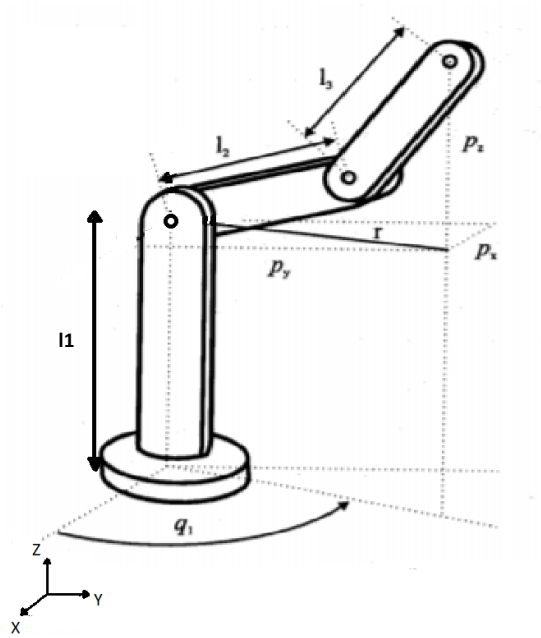


Fig. 2: Brazo robótico con 3 grados de libertad. Fuente [1].

.

Para la resolución del primer ángulo solo se tendrán en cuenta los ejes X e Y.

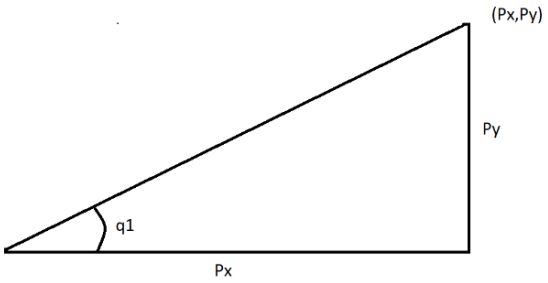


Fig. 3: Representación geométrica simple de la primera articulación del brazo robótico visto desde arriba. Fuente propia.

Para la resolución de la posición de la primera articulación se aplicará la arco tangente:

$$q1 = \tan^{-1}\left(\frac{P_y}{P_x}\right)$$

Para el cálculo de la posición de la tercera articulación, se aplicará el teorema del coseno:

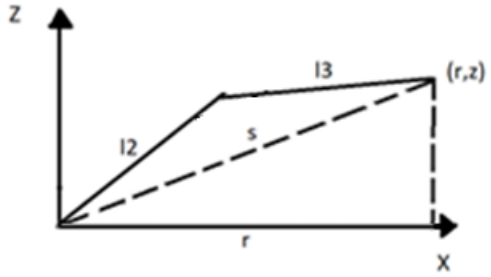


Fig. 4: Representación geométrica simplificada del robot. Fuente propia.

$$s^2 = l_2^2 \cdot l_3^2 - 2 \cdot l_2 \cdot l_3 \cdot \cos(q3) \rightarrow$$

$$\left(\sqrt{r^2 + p_z^2}\right)^2 = l_2^2 \cdot l_3^2 - 2 \cdot l_2 \cdot l_3 \cdot \cos(q3) \rightarrow$$

$$p_x^2 + p_y^2 + p_z^2 = l_2^2 \cdot l_3^2 - 2 \cdot l_2 \cdot l_3 \cdot \cos(q3)$$

$$q3 = \cos^{-1}\left(\frac{p_x^2 + p_y^2 + p_z^2 - l_2^2 - l_3^2}{2 \cdot l_2 \cdot l_3}\right)$$

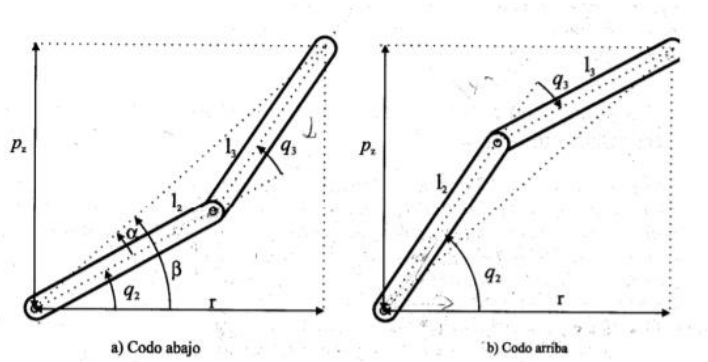


Fig. 5: Representación geométrica de las dos posibles soluciones de los codos. Fuente [4].

Para obtener la posición de la segunda articulación (q2), haremos lo siguiente:

$$q2 = \beta - \alpha$$

El ángulo β se calculará aplicando la función tangente sobre el triángulo formado por la línea imaginaria 'r' y la coordenada z del extremo del robot.

$$\beta = \tan^{-1}\left(\frac{p_z}{r}\right) = \tan^{-1}\left(\frac{p_z}{\pm\sqrt{p_x^2 + p_y^2}}\right)$$

El ángulo α se calculará proyectando el tercer eslabón con respecto al ángulo q3 simplificando así el triángulo y aplicando la función de la tangente sobre un triángulo rectángulo.

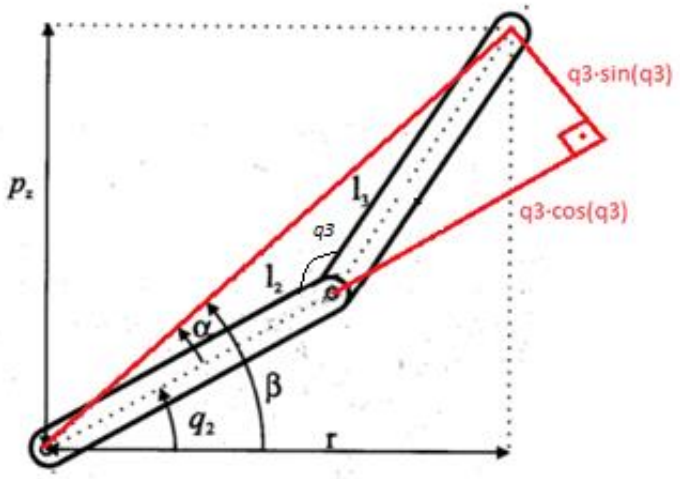


Fig. 6: Proyección del tercer eslabón con respecto a la tercera articulación. Fuente [4].

$$\tan(\alpha) = \frac{l_3 \cdot \cos(q_3)}{l_2 + l_3 \cdot \sin(q_3)} \rightarrow \alpha = \tan^{-1}\left(\frac{l_3 \cdot \cos(q_3)}{l_2 + l_3 \cdot \sin(q_3)}\right)$$

$$q_2 = \tan^{-1}\left(\frac{p_z}{\pm\sqrt{p_x^2 + p_y^2}}\right) - \tan^{-1}\left(\frac{l_3 \cdot \cos(q_3)}{l_2 + l_3 \cdot \sin(q_3)}\right)$$

Con la doble solución de la raíz cuadrada obtenemos las dos posibles soluciones de la segunda articulación.

3. Estudio de necesidades

3.1. Especificaciones del encargo

El proyecto tenía que cumplir una serie de requisitos para que fuera de utilidad en asignaturas de robótica y programación. Algunos de estos requisitos fueron:

- Tratarse de un robot funcional, que pudiera moverse sin problemas.
- El diseño fuera lo más simple posible para facilitar su construcción, evitar problemas de montaje y facilitar el mantenimiento.
- Evitar que la configuración fuera demasiado compleja.
- Diseñar el robot con los componentes electrónicos de la forma más económica posible.
- Capacidad de poder explicar conceptos básicos de robótica apoyándose en el robot.
- Poder comprobar cálculos teóricos mediante la utilización del brazo.

3.2. Estudios propios

Los estudios propios realizados para este proyecto han sido los referidos a los cálculos de la cinemática directa e inversa mediante la transformación de matrices homogéneas y métodos geométricos respectivamente. También, según las hojas de especificaciones de cada componente, se tuvo en cuenta el voltaje e intensidad necesario por la fuente de alimentación.

3.3. Limitaciones y condicionantes

Los principales limitantes fueron el ajustar al máximo posible el presupuesto de los componentes que forman el robot y hacer que fuera totalmente funcional y con la posibilidad de aumentar el número de grados de libertad. Estas limitaciones y necesidades fueron fundamentales para la elección de los componentes del robot, tanto electrónicos (como los motores), como físicos (material de las piezas). Una necesidad fundamental que se tuvo que cubrir fue que el robot pudiera adaptarse a los temarios básicos de robótica con los mínimos cambios en su diseño y/o configuración.

4. Alternativas y solución adoptada

A la hora de elegir un componente frente a otro o una solución a un problema dado se debe tener una visión holística del proyecto y no centrarse simplemente en un problema de forma cerrada, sino analizar las diferentes interacciones de los elementos. En este proyecto en concreto, es importante analizar los componentes de esta forma ya que, por ejemplo, los motores deben ser lo suficientemente potentes como para mover el brazo, pero a la vez no deben pesar demasiado porque forman parte del brazo y un peso excesivo en los motores podría imposibilitar el movimiento fluido del brazo.

4.1. Motores

Para la elección de los motores, se tuvieron que analizar los motores brushless, motor paso a paso y servo motor. Los motores brushless se descartaron ya que no proporcionan un control de la posición preciso. La elección del motor quedó entre el servo motor y el motor paso a paso. A pesar de que el motor paso a paso tiene un coste menor y tiene menor complejidad, se optó por el servo motor ya que tiene mayor precisión y velocidad, realiza unos movimientos más suaves y tiene un mejor rendimiento. Hubo 3 servo motores posibles para el proyecto:

- Mini SG90 servo micromotor
- MG996R Micro Digital Servo Motor con engranaje metálico
- 25kg Motor de dirección servo digital Coreless Metal Gear para RC modelo

Los tres motores están ordenados de más barato a más caro. El primer servo motor, es el que menos par ofrece, además los engranajes son de plástico. A pesar de ser el más barato y ligero no ofrece la fuerza necesaria como para mover el brazo. Este tipo de motores podrían servir en la parte final de un brazo robótico con más grados de libertad o como accionadores de pinzas. El segundo, se presenta como un motor con una fuerza razonablemente buena, con unos engranajes metálicos, lo que ofrecen mayor vida útil y mayor resistencia, a un precio asequible. El último de los motores se presenta como un motor con mucha fuerza y de buena calidad, lo que repercute en la subida del precio.

Al analizar los tres motores, se llegó a la conclusión de que el motor MG996R (segunda opción) era la mejor opción ya que ofrecía un par suficientemente elevado como para mover el brazo sin problema e incluso

tenía la suficiente fuerza como para añadir más grados de libertad al brazo. El tercer motor también cumplía con estos requisitos, pero además de tener un peso muy elevado, lo que dificultaría el movimiento fluido, tiene un precio demasiado elevado para este proyecto.

4.2. Controlador de motores

Las opciones planteadas como controladores fueron:

- Bluetooth 16 canales PWM Servo Motor Controlador de placa de circuito PCB engranaje de dirección para SG90 MG995 Arduino Robot servo escudo Raspberry Pi DIY ofrece software para PC/Android libremente
- WayinTop PCA9685 PWM Servo Motor Driver 16 Canales 12 bit Interfaz I2C IIC Controlador para Arduino y Raspberry Pi
- Arduino Nano + Controlador de servo motor

Se ha pretendido plantear una opción diferente a la que se ve en la mayoría de los proyectos de robótica en la que se utiliza un microcontrolador encargado de relacionar las entradas y las salidas como puede ser Arduino, además, aunque se utilice este microcontrolador es necesario adquirir el controlador de servo motores, así que optaremos por adquirir simplemente el controlador de servos y programarlo desde un programa de simulación e implementación. Los dos controladores de motores que se plantean son muy similares y aunque el WayinTop es más barato y sería funcional para nuestro proyecto, se ha optado por la primera opción ya que, aunque sea un poco más caro, ofrece la posibilidad de controlar los motores por bluetooth mediante el teléfono móvil, lo cual es una opción bastante interesante para futuras ampliaciones.

4.3. Fuente de alimentación

La elección de la fuente de alimentación dependió de los motores y el controlador ya que estos necesitan un voltaje e intensidad específicos para funcionar. Las opciones para alimentar el robot fueron:

- Tangsfire Batería Recargable 6V 700mAh AA Ni-CD Packs SM 2P Plug para Juguetes Power Bank con Cargador
- Energizer Max - Pilas alcalinas AA/Mignon LR6 + GTIWUNG 4 x 1,5V AA Caja de Soporte de Batería Caja de Almacenamiento de Batería de Plástico (4 Solts × 6 piezas)

No se necesita una gran cantidad de energía para alimentar el controlador y los motores, por esa razón, no existe una gran diferencia entre el uso de pilas desechables o batería recargable así que, para abaratar los costes de fabricación, se utilizarán las pilas desechables.

4.4. Diseño del robot

El diseño del brazo robótico ha estado evolucionando constantemente, desde la base hasta los eslabones. Se ha pretendido un diseño simple que se pudiera crear fácilmente mediante una impresora 3D o mediante una cortadora laser sin tener que emplear demasiado tiempo en ello. En primer lugar, empezaremos a hablar de la base.

En un primer momento, se puso como base una caja de madera rectangular en la que se ocultaba el primer motor, la fuente de alimentación, y el controlador. Se le hicieron agujeros en la tapa para pasar el cableado del resto de motores. Aunque esta alternativa era funcional y estética, era un poco complicado hacerlo con impresión 3D o corte láser y en lugar de eso, se utilizó una tabla circular relativamente pesada, la cual se puede obtener fácilmente en cualquier ferretería o incluso se podría fabricar.

Para colocar el primer motor, que gira sobre el eje Z, se planteó en un primer momento colocarlo directamente sobre la base, pero no tenía la suficiente estabilidad, además, al colocar el eslabón que debía mover, éste balanceaba demasiado lo cual podría llegar a romper o deteriorar el propio motor. En lugar de eso, se fijó una segunda base mediante patas ancladas a la base principal. En esta segunda base se atornilló el motor dejándolo totalmente estático. Para evitar el balanceo de este primer eslabón, debido a la amplitud de la base, se añadieron tornillos entre la segunda base y el propio eslabón para que pueda apoyarse sobre estos.

En cuanto al diseño de los eslabones, aunque no se llegaron a montar, se diseñaron en un primer momento con doble lámina, una a cada lado del motor, para aportar mayor robustez y estética al brazo, pero aumentaba el peso del propio brazo y se hubieran necesitado motores con mayor par de fuerza lo que habría hecho aumentar el coste. Al ver que no esto no añadía ninguna funcionalidad extra al brazo y que las simples respondían bien al movimiento del robot se optó por estas últimas.

El material del cual están compuestos los eslabones puede ser muy variado ya que se puede utilizar madera, aluminio, plástico… Un requisito fundamental en la elección del material es que sea manejable (a la hora de cortarlo y moldearlo), y que sea ligero pero resistente, es decir, debe ser ligero para reducir el peso que tengan que mover los motores, pero a la vez debe ser resistente ya que no deberá doblegarse ni romperse al incorporar los motores al final de cada eslabón. En este caso, se ha utilizado el aluminio para construir el robot, ya que debido a su densidad y resistencia es el más idóneo para el proyecto. Tiene la suficiente rigidez como para no doblegarse

al incorporar los motores y tiene un peso muy reducido. En un primer momento se utilizaron láminas de madera por su ligereza, pero no tenían la rigidez necesaria.

La base secundaria (explicada en el apartado anterior) y el primer eslabón, son de aluminio con relleno de poliestireno lo que proporciona más cuerpo y rigidez. El resto de los eslabones están compuestos por láminas simples de aluminio.

4.5. Modos de funcionamiento

Los modos de funcionamiento que se plantearon en un principio fueron el control por motores, el control por ejes o por posición, y la ejecución de una trayectoria predefinida. No se han podido combinar estos tres modos de funcionamiento debido a la incompatibilidad en la configuración de las articulaciones. Se pretendía que mediante el panel de interfaz de usuario del programa de simulación se pudiera controlar el movimiento tanto por motores como por ejes, pero debido a esta incompatibilidad se optó por preservar el control por ejes y la creación de una trayectoria que siguiera el extremo del robot.

5. Descripción y justificación detallada de la solución

5.1. Motores

Tal y como hemos comentado en la sección anterior, se eligieron los servomotores MG996R. Un servomotor es un dispositivo de accionamiento para el control de precisión de velocidad, par motor y posición. En el interior del servomotor, se encuentra un *encoder,* el cual convierte el movimiento mecánico en pulsos digitales interpretados por un controlador de movimiento. Se deben acompañar de un *driver,* y en conjunto forman un circuito capaz de controlar la posición, el par y la velocidad. El servomotor elegido cuenta con un par fuerza de 11kg/cm alimentado a 6V y engranajes metálicos. El engranaje metálico le aporta durabilidad a la vida útil del motor, sobre todo si éste ha de mover cargas pesadas o llegar a su límite de carga. Los motores son fáciles de acoplar a cualquier robot o máquina mediante 4 tornillos, además, el cableado está fabricado de forma que no haga falta ningún tipo de soldadura, ya que el cableado es de tipo hembra siendo más sencillo conectarlo a cualquier tipo de controlador. Los motores incorporan con ellos diferentes acoples para el eje de transmisión, se utilizará uno u otro en función del movimiento del eslabón que se quiera mover. Es importante destacar que el rango de recorrido de estos motores es de 180^{o}, lo cual se deberá tener en cuenta en la configuración de las articulaciones en el programa de simulación.

5.2. Controlador de motores

El controlador que se eligió fue el de la marca witMotion, está preparado para controlar hasta 16 servo motores. Se debe alimentar entre 5 y 7.2V. Una de las características más interesantes de este controlador es el programa de ordenador que han desarrollado que te permite controlar la posición y velocidad de cada servo mediante una sencilla interfaz y conexión USB. Además, también puede ser controlado desde cualquier teléfono móvil de manera inalámbrica con el uso de bluetooth y la aplicación propia de la marca. La velocidad de transmisión es de 9600Kps Los pines a los que se conectan los motores tienen identificadores de color amarillo, rojo y negro para que la conexión sea sencilla y no haya lugar a dudas. Para que el programa de simulación pueda enviar la información al controlador, y éste a los motores, es necesario transmitir la información mediante puerto serie. El controlador no proporciona esta funcionalidad de serie, pero está preparado para acoplar un convertidor. Para enviar la

información de la posiciones y velocidades de cada motor por puerto serie, el controlador tiene su propia codificación detallada en la hoja de especificaciones que proporciona.

5.3. Fuente de alimentación

La fuente de alimentación que finalmente se eligió fueron las pilas alcalinas AA con un voltaje de 1.5V por pila. El controlador debía estar alimentado con una tensión de entre 5 y 7.2V por esa razón se obtuvo un porta pilas con 4 espacios para obtener un total de 6V. Según el fabricante, estas pilas pueden mantener la carga por largos periodos de tiempoo y es una buena opción para aparatos de uso diario. De media, para esta aplicación, este tipo de pila dura entre 1 y 3 años, dependiendo del uso que se le dé, lo cual hará que los alumnos no se deban preocupar por cambiar las pilas, ya que, al ser un proyecto con fines académicos, la duración media de uso del robot será de aproximadamente 1 año.

5.4. Materiales y diseño del robot

El material de las piezas con el que se ha construido el robot es diferente para cada módulo de este. La base del robot, donde están todas las piezas apoyadas, es un círculo de madera maciza con una densidad de aproximadamente 720kg/m^3 ya que su peso aproximado es de 0.7kg y su volumen de 968cm^3 (0.000968m^3). Es necesario que la base del brazo robótico ancha y relativamente pesada para evitar que ésta se levante o balancee por el peso generado por el momento de fuerzas de los eslabones al moverse. Con estas dimensiones y peso se ha conseguido una gran estabilidad en el diseño.

Fig. 7: Base del robot. Fuente propia.

A la base principal se le han atornillado 3 patas de madera con el objetivo de elevar la plataforma a la que va anclado el primer motor y así conseguir que éste quede fijo, ya que estará bien sujeto a una plataforma anclada a la base de madera. Este primer motor constituye la primera articulación de revolución del brazo robótico. A él, está sujeto el primer eslabón. Entre la plataforma y el eslabón móvil se han colocado 3 tornillos para que sirvan de apoyo del eslabón ya que al moverse el robot se observó que el mismo eslabón balanceaba de manera que podría dañar el eje de transmisión del motor.

Sobre el primer eslabón se fijó la base a la que iría anclado el siguiente motor, que constituye la segunda articulación de revolución. Tanto las plataformas fijas donde se anclan los dos primeros motores como el primer eslabón, son 2 placas de aluminio muy finas cuyo interior está relleno de poliestireno para aumentar el grosor de las piezas y conseguir una mayor rigidez.

En la siguiente imagen se pueden observar los tornillos sobre los que se apoya el primer eslabón móvil y también se puede apreciar el material del que está hecho la base de los dos primeros motores y del eslabón.

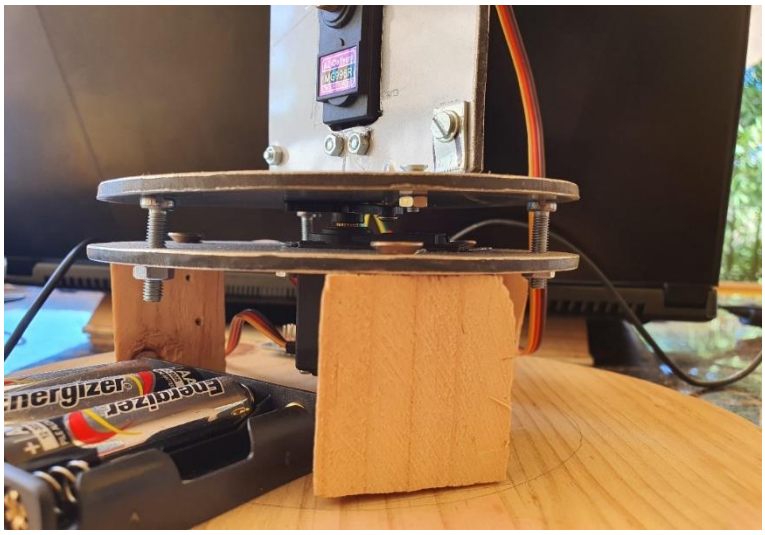


Fig. 8: Imagen del aluminio con relleno de polipropileno y apoyo del primer eslabón sobre los tornillos. Fuente propia.

El segundo y el tercer eslabón son láminas de aluminio muy ligeras. Sobre el segundo eslabón se encuentra anclado el tercer, y último, motor. Aunque estas láminas no tienen la misma rigidez que las piezas anteriores, se han elegido debido a su ligereza y que pueden soportar de forma holgada el peso del motor y pueden realizar los movimientos sin demasiado esfuerzo.

Fig. 9: Segunda y tercera articulación. Fuente propia.

Para asegurar la sujeción y estabilidad de los motores en el robot, se han anclado mediante 4 tornillos y 4 tuercas por los 4 agujeros que ofrece cada motor para ello. El diseño final del robot, aunque simple, está pensado para poder añadir más motores y aumentar los grados de libertad. Posee tres motores relativamente potentes tanto en par como en resistencia de materiales (engranajes metálicos) para poder soportar el peso de varios motores más. Al igual que se podrían añadir estos motores para aumentar los grados de libertad, se podrían añadir también para crear una pinza al extremo del último eslabón. Todos estos extras dependerán de los objetivos de la asignatura que utilice el robot.

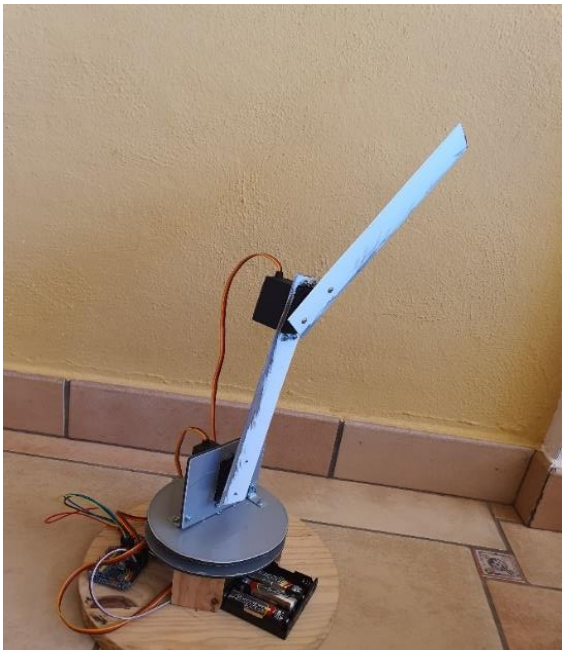


Fig. 10: Diseño final del robot. Fuente propia.

Todas las piezas que se acaban de explicar fueron diseñadas previamente con el software de diseño SolidWorks, exceptuando los motores, que se obtuvieron de una página web llamada GrabCAD en la que se suben múltiples diseños de piezas y componentes. SolidWorks es un programa de diseño CAD 3D (diseño asistido por computadora) para modelar piezas y ensamblajes en 3D y planos en 2D.

Este proceso fue uno de los más importantes del proyecto ya que el diseño que se adopte deberá ser funcional en la realidad, además, éste deberá importarse al programa de simulación donde se realizará toda la programación y las pruebas sobre el mismo. El robot físico y el diseñado desde el programa deben ser idénticos para que cuando se realice la simulación, los movimientos del robot real se correspondan con los del programa.

Fig. 11: Diseño del brazo robótico en SolidWorks. Fuente propia.

5.5. Software de simulación CoppeliaSim

El software de simulación que se ha utilizado es el CoppeliaSim. Este es un programa de simulación robótico con un entorno de desarrollo integrado basado en una arquitectura de control distribuido, es decir, cada objeto/modelo se puede controlar individualmente a través de un script integrado. Es un programa muy versátil para aplicaciones de robots. Se pueden utilizar los lenguajes C, C++, Python, Java, Lua, Matlab u Octave. En este proyecto se han escrito los scripts en lenguaje Lua. El programa ofrece la posibilidad de crear una interfaz de usuario para poder interactuar con el robot durante la simulación. Se ha utilizado algún módulo predefinido del programa, como por ejemplo, el módulo de la cinemática inversa, en el que el propio programa calcula las posiciones que deben tomar las articulaciones del robot para que el extremo de este llegue a una posición determinada.

Una parte fundamental del proyecto es la comunicación del software de simulación con el controlador de los motores. Según la hoja de especificaciones del controlador, esta comunicación se puede realizar mediante puerto serie. El puerto serie es una interfaz de comunicación digital de datos en la que la información se transmite de forma secuencial bit a bit, es decir, para enviar toda la información, se debe enviar cada bit por separado y hasta que no se recibe el bit actual no puede lanzarse el siguiente. Aunque parezca un tipo de conexión bastante lenta, actualmente hay versiones mejoradas que no tienen nada que ver con los puertos serie más antiguos. El controlador que se ha utilizado no tiene conexión por puerto serie directa, sino que necesita de un convertidor para su conexión. Para enviar la información al controlador se tiene que codificar previamente. Para dicha codificación es necesario consultar con la hoja de especificaciones del fabricante. En este caso, el código que se envía debe estar en formato hexadecimal.

Fig. 12: Adaptador para conexión por puerto serie. Fuente propia.

6. Justificación detallada de la solución adoptada

6.1. Primeros bocetos

El brazo robótico fue diseñado empleando el software de diseño SolidWorks. Antes de hacer el diseño con este programa se hicieron una serie de bocetos para tener claras las piezas y su futuro ensamblaje.

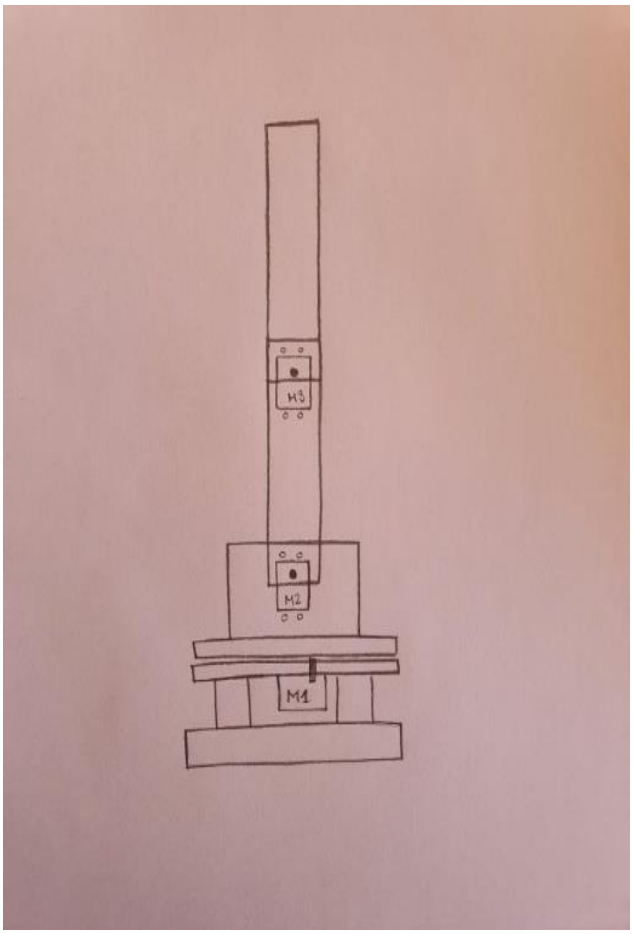


Fig. 13: Boceto final del brazo robótico. Fuente propia.

6.2. Diseño en SolidWorks y exportación a CoppeliaSim

Como ya se ha comentado, el diseño ha ido evolucionando constantemente con el objetivo de hacerlo más ligero y reducir los costes totales de fabricación. Mediante el programa SolidWorks, se diseñaron todas las piezas del robot, en ficheros separados, con sus correspondientes agujeros de taladro para los tornillos representando así las piezas con el mayor realismo posible. Una vez diseñadas, se procedió al ensamblaje de estas, para comprobar la ausencia de errores y para tener una visión completa del brazo robótico. Cuando se comprobó que todo estaba correcto, fue necesario exportar las piezas de las que se componía el robot al programa de simulación para poder comprobar las características dinámicas de las piezas moviéndose en conjunto y programar el funcionamiento del brazo. Para ello, se abrió el archivo del tipo .SLDASM (archivo de ensamblaje generado por SolidWorks) donde se encontraba el ensamblaje de las piezas. Desde *Archivo->Pack and go*, se exportaron todas las piezas al tipo de archivo .STL el cual es compatible con el programa de simulación. Para importar las piezas al programa de simulación CoppeliaSim se seleccionó *File->Import->Mesh*. Se seleccionaron todos los archivos .STL que se habían generado y se importaron al programa. Una vez importados, las piezas no estaban colocadas tal y como se encontraban en el ensamblaje, hubo que voltearlas un determinado número de grados, e incluso, modificar la posición de alguna ellas. Para ello hay herramientas dentro del propio programa que permiten copiar la posición u orientación de otros objetos para solucionar este problema. Una vez se colocaron las piezas correctamente, se simplificaron las formas de las figuras, es decir, se copió cada pieza adaptándola a una forma geométrica pura, como por ejemplo, un cuboide, un cilindro o una esfera. De esta forma es más sencillo para el programa y la tarjeta gráfica realizar los movimientos en los eslabones que producirán las articulaciones. Además, será menos propenso a generar errores o problemas en la simulación.

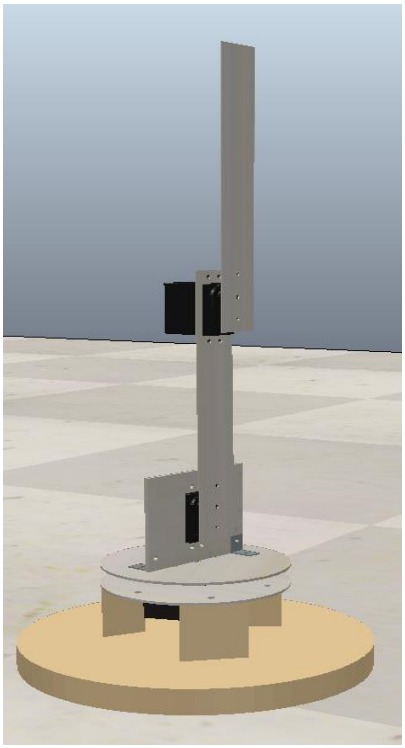


Fig. 14: Piezas importadas al programa CoppeliaSim. Fuente propia.

6.3. Configuración de articulaciones en CoppeliaSim

Lo siguiente que se debe incorporar en la simulación son las articulaciones. Hay diferentes tipos de articulaciones:

- Revolución: Giran alrededor de un eje.
- Prismática: Desplazan un eje.
- Esféricas: Giran alrededor de un punto cambiando la orientación.

En este caso, las tres articulaciones que se han creado han sido de revolución. Para añadirlas en el programa, se debe añadir un *Revolution Joint*, trasladarlo espacialmente al eje del motor y jerarquizarlo en el árbol de componentes.

Una vez está colocado se debe configurar:

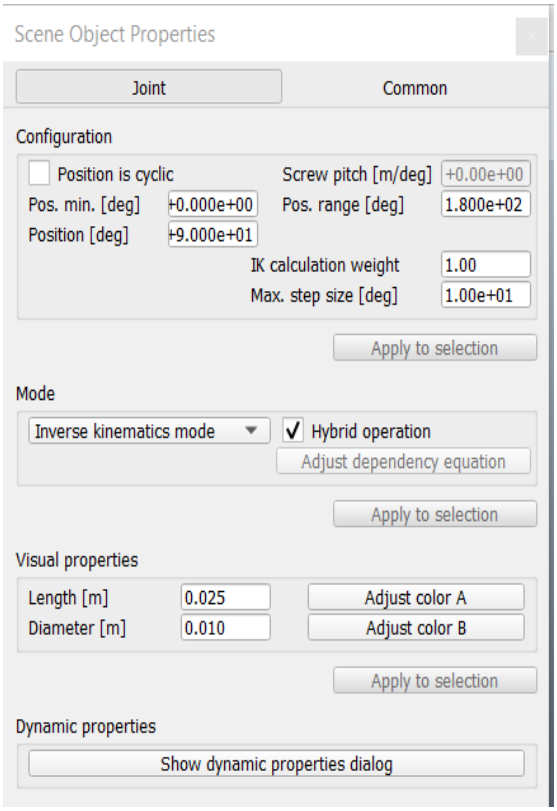


Fig. 15: Ventana de configuración de propiedades de la primera articulación. Fuente propia.

En primer lugar, se desactivará la casilla de '*Position is cyclic*' para poder ajustar las posiciones máximas y mínimas que puede alcanzar la articulación para adecuarlas al servo motor real. El rango de posiciones es de 0 a 180º. La posición inicial en la que deben colocarse todas las articulaciones es de 90º ya que así estarán los motores en el robot físico. Lo siguiente que se deberá configurar será el modo de funcionamiento. Hay varios modos:

- *Passive mode*: La posición de la articulación no se controla directamente (está fija), aunque se puede modificar por programación (API)
- *Inverse kinematics mode*: La articulación es pasiva pero su posición se modifica con el módulo de cinemática inversa. Puede trabajar en modo híbrido, es decir, un modo de control de par/fuerza, pero cuya posición de referencia la establece el módulo de cinemática inversa.
- *Dependant mode*: La posición de la articulación depende de la posición de otra articulación. Puede trabajar en modo híbrido.
- *Torque/Force mode*: La articulación se controla a través del motor de físicas y puede estar actuado por un motor o puede estar libre (sin motor). Sus valores de referencia se pueden establecer por programación (API).

En este proyecto se querían ofrecer varios modos de funcionamiento, uno de ellos era el control por motores del robot, es decir, cambiar la posición de cada motor para que el brazo se fuera moviendo. Otro modo era el de posición, mover el brazo robótico siguiendo unos ejes de coordenadas. Para el primero era necesario configurar las articulaciones en modo *Torque/Force Mode* y para el segundo modo debía ser en *Inverse kinematics mode*. Debido a esta incompatibilidad de configuración se optó por elegir el modo *Inverse kinematics mode* debido a que su funcionalidad es mayor, y es preferible mover un brazo robótico por ejes utilizando la cinemática inversa (que calcula la posición de cada motor), que, por motores, donde el control de la posición no está tan controlado.

Presionando la pestaña *'Show Dynamic properties dialog'* de la ventana de la configuración de la articulación se accede a la siguiente ventana:

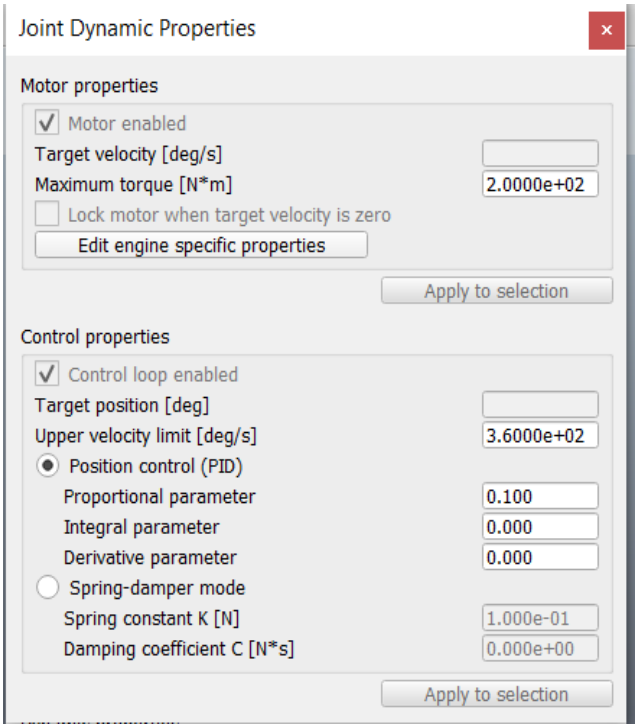


Fig. 16: Ventana de configuración de propiedades dinámicas de la primera articulación. Fuente propia.

Al estar la articulación en modo de cinemática inversa, está todo desactivado excepto *'Position Control (PID)'* y *'Spring-damper mode'*. Se debe seleccionar el control de posición (PID). Debido a que no necesitamos una gran precisión ni velocidad de ajuste, se pueden dejar los parámetros del control por defecto.

6.4. Configuración del módulo de cinemática inversa en CoppeliaSim

Una vez configuradas las articulaciones, se debe configurar también el módulo de la cinemática inversa. En primer lugar, se deben crear 2 *dummies* colocados en el extremo de la última articulación. Uno de ellos estará jerarquizado para que siempre esté en el extremo, y el otro solo dependerá de la base del robot, lo cual hará que pueda moverse libremente. El primer *dummy* (el que siempre está en el extremo del robot), seguirá al segundo, (controlado desde la interfaz de usuario), para ello, el módulo de cinemática inversa del propio programa hará los cálculos necesarios para saber la posición que deberá adoptar cada motor. Es importante destacar que la pieza o conjunto de piezas que sean hijos directos, es decir, que dependa directamente de la articulación de revolución, deben ser dinámicas y responsables tal y como se marca en la siguiente imagen.

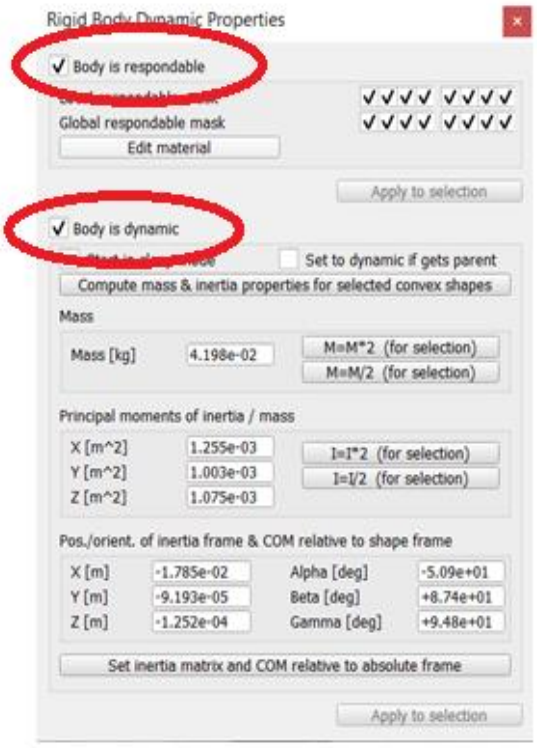


Fig. 17: Ventana de configuración de propiedades dinámicas de una forma pura, hijo directo de una articulación. Fuente propia.

También cabe destacar que esta pieza o conjunto de piezas, que serán dinámicas y responsables, deben ser puras para una no sobrecargar el programa ni la tarjeta gráfica cuando se produzca el movimiento, además

generará menos problemas dinámicos durante la simulación. Las piezas puras serán hijos directos de las articulaciones, pero no serán visibles, al igual que las articulaciones de revolución, ya que se ubicarán en una capa no visible. En la capa visible se mostrarán las formas no puras representando de la forma más realista posible el robot.

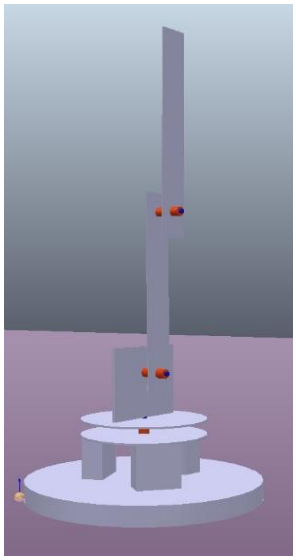


Fig. 18: Brazo robótico en la capa no visible del programa CoppeliaSim. Fuente propia.

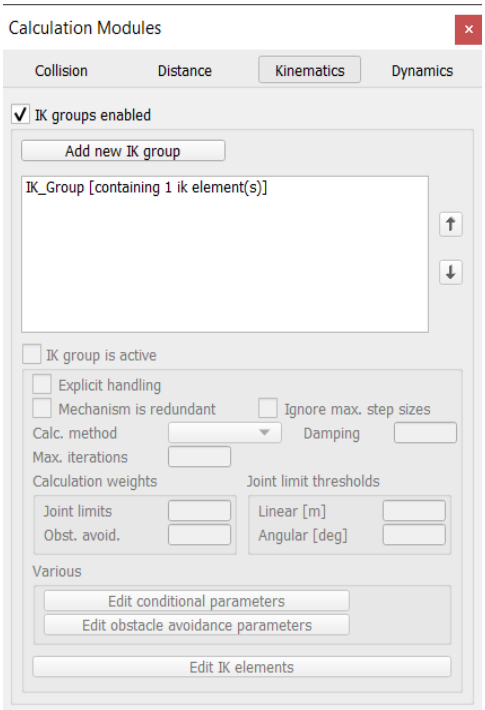


Fig. 19: Ventana de módulos de cálculo. Fuente propia.

Para configurar el módulo de cinemática inversa, se debe acceder a *Tools->Calculation Module Properties* y seleccionamos la pestaña *'Kinematics'*.

Debe estar seleccionada la casilla de '*IK groups enabled*' y presionar el botón '*Add new IK group*'. En el cuadro de texto se verá el grupo creado con el nombre '*IK_Group*', al seleccionarlo con el ratón se activarán todas las funciones de la ventana, y para configurarlo se deben seleccionar las casillas '*IK group is active*' y '*Ignore max. step sizes*'. En '*Calc. Method*' seleccionar '*DLS*', y en las casillas de '*Damping*' y '*Max iterations*' escribir 0.01 y 100 respectivamente. Con esta configuración se ejecuta la cinemática inversa de forma relativamente rápida y con una precisión más que aceptable. Al presionar el botón '*Edit IK elements*', nos abre la siguiente ventana:

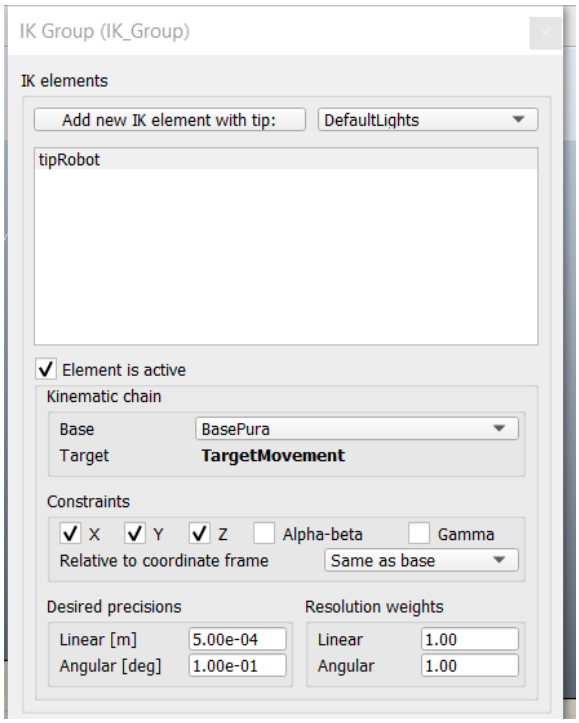


Fig. 20: Ventana de configuración del módulo de cinemática inversa. Fuente propia.

En la parte superior, en la pestaña derecha, se debe seleccionar el *dummy* que irá anclado al extremo del robot, que en este caso se ha nombrado '*tipRobot*'. A continuación, se debe seleccionar la casilla '*Element is active*', seleccionar como '*Base*' la base del robot, y en el '*Target*' seleccionar el *dummy* libre, que en este caso se ha nombrado '*TargetMovement*'. En la última parte de la configuración del módulo, la parte de '*Constraint*', se deberían seleccionar todas las casillas si se quisiera hacer un seguimiento de la orientación de los 'dummies', pero en este caso, al ser un brazo con 3 grados de libertad y ninguno modifica la orientación, no se deben seleccionar, tan solo se seleccionan las 3 primeras casillas correspondientes a la posición.

6.5. Espacio de trabajo en CoppeliaSim

Todos los brazos robóticos tienen un espacio de trabajo también conocido como *workspace*. Este espacio está compuesto por todos los puntos a los que el robot es capaz de alcanzar con su elector-final. El espacio de trabajo es de gran importancia ya que es necesario conocer la capacidad espacial del robot y sus limitaciones de movimiento. Para obtener el espacio de trabajo de nuestro robot se ha procedido de la siguiente manera:

En primer lugar, configuraremos las articulaciones en modo par/fuerza, a continuación, se hará un barrido con la combinación de las posiciones de las articulaciones y se guardaran los puntos que se alcancen por el extremo del robot en un fichero. Las posiciones que se generarán en el barrido deberán filtrarse para eliminar aquellas en las que se produzca algún tipo de colisión, ya sea con el propio robot o con el suelo. Desde el software Matlab, cargaremos el fichero generado y guardaremos en dos variables los puntos generados en los ejes 'x' y 'z' correspondientes con el plano en el que se ha producido el barrido. A continuación, se hará una triangulación para generar un sólido en .STL y poder importarlo al modelo. Al ejecutar el *script* de Matlab y crearse el sólido, se importará con el mismo procedimiento que se ha seguido para importar el resto de las piezas y se ajustará al robot. Además, al ser un sólido de 2 dimensiones, deberá estar jerarquizado para que cuando el brazo se mueva alrededor del eje 'z', el sólido se mueva con él, de forma que el '*dummy*' del robot esté siempre visible en el sólido del espacio de trabajo.

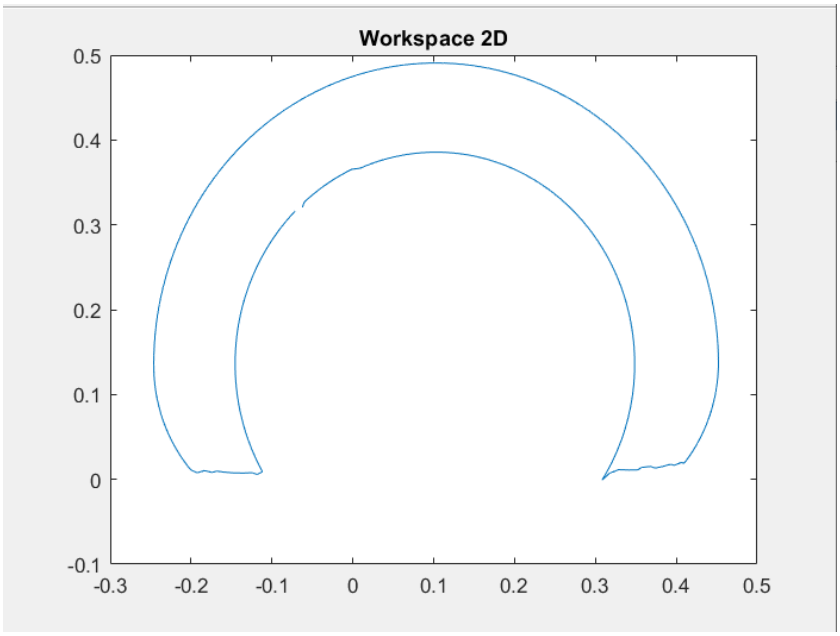


Fig. 21: Contorno del sólido del espacio de trabajado generado por el software Matlab. Fuente propia.

6.6. Creación de trayectoria en CoppeliaSim

Para el control del brazo robótico, además del control manual por ejes, se le ha añadido una pequeña trayectoria a seguir de forma automática. Esta creación de trayectorias puede ser muy útil en el sector industrial, ya que se puede repetir el mismo movimiento de manera indefinida. En cualquier proceso industrial en el que el brazo robótico tenga que ir desde un punto A a un punto B de manera cíclica se puede implementar esta técnica. Para ello debemos añadir un camino o *path* desde *Add->Path->Segment type.* Una vez creado el camino, crearemos un *dummy* dependiente del camino, es decir, hijo del elemento *path*. Para configurar el camino debemos tenerlo seleccionado y pulsar sobre el séptimo icono de la barra de herramientas de la derecha: 

Se abrirá la siguiente ventana:

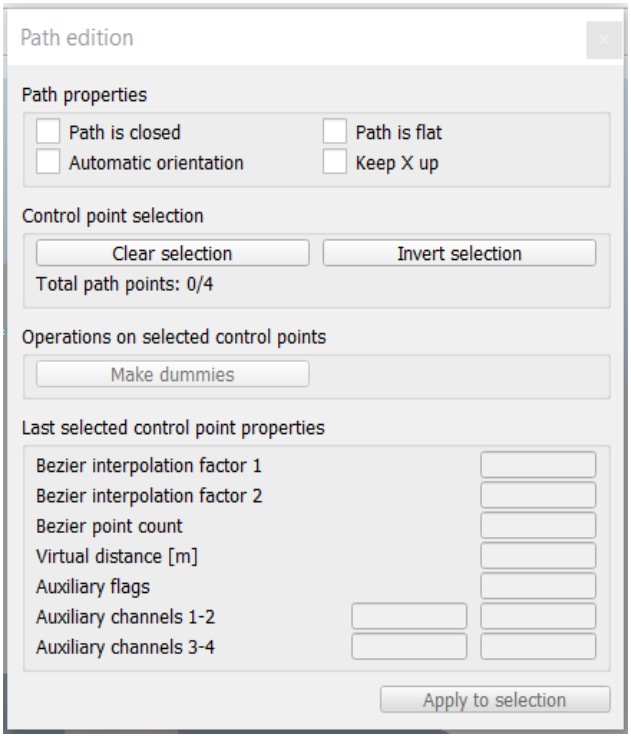


Fig. 22: Ventana de configuración de la trayectoria. Fuente propia.

En la parte superior se pueden observar 4 casillas, si se seleccionara la primera, '*Path is closed*', el último punto de control se vincularía al primero para cerrar el camino y se convertiría en una ruta cíclica (se necesitan un mínimo de 3 puntos de control para seleccionarla). La segunda casilla, '*Path is flat*', hace que todos los puntos de control estén restringidos al plano z=0 del marco de referencia local del camino. El tercer punto, '*automatic orientation*', al estar deshabilitada, el usuario puede determinar la orientación del punto de control. La última casilla, '*Keep X up*' está relacionada con la anterior, ya que no se seleccionará la anterior, esta no tiene relevancia. De estas 4 casillas la única que podría estar seleccionada en este caso podría ser '*Path is closed*'. No se ha seleccionado ya que no era el modo de funcionamiento que se deseaba para el brazo.

El control de selección del punto (*Control point selection*) es simplemente para seleccionar un punto o grupo de puntos. El resto de la ventana no se ha modificado.

6.7. Pruebas de simulación en CoppeliaSim

En las primeras pruebas de simulación que se realizaron, al mover el robot utilizando el módulo de la cinemática inversa, había posiciones en las que el brazo no podía llegar ya fuera porque la longitud de los eslabones no se lo permitía o porque simplemente no era una posición válida. En estos puntos, las piezas del brazo robótico se separaban y se producía un error en la simulación. Esto sucedía debido a que las piezas eran independientes unas de otras. Para solucionar el problema se tuvieron que agrupar las piezas de la siguiente manera: Las agrupaciones se hicieron antes y después de cada articulación, es decir, todas las piezas de las que dependía una articulación se agruparon, y las piezas que dependían de la propia articulación se agruparon en otro módulo. Además, también se separaron las piezas puras de las formas irregulares. Finalmente, el árbol de componentes quedó tal que así:

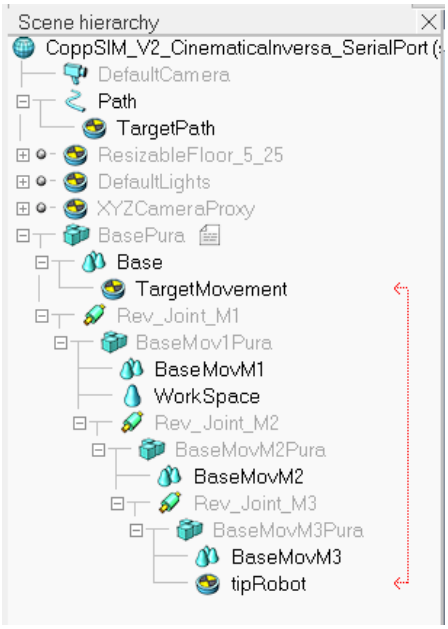


Fig. 23: Árbol de componentes en el programa CoppeliaSim. Fuente propia.

En las agrupaciones se excluyeron los *dummies* y el espacio de trabajo.

6.8. Scripts en CoppeliaSim

La configuración del brazo robótico en el programa de simulación ya se ha completado, para efectuar el movimiento del robot según los modos de funcionamiento descritos anteriormente se deberá realizar mediante la programación de un *script*.

El *script* sobre el que se ha implementado el código es un *Non-threaded child script*. El denominado *child script* (según el software CoppeliaSim) es un *script* de simulación. En cada escena creada puede haber un número ilimitado de este tipo de *scripts*. Cada uno puede representar una pequeña colección de rutinas y deben estar sujetos o adjuntos a algún objeto de la escena. El lenguaje en el que se trabaja es Lua. El objeto al que se ha adjuntado el *script* es la base (pura) del robot. Un *Non-threaded child script* se caracteriza por realizar alguna tarea y devolver el control al *script* principal. Las funciones de llamada al sistema que se han utilizado han sido:

- sysCall_init: Es una función obligatoria y se ejecutará tan solo una vez, lo cual sucederá la primera vez que el *script* sea llamado.
- sysCall_actuation: Se trata de la función de actuación, es llamada en cada paso de la simulación.

6.8.1. Función 'sysCall_init'

Al iniciar el programa, se ejecuta la función sysCall_init (dada por el propio software) en la que se crearán las variables para almacenar los manejadores de los objetos que utilizaremos durante la ejecución del programa tales como las articulaciones de revolución, la trayectoria, el *dummy* objetivo y el *dummy* que se ha creado para el seguimiento de la trayectoria. Se inicializan 4 variables con un valor asignado. La primera de ellas corresponde con la posición actual del robot. Esta primera variable es de tipo vector y tiene 3 posiciones, cada posición corresponde con la coordenada en los ejes 'x', 'y' y 'z'. La segunda variable es de tipo booleano y se utilizará para saber si el robot está siguiendo o no la trayectoria. La tercera variable posee el valor de la velocidad de transmisión del puerto serie. La última variable inicializada en esta función configura el puerto por el que se realizará la comunicación serie. La última parte de la función de inicialización es la configuración de la interfaz de usuario en la que se han creado 3 *sliders,* cada uno corresponde con un eje de coordenadas. Los dos primeros ('x' e 'y') van desde los valores -100 a 100, y el tercero ('z') está comprendido entre -100 y 0. Al inicializarse el programa todos los *sliders* estarán colocados en el valor de 0.

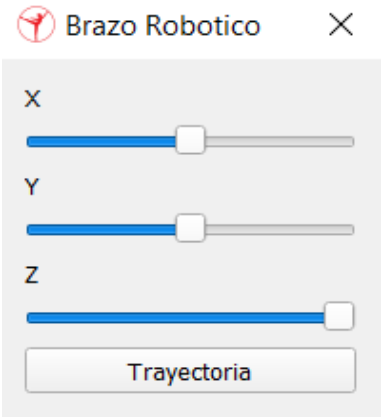


Fig. 24: Interfaz de usuario del programa CoppeliaSim. Fuente propia.

Cada *slider* tiene una función asignada en la que se calcula la posición del robot en función de la posición del *slider.* Las funciones correspondientes a los ejes 'x' e 'y' son prácticamente idénticos ya que tienen el mismo recorrido en el *slider* (200 valores). El eje 'z' en cambio variará ligeramente debido a la reducción de valores que puede tomar. Un patrón común en las 3 funciones es el asignar el valor de falso a la variable 'mov_path', que indica si el robot se encuentra realizando la trayectoria predefinida.

```
function onXChange(uiHandle, id, newValue)
    mov_path = false
    px = ((newValue*0.55)/200) + 0.1439
    position[1]=px
    sim.setObjectPosition(ref_point,-1,position)
end
```

Fig. 25: Función del script principal asociada al movimiento del primer *slider*. Fuente propia.

Para el cálculo de la posición en el eje 'x' se siguió el razonamiento basado en la siguiente gráfica:

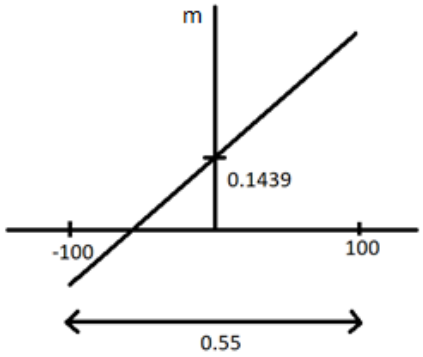


Fig. 26: Gráfica para la relación entre la posición del *slider* y la posición del robot. Fuente propia.

El valor de la posición inicial en el eje 'x' del robot es 0.1439, que será el valor que tomaremos como offset. Después de varias pruebas empíicas y la ayuda del espacio de trabajo, se ha concluido que las posiciones entre las que se moverá el robot por el eje 'x' serán -0.1311 y 0.4189. Para formar la ecuación de la recta necesitaremos calcular la pendiente:

$$Pendiente = \frac{(0.4189 - (-0.1311))}{200} = \frac{0.55}{200}$$

$$Ecuación\ de\ la\ recta \rightarrow y = a \cdot x + b$$

Donde:

- a = pendiente
- x = valor del slider
- b = offset

La variable *position* se inicializa con los valores iniciales del robot al ejecutar por primera vez el programa y se va actualizando al mover los *sliders*.

Para calcular la relación de posición del eje 'y' se sigue el mismo procedimiento que para el eje 'x' exceptuando que los valores del offset y la pendiente cambian. El rango de valores en el que se puede mover el robot sobre el eje 'y' es de 0.6 y el offset es de 0.032926.

```
8  function onYChange(uiHandle, id, newValue)
9
10     mov_path = false
11     py = ((newValue*0.6)/200) + 3.2926e-02
12     position[2]=py
13     sim.setObjectPosition(ref_point,-1,position)
14 end
```

Fig. 27: Función del script principal asociada al movimiento del segundo *slider*. Fuente propia.

Para el cálculo de la relación en el eje 'z' se ha calculado la pendiente teniendo en cuenta que el *slider* cuenta con 100 posiciones y los valores de offset y rango máximo de movimiento son 0.4944 y 0.49. Se ha dejado un pequeño margen de 0.0044 para evitar posibles colisiones con el suelo.

```
16 function onZChange(uiHandle, id, newValue)
17
18     mov_path = false
19     pz = (0.49/100)*newValue + 0.4944
20     position[3]=pz
21     sim.setObjectPosition(ref_point,-1,position)
22 end
```

Fig. 28: Función del script principal asociada al movimiento del tercer *slider*. Fuente propia.

6.8.2. Trayectoria

En la interfaz de usuario podemos observar un botón cuya etiqueta pone 'trayectoria'. Al pulsar sobre ese botón, se llamará a la función interna del programa 'sim.setPathTargetNominalVelocity' que le asignará al dummy del *'path'* creado la velocidad que le hayamos asignado, esta será la velocidad a la que se mueva el robot.

6.8.3. Puerto serie

Se ha creado una función auxiliar para convertir números decimales a hexadecimales y otra donde se codifican las posiciones, las articulaciones y el canal de conexión al controlador de motores para la comunicación por el puerto serie. En esta última se ha tenido que consultar la hoja de especificaciones del controlador de motores. El código que se envía por puerto serie consta de 5 bytes. Cabe destacar que la información se envía en formato hexadecimal, no en formato ASCII. El primer byte es siempre 0xFF, el segundo byte corresponde con el comando que se quiera ejecutar. Hay 4 tipos de comandos:

- Control de velocidad: 0x01
- Control de posición: 0x02
- Activar grupo de acciones: 0x09
- Recuperación/parada de emergencia: 0x0b

El tercer byte corresponde con el canal del controlador que estemos utilizando, los canales van desde 0 hasta 15 (16 canales). El cuarto y el quinto byte son los bytes de menor y mayor importancia respectivamente de la información que se desea enviar.

Para enviar los datos del control de velocidad, mediante el cuarto y quinto byte se debe tener en cuenta que el valor (en formato decimal) que pueden tomar estos dos bytes en conjunto va desde 1 hasta 20, correspondiendo el valor 1 a la velocidad de 9^{o}/s y el valor de 20 a la velocidad de 180^{o}/s. Sabiendo estos dos valores y que la relación es lineal, se podría calcular el valor que se debe enviar. Por ejemplo, si queremos que el motor se mueva a una velocidad de 90^{o}/s debería tomar el valor de 10.

Para el control de posición se debe hacer algo parecido, pero con otros valores. El rango de valores (en formato decimal) entre el que trabajarán el cuarto y quinto byte estará comprendido entre 500 y 2500. El valor de 500 corresponde con el ángulo 0^{o}, y el valor de 2500 con el ángulo de 180^{o}.

La función en la que se han implementado todas estas relaciones y se ha enviado la información es la siguiente:

```
41 function serialPort()
42
43     posM1 = sim.getJointPosition(m1)
44     posM2 = sim.getJointPosition(m2)
45     posM3 = sim.getJointPosition(m3)
46     posM3_inv = math.pi -posM3
47
48     convM1 = 500 + (2000/math.pi)*(posM1)
49     convM2 = 500 + (2000/math.pi)*(posM2)
50     convM3 = 500 + (2000/math.pi)*(posM3_inv)
51
```

Fig. 29: Parte inicial de la función encargada de enviar las posiciones de las articulaciones por puerto serie al robot. Fuente propia.

En primer lugar, se guardarán las posiciones actuales de las articulaciones del programa de simulación. A continuación, se convertirán los valores de dichas posiciones a las relaciones que se han explicado anteriormente, para ello se ha seguido el siguiente razonamiento. Se ha tomado como valor de offset 500, que es el mínimo valor, correspondiente con 0º, y la pendiente que se ha tomado es la longitud del intervalo comprendido entre 500 y 2500 (longitud de 2000) partido entre el rango de posiciones expresado en radianes (pi). A la pendiente se le multiplica la posición de la articulación y ya tendríamos la conversión de la posición.

```
52     --Comunicacion del motor 1
53     if(string.len(DEC_HEX(convM1)) == 3) then
54         dataL_M1 = string.sub(DEC_HEX(convM1), 2, 3)
55         dataH_M1 = string.sub(DEC_HEX(convM1), 1, 1)
56     end
57
58     if(string.len(DEC_HEX(convM1)) == 4) then
59         dataL_M1 = string.sub(DEC_HEX(convM1), 3, 4)
60         dataH_M1 = string.sub(DEC_HEX(convM1), 1, 2)
61     end
62
63     intL_M1 = tonumber(dataL_M1,16)
64     intH_M1 = tonumber(dataH_M1,16)
65     sendM1 = sim.serialSend(port,string.char(255)..string.char(2)..
66     string.char(0)..string.char(intL_M1)..string.char(intH_M1))
```

Fig. 30: Código que codifica la posición de la primera articulación y la envía por puerto serie al robot. Fuente propia.

Para enviar la información de la posición por puerto serie todavía se deben hacer algunas conversiones. En primer lugar, comprobaremos si el

valor de la posición del motor expresado en hexadecimal contiene 3 o 4 dígitos ya que dependiendo de esto se deberán tomar unos u otros como los bytes de mayor y menor importancia. Si se expresa con 3 dígitos, los bytes de mayor importancia serán el 2 y el 3, en cambio, si tiene 4, los de mayor importancia serán el 3 y el 4. Una vez se separan entre los de mayor y menor importancia, hay que volverlos a convertir a formato decimal, y es lo que se hace en las líneas 63 y 64. En las líneas 65 y 66 se envía la información mediante el comando sim.serialSend(), cuyos parámetros son el puerto COM por el que se enviará la información, y los 5 bytes explicados anteriormente.

El código para la comunicación para el segundo y el tercer motor se implementa de la misma manera:

```
68      --Comunicacion del motor 2
69      if(string.len(DEC_HEX(convM2)) == 3) then
70          dataL_M2 = string.sub(DEC_HEX(convM2), 2, 3)
71          dataH_M2 = string.sub(DEC_HEX(convM2), 1, 1)
72      end
73
74      if(string.len(DEC_HEX(convM2)) == 4) then
75          dataL_M2 = string.sub(DEC_HEX(convM2), 3, 4)
76          dataH_M2 = string.sub(DEC_HEX(convM2), 1, 2)
77      end
78
79      intL_M2 = tonumber(dataL_M2,16)
80      intH_M2 = tonumber(dataH_M2,16)
81      sendM2 = sim.serialSend(port,string.char(255)..string.char(2)..
82      string.char(3)..string.char(intL_M2)..string.char(intH_M2))
83
84      --Comunicacion del motor 3
85      if(string.len(DEC_HEX(convM3)) == 3) then
86          dataL_M3 = string.sub(DEC_HEX(convM3), 2, 3)
87          dataH_M3 = string.sub(DEC_HEX(convM3), 1, 1)
88      end
89
90      if(string.len(DEC_HEX(convM3)) == 4) then
91          dataL_M3 = string.sub(DEC_HEX(convM3), 3, 4)
92          dataH_M3 = string.sub(DEC_HEX(convM3), 1, 2)
93      end
94
95      intL_M3 = tonumber(dataL_M3,16)
96      intH_M3 = tonumber(dataH_M3,16)
97      sendM3 = sim.serialSend(port,string.char(255)..string.char(2)..
98      string.char(4)..string.char(intL_M3)..string.char(intH_M3))
99
```

Fig. 31: Código que codifica la posición de la segunda y tercera articulación y la envía por puerto serie al robot. Fuente propia.

6.8.4. Función 'sysCall_Actuation'

En la función de actuación sysCall_Actuation (que se ejecuta cíclicamente), actualizaremos constantemente el *dummy* que hace moverse el robot para que siempre esté en la posición indicada por el usuario. También llamaremos a la función SerialPort(), para enviar constantemente la información de la posición de los motores al robot físico.

Finalmente se consiguió una correcta sincronización entre el programa de simulación y el robot transmitiendo la información por el puerto serie. Además, esta sincronización de la información se realiza con bastante rapidez y exactitud. El control del brazo robótico se realiza desde la interfaz de usuario del software de simulación y sigue los modos de funcionamiento de control por ejes de coordenadas y por seguimiento de una trayectoria. En las siguientes imágenes se puede observar como el programa y el robot se mueven a la par.

6.9. Programa de simulación y robot en funcionamiento

Fig. 32: Programa y robot en funcionamiento. Prueba nº1. Fuente propia.

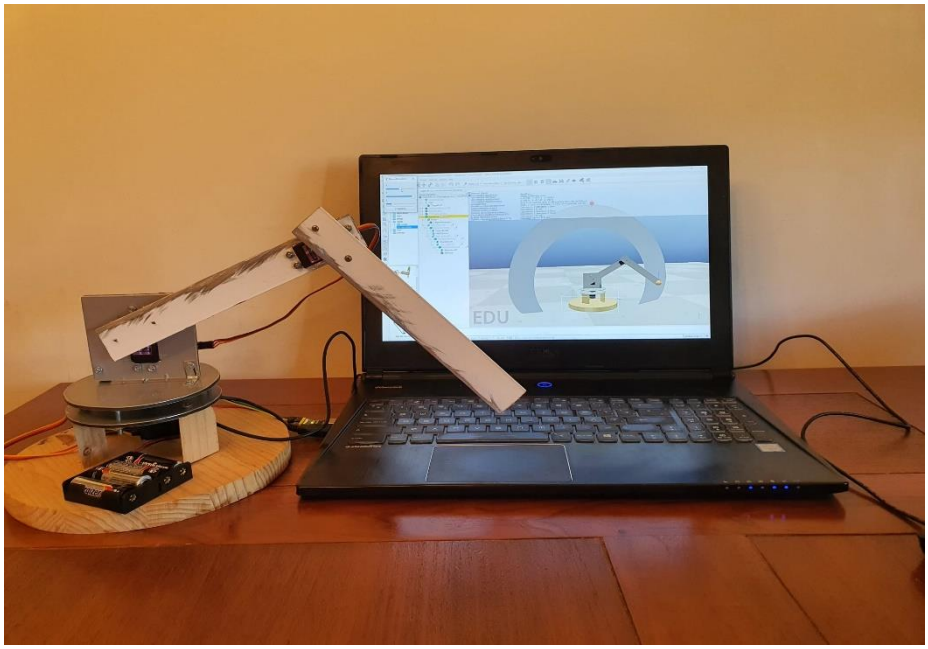


Fig. 33: Programa y robot en funcionamiento. Prueba nº2. Fuente propia.

Fig. 34: Programa y robot en funcionamiento. Prueba nº3. Fuente propia.

Fig. 35: Programa y robot en funcionamiento. Prueba nº4. Fuente propia.

7. Conclusiones y posibles mejoras

La investigación y el desarrollo del proyecto ha supuesto un aprendizaje constante, tanto a nivel personal como profesional. He aprendido y profundizado en una gran cantidad de materias relacionadas con la robótica y la programación, y gracias a este proyecto me he podido acercar un poco más al mundo de los brazos robóticos.

El robot que se ha diseñado es simple, pero eso no implica que no sea tan útil como cualquier otro, ya que está diseñado para cumplir unos objetivos específicos. Aunque se ha buscado esta simpleza en el diseño para evitar distracciones por un diseño demasiado complejo, se le pueden añadir funcionalidades tales como la ampliación de los grados de libertad, algún mecanismo en el extremo del robot como pudiera ser una pinza, o también se podrían modificar los eslabones para que fueran más o menos largos, en función de lo que se pretenda.

La parte más costosa fue la del diseño de las piezas y su ensamblaje, ya que no es una materia en la que se profundice demasiado en el propio grado, a pesar de eso, realizar estas dos tareas ha sido de gran utilidad para mi desarrollo profesional, ya que he aprendido a realizar diferentes diseños y a manejar la interfaz de un programa de diseño en 3D de forma fluida. Además, he profundizado mis conocimientos y habilidades en otras materias relacionadas directamente con el grado, como, por ejemplo, la programación del robot, las conexiones de la electrónica, la elección de componentes electrónicos, la combinación y el funcionamiento de dichos componentes...

Por último, el proyecto me ha resultado de gran interés, cuanto más investigaba, más modificaciones le realizaba al brazo, y aunque presente este brazo robótico como proyecto, seguiré realizando cambios en el diseño y la programación de este, ya que no ha sido tan solo un proyecto académico, sino también un proyecto personal en el que se ha invertido mucho tiempo e ilusión.

8. Bibliografía

[1] Cinemática inversa. Wikipedia.

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

[2] Robótica. RobotEsfera.

https://robotesfera.com/que-son-brazos-roboticos

[3] Dpto. de Ingeniería de sistemas y automática (DISA) UPV, <<Cinemática directa>>

[4] Dpto. de Ingeniería de sistemas y automática (DISA) UPV, <<Cinemática inversa>>

[5] Mini SG90 servo micromotor. AZ delivery. Amazon.

https://www.amazon.es/AZDelivery-Servo-MG90S-engranajes-incluido/dp/B086V3VP72/ref=sr_1_1?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=mini+SG90+servo+micromotor&qid=1597050360&s=tools&sr=1-1

[6] MG996R Micro digital servo motor. AZ delivery. Amazon.

https://www.amazon.es/AZDelivery-SG90-Micro-hubchrauber-MG996RDigital/dp/B07H87592P/ref=sr_1_1?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=MG996R%2BAZ&qid=1597050680&s=tools&sr=1-1&th=1

[7] Motor de dirección servo digital 25Kg. Amazon.

https://www.amazon.es/Motor-Direcci%C3%B3n-Digital-Coreless-Modelo/dp/B07QCT37J7/ref=sr_1_2?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=25kg+Motor+de+direccion+servo+digital+coreless+metal+gear&qid=1597050761&s=tools&sr=1-2

[8] Servo motor controlador de 16 canales con opción bluetooth. WitMotion. Amazon.

https://www.amazon.es/Bluetooth-Controlador-engranaje-direcci%C3%B3nlibremente/dp/B06Y3ZV4B8/ref=sr_1_1?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=bluetooth+16+canales+witmotion&qid=1597050824&s=tools&sr=1-1

[9] Servo motor controlador 16 canales. WayinTop. Amazon.

https://www.amazon.es/WayinTop-PCA9685-Interfaz-Controlador-Raspberry/dp/B07V5GVYGM/ref=sr_1_1?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=wayintop+servo+motor&qid=1597050905&s=tools&sr=1-1

[10] Arduino nano. Arduino Store.

https://store.arduino.cc/arduino-nano

[11] Batería recargable 6V, 700mAh. Tangsfire. Amazon.

https://www.amazon.es/Bater%C3%ADa-recargable-700mAh-Juguetes-Cargador/dp/B06ZXRNYBV/ref=sr_1_1?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=tangsfire+bateria+6V&qid=1597051151&sr=8-1

[12] Pilas AA. Energizer. Amazon.

https://www.amazon.es/Energizer-alcalinas-Mignon-bater%C3%ADa-12unidades/dp/B00IE4HV2Y/ref=sr_1_2?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=%2BEnergizer%2BMax%2B%2BPilas%2Balcalinas%2BAA%2FMignon%2BLR6&qid=1595867824&s=tools&sr=1-2&th=1

[13] Porta pilas AA con 4 *slots*. Amazon.

https://www.amazon.es/Portapilas-Pilas-Caja-bater%C3%ADa-para/dp/B07FDY97FS/ref=sr_1_3?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=portapilas+aa&qid=1597051209&sr=8-3

[14] Puerto serie – Qué es, para que sirve y tipos. ProfesionalReview.

https://www.profesionalreview.com/2020/03/07/puerto-serie-que-es-para-que-sirve-y-tipos/

[15] SolidWorks- Qué es y para que sirve. SolidBi.

https://solid-bi.es/solidworks/

[16] CoppeliaSim. CoppeliaRobotics.

https://www.coppeliarobotics.com/

[17] Hoja de especificaciones del controlador de servomotores. WitMotion.

https://drive.google.com/file/d/1MP4HiKtkfaAbWIOiTFHfQnZqJ8kWKUsP/view

[18] Hoja de especificaciones del adaptador USB-Puerto serie. WitMotion.

https://drive.google.com/file/d/1NIGTBXizaN26oJWw3JvhEpCVb1XKgZK/view

[19] Canal de Youtube de Leopoldo Armesto. Youtube.

https://www.youtube.com/channel/UCstH8IDCcJV_l-pbfFEn37Q

[20] Documentación de ayuda SolidWorks. SolidWorks.

http://help.solidworks.com/2018/spanish/SolidWorks/sldworks/c_detailing_drawing.htm

[21] Documentación de ayuda de CoppeliaSim. CoppeliaSim.

https://www.coppeliarobotics.com/helpFiles/

[22] Manual de referencia del lenguaje de programación Lua.

https://www.lua.org/manual/5.1/es/

[23] Ayuda y aprendizaje de Word. Microsoft.

https://support.microsoft.com/es-es/word

[24] GrabCad Community. MG996R. Diseño del servomotor SolidWorks.

https://grabcad.com/library/servomotor-mg996r-3

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos.

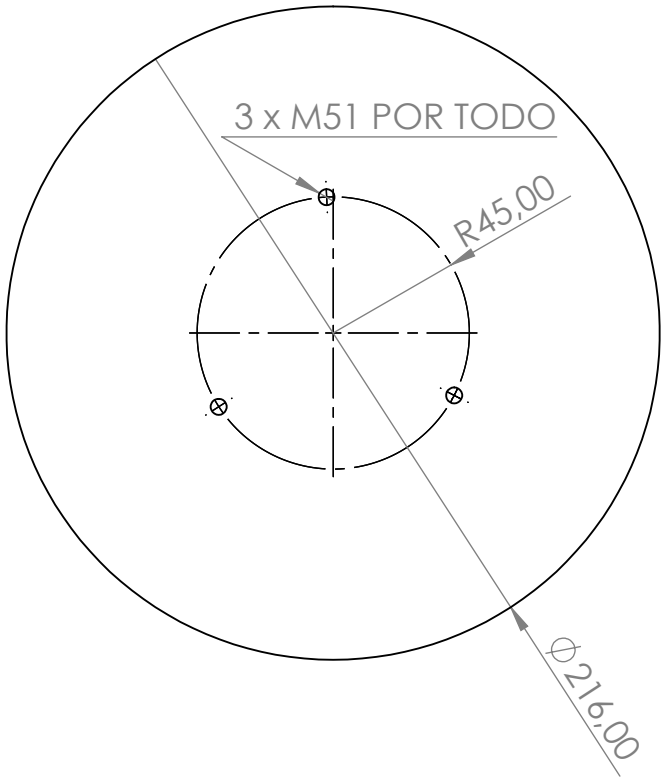
II. PLANOS

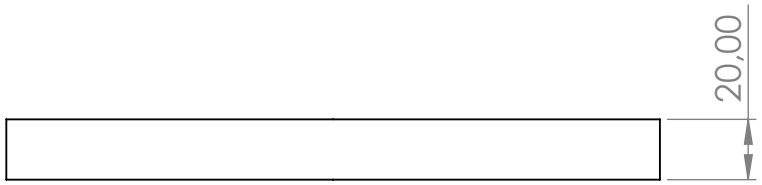
AUTOR: Vicente Ferrando Santamaría

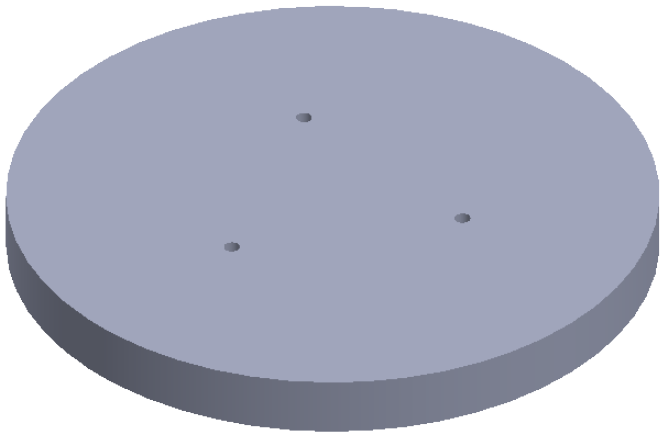
TUTOR: Leopoldo Armesto Ángel

Curso Académico: 2019/2020

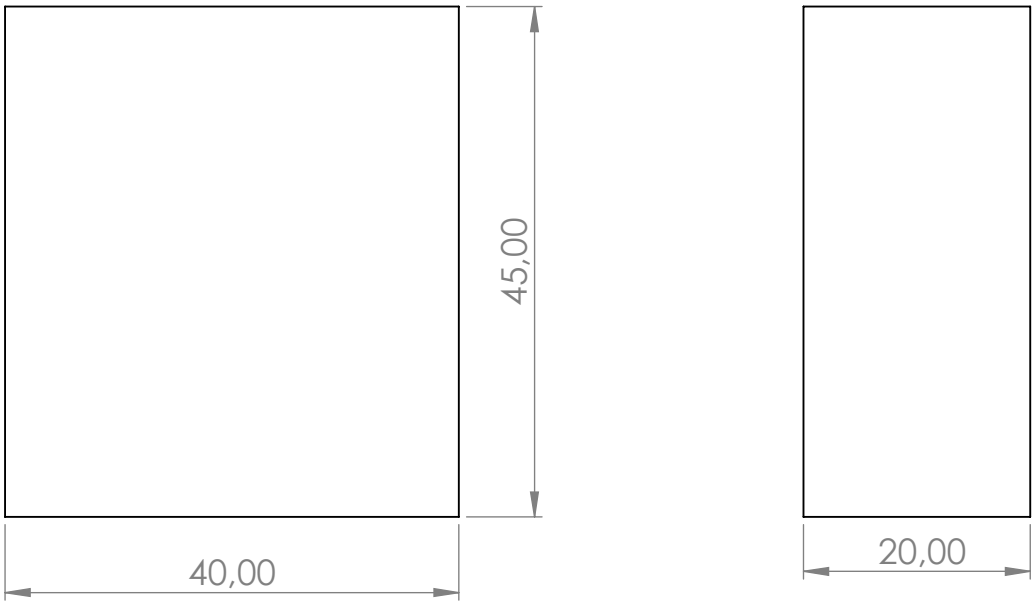
Índice planos

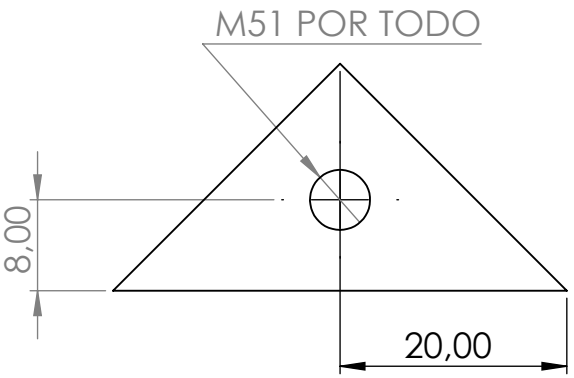


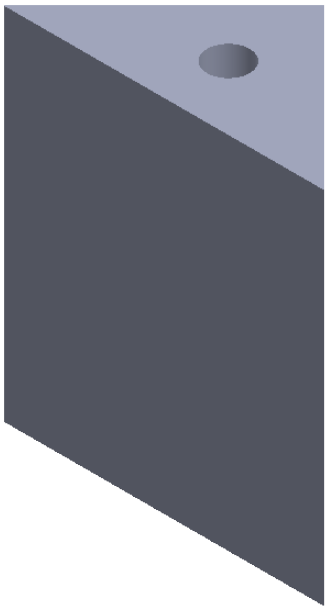




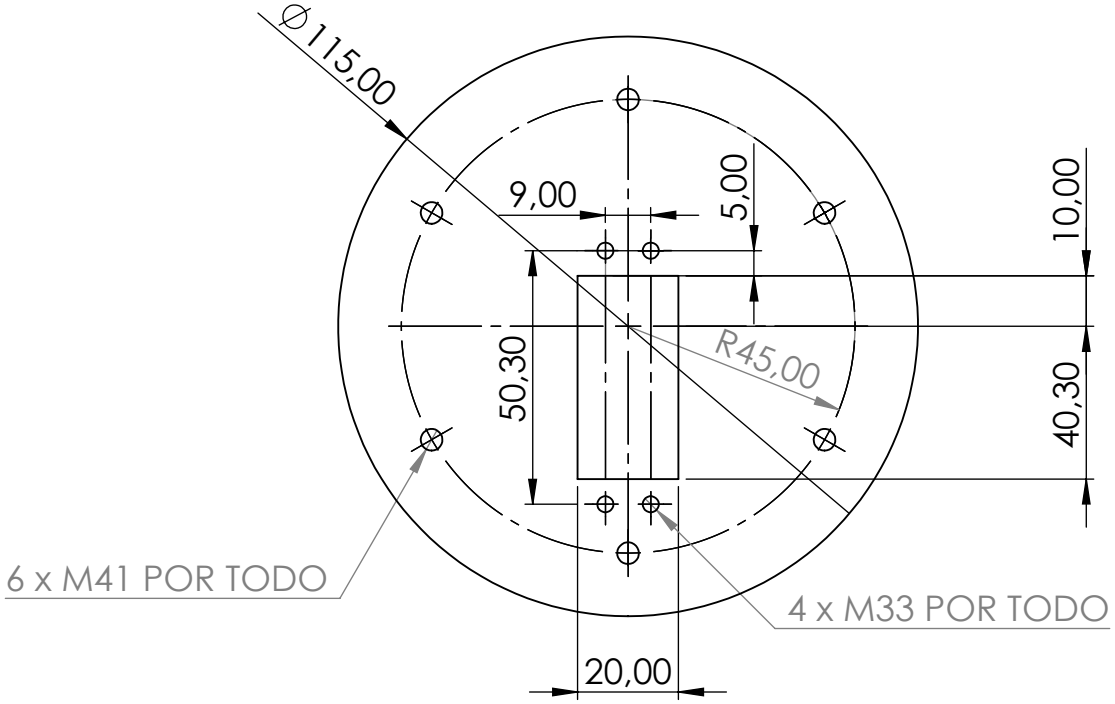
TFG EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	SI NO SE INDICA LO CONTRARIO LAS COTAS SE EXPRESAN EN MM	Título: Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos	
	Creado por: Vicente Ferrando Santamaría		
	Fecha de creación: 16/08/2020		
	ESCALA: 2:5	Plano: Base principal	Nº plano: 1 / A4
	NO CAMBIE LA ESCALA		

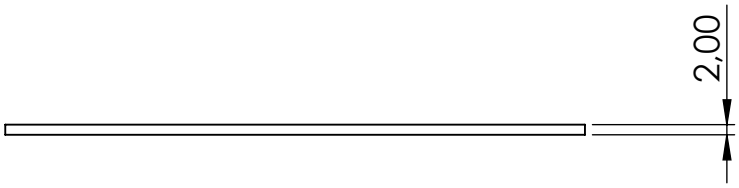


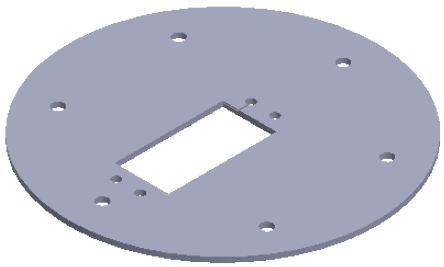




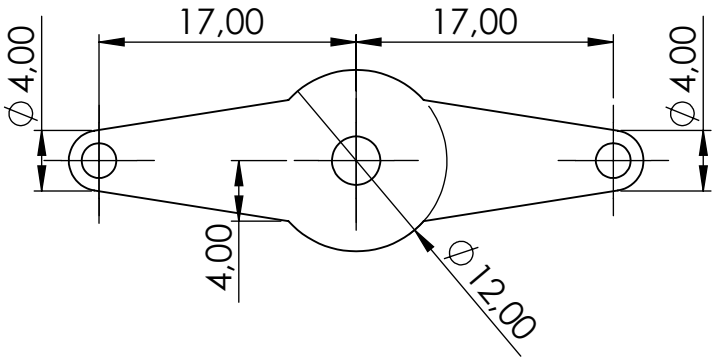
TFG EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	SI NO SE INDICA LO CONTRARIO LAS COTAS SE EXPRESAN EN MM	Título: Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos
	Creado por: Vicente Ferrando Santamaría	
	Fecha de creación: 16/08/2020	
	ESCALA: 3:2	Plano: Pata / Nº plano: 2 / A4
	NO CAMBIE LA ESCALA	

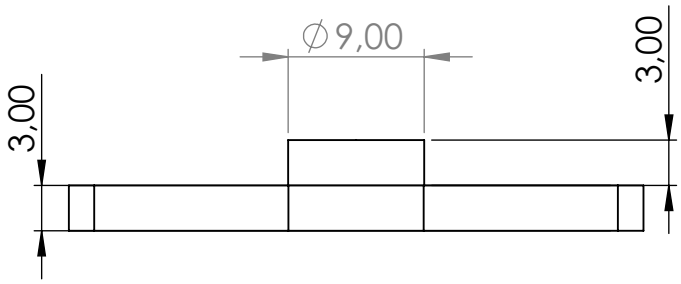


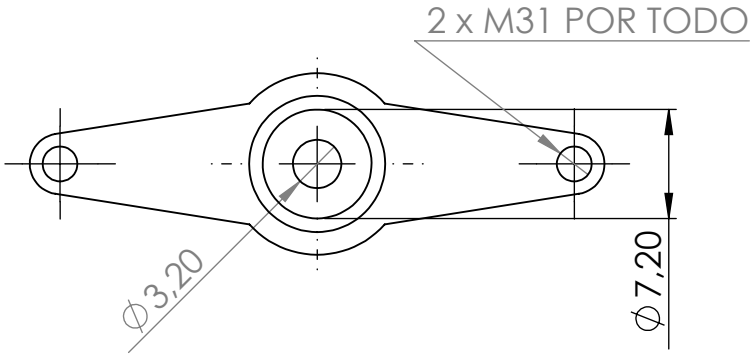


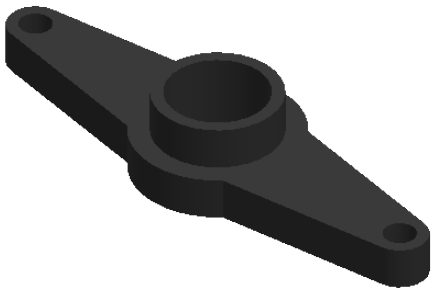


TFG EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	SI NO SE INDICA LO CONTRARIO LAS COTAS SE EXPRESAN EN MM	Título: Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos
	Creado por: Vicente Ferrando Santamaría	
	Fecha de creación: 16/08/2020	
	ESCALA: 2:3	Plano: BaseMotor1 / Nº plano: 3 / A4
	NO CAMBIE LA ESCALA	

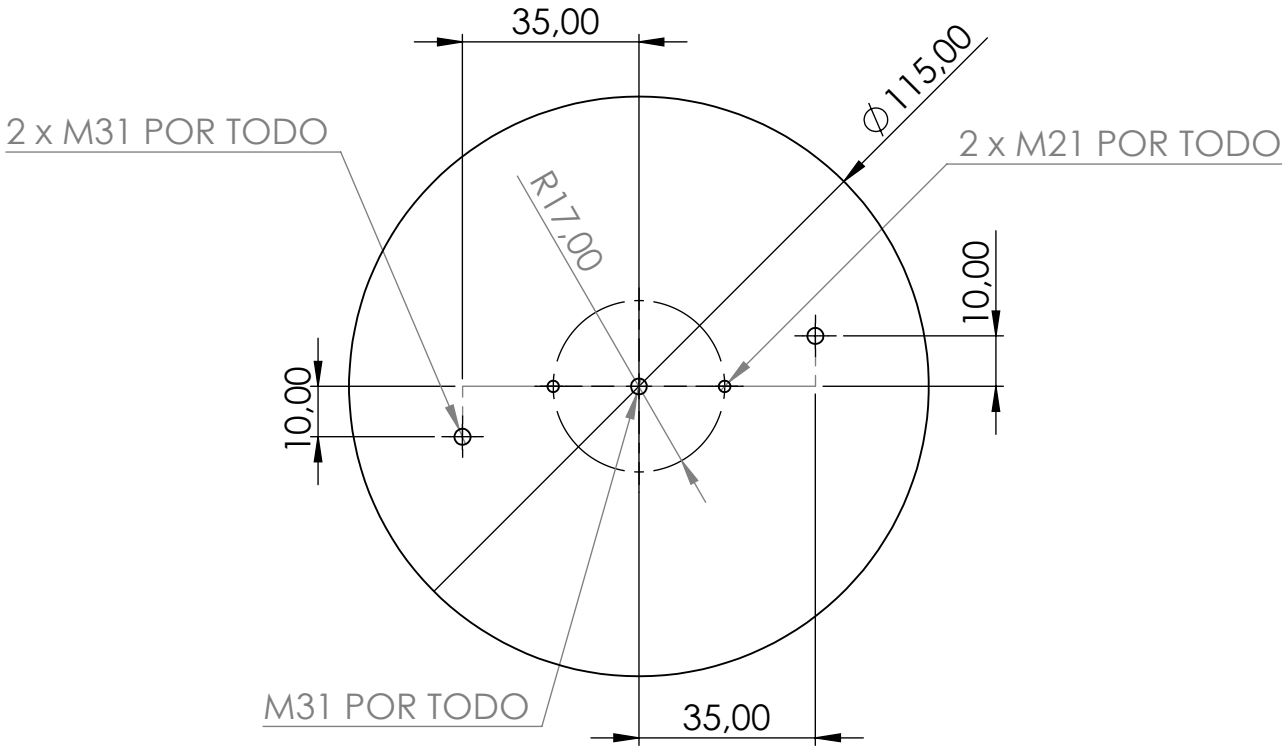


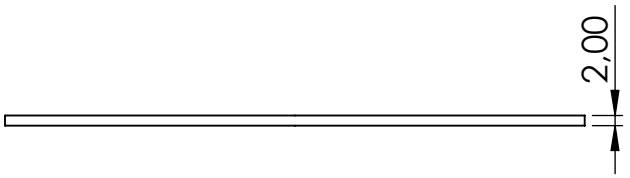


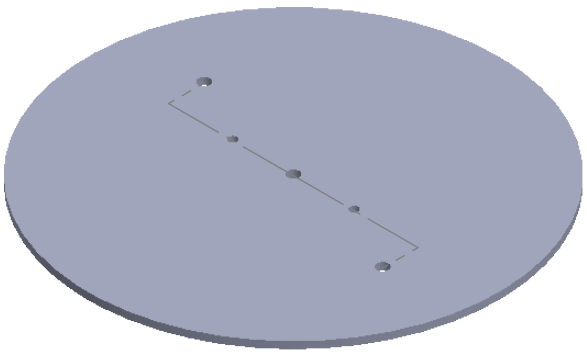




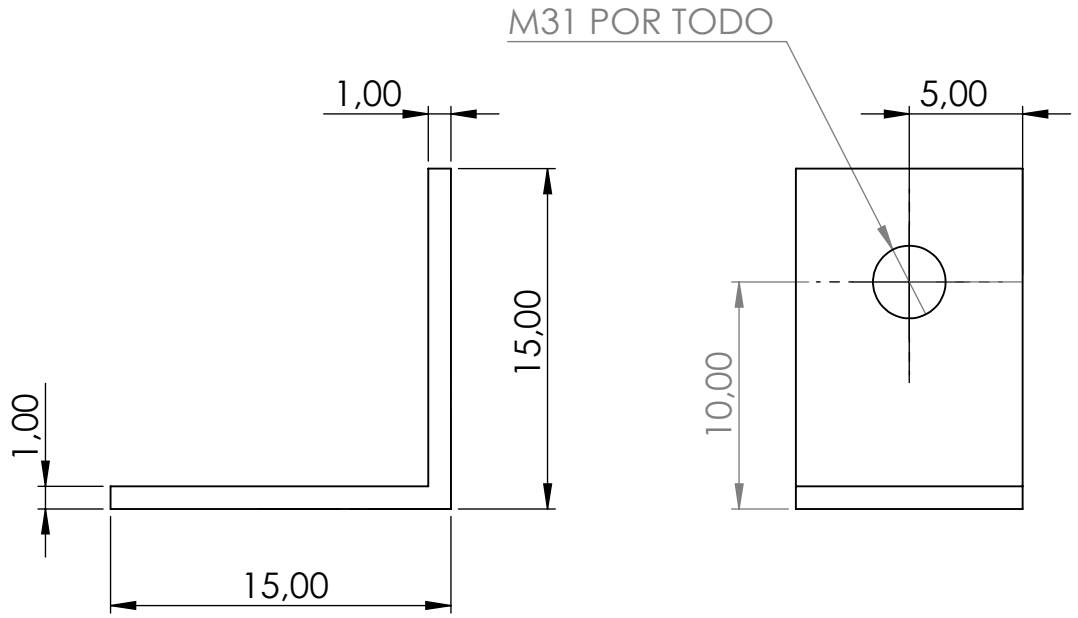
TFG EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	SI NO SE INDICA LO CONTRARIO LAS COTAS SE EXPRESAN EN MM	Título: Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos	
	Creado por: Vicente Ferrando Santamaría		
	Fecha de creación: 16/08/2020		
	ESCALA: 2:1	Plano: Acople motor	Nº plano: 4 / A4
	NO CAMBIE LA ESCALA		

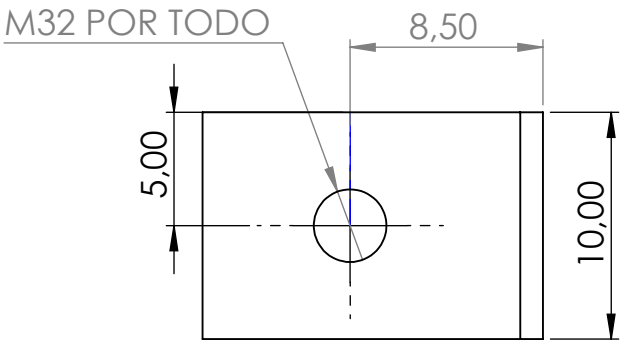


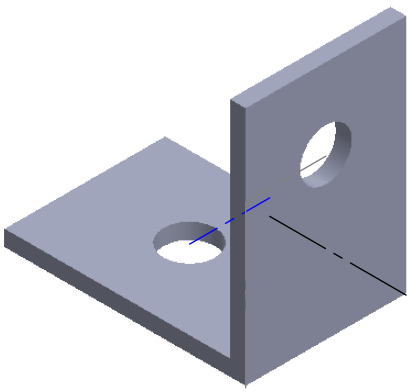




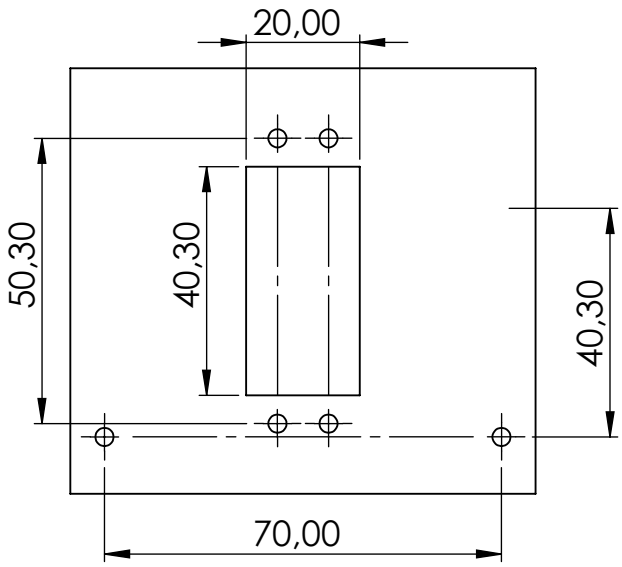
TFG EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	SI NO SE INDICA LO CONTRARIO LAS COTAS SE EXPRESAN EN MM	Título: Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos	
	Creado por: Vicente Ferrando Santamaría		
	Fecha de creación: 16/08/2020		
	ESCALA: 2:3	Plano: Eslabón 1	Nº plano: 5 / A4
	NO CAMBIE LA ESCALA		

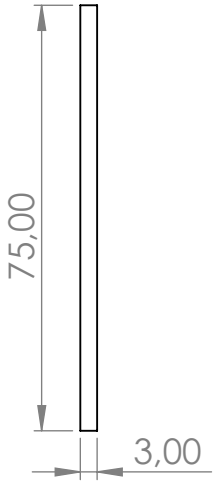


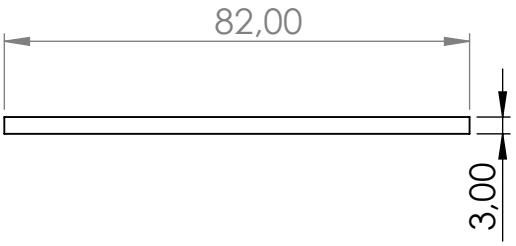


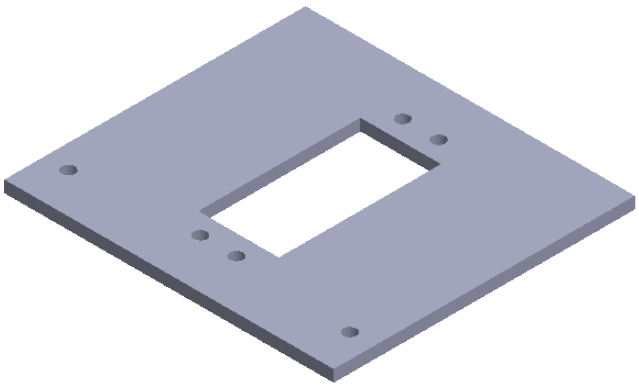


TFG EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	SI NO SE INDICA LO CONTRARIO LAS COTAS SE EXPRESAN EN MM	Título: Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos	
	Creado por: Vicente Ferrando Santamaría		
	Fecha de creación: 16/08/2020		
	ESCALA: 3:1	Plano: Escuadra	Nº plano: 6 / A4
	NO CAMBIE LA ESCALA		

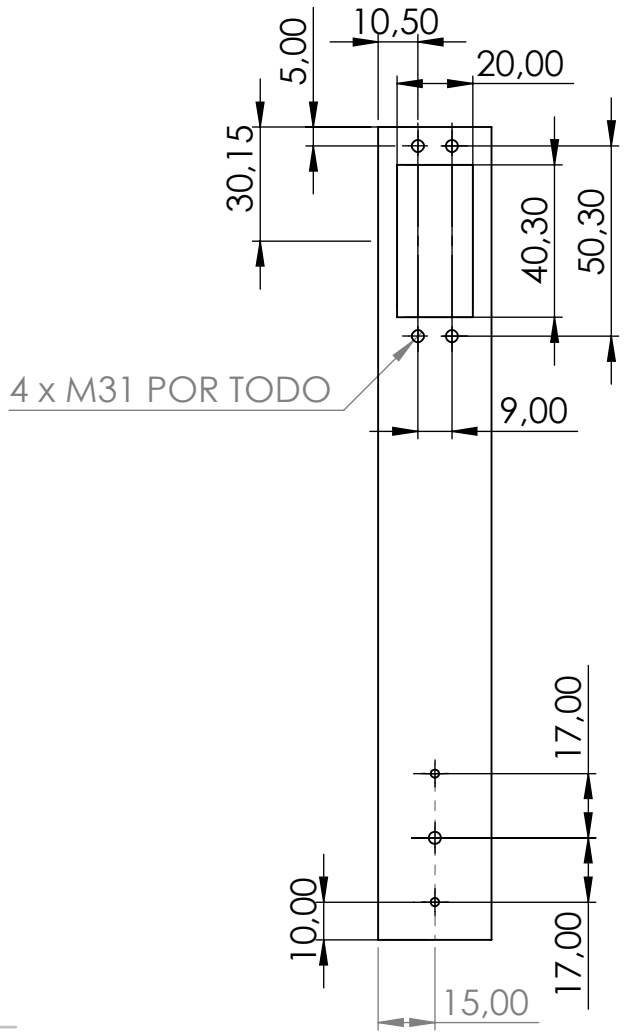


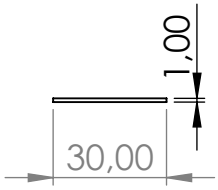


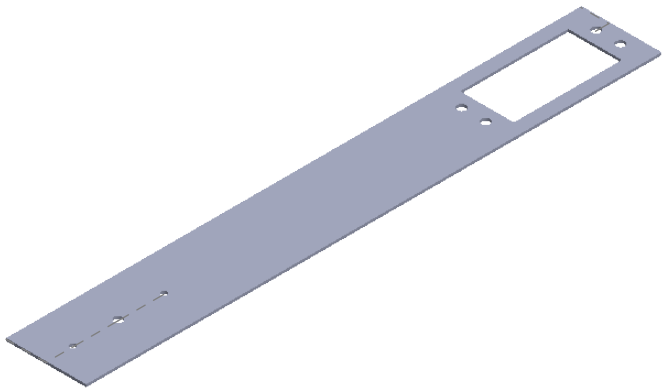




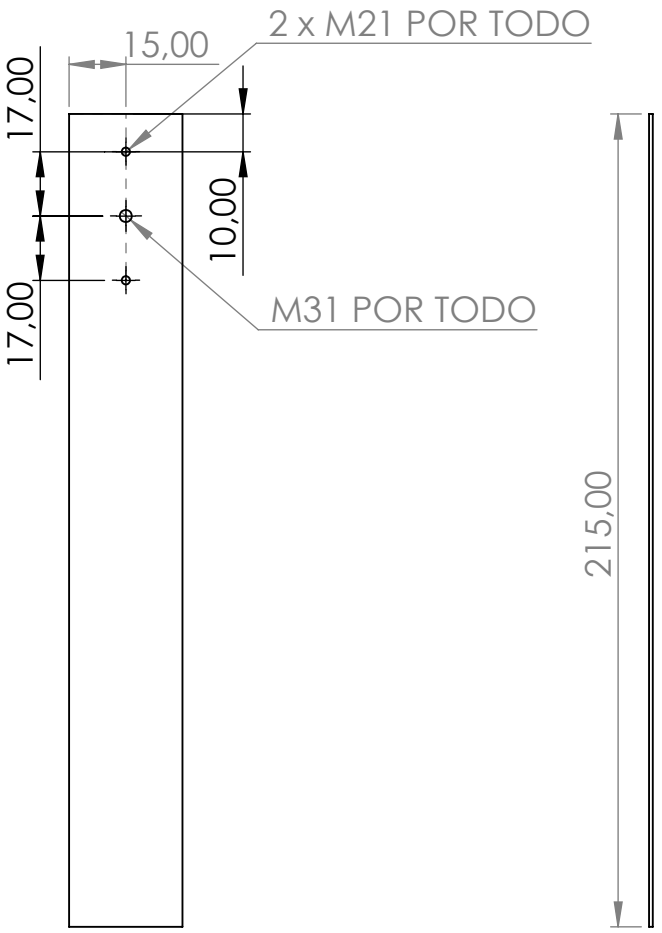
TFG EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	SI NO SE INDICA LO CONTRARIO LAS COTAS SE EXPRESAN EN MM	Título: Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos	
	Creado por: Vicente Ferrando Santamaría		
	Fecha de creación: 16/08/2020		
	ESCALA: 3:4	Plano: Base Motor 2	Nº plano: 7 / A4
	NO CAMBIE LA ESCALA		

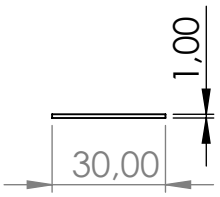


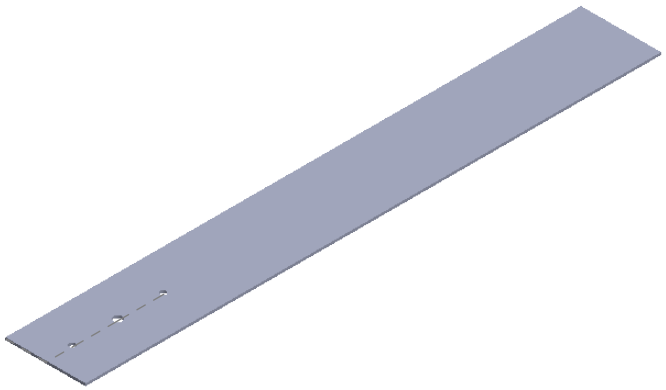




TFG EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	SI NO SE INDICA LO CONTRARIO LAS COTAS SE EXPRESAN EN MM	Título: Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos	
	Creado por: Vicente Ferrando Santamaría		
	Fecha de creación: 16/08/2020		
	ESCALA: 1:2	Plano: Eslabón 2	Nº plano: 8 / A4
	NO CAMBIE LA ESCALA		







TFG EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA	SI NO SE INDICA LO CONTRARIO LAS COTAS SE EXPRESAN EN MM	Título: Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos	
	Creado por: Vicente Ferrando Santamaría		
	Fecha de creación: 16/08/2020		
	ESCALA: 1:2	Plano: Eslabón 3	Nº plano: 9 / A4
	NO CAMBIE LA ESCALA		

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos.

III. PLIEGO DE CONDICIONES

AUTOR: Vicente Ferrando Santamaría

TUTOR: Leopoldo Armesto Ángel

Curso Académico: 2019/2020

Índice pliego de condiciones

1. Objeto

El objeto de este contrato es el de establecer los requerimientos técnicos mínimos que deben cumplirse en el diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad.

El ámbito de aplicación de este pliego de condiciones técnicas se extiende a los sistemas mecánicos y electrónicos que forman parte del producto.

El alcance de este proyecto contempla:

- Diseño y montaje de la estructura del robot para realizar los movimientos necesarios de forma holgada.
- Conexión de los elementos electrónicos para el correcto funcionamiento del brazo robótico.
- Comprobación y puesta a punto del proyecto.

2. Condiciones generales

2.1 Normativa aplicable

Real decreto 842/2002: Reglamento electrotécnico para baja tensión.

Directiva 2006/42/CE del parlamento europeo y del consejo relativa a las máquinas.

2.2 Condiciones de las piezas

Se utilizarán diferentes materiales para la construcción del brazo. Estos materiales se elegirán en función de las necesidades de cada módulo, ya que algunas partes necesitan de mayor resistencia a la flexión, otras deben ser más o menos ligeras y también, se han elegido en función a su rigidez.

2.2.1 Descripción piezas mecánicas

Las piezas más importantes del brazo robótico serán las siguientes:

- Base del robot: La base sobre la que se apoye el robot deberá ser lo suficientemente pesada y grande como para soportar las fuerzas generadas por los eslabones del robot sin que éste vuelque. Todos los elementos del robot dependerán de la estabilidad de esta base ya que todos los elementos estarán anclados aquí.
- Bases de los motores: Las piezas en las que se atornillen los motores, deberán ser lo suficientemente resistentes como para no romperse ni doblegarse por el peso de estos, aunque tampoco pueden ser pesadas ya que cuanto mayor sea el peso de antes más par motor se necesitará.

2.2.2 Control de calidad de la piezas

Se deberá comprobar la resistencia de los materiales en términos de flexión y torsión, para evitar roturas o desperfectos, en particular del segundo y tercer eslabón, cuyo material debe ser el más ligero y por tanto será el menos resistente. Se deberán comprobar las combinaciones de posiciones más críticas para estas dos piezas, centrándose en su flexión y resistencia. Otro punto crítico que se deberá tener en cuenta es el peso y las dimensiones de la base de madera, ya que debe tener el peso y dimensiones suficientes como para que el robot no vuelque cuando el momento de las fuerzas sea máximo.

2.3 Condiciones de componentes electrónicos

Ninguno de los componentes electrónicos citados a continuación ni generarán, ni conducirán, ni consumirán un voltaje peligroso para la seguridad de los usuarios que utilicen el robot.

2.3.1 Descripción componentes electrónicos

Fuente de alimentación

Se deberá alimentar el robot con un voltaje mínimo de 5V, aunque es recomendable que sea de aproximadamente 6V. En ningún caso, deberá superar los 7.2V.

Controlador de servo motores

Se utilizará un controlador de la marca witMotion. Está diseñado para controlar hasta 16 servo motores. Se alimenta con un voltaje de entre 5 y 7.2V. Se puede conectar tanto a un ordenador como a un teléfono móvil para enviar instrucciones a los motores. En el caso del ordenador, el propio fabricante proporciona un software para controlar la posición y la velocidad a través de conexión USB. También se puede conectar mediante puerto serie (necesidad de compra de un adaptador). Para realizar la conexión con el teléfono móvil se le puede acoplar un módulo bluetooth y desde la aplicación que proporcionan, se envían las instrucciones.

Motores

Se utilizarán 3 servo motores eléctricos, la fuerza mínima que deberán de tener es de 11kg/cm, y el voltaje de trabajo deberá estar comprendido entre 4.5 y 7. Será recomendable que los engranajes de los motores sean metálicos para alargar la vida útil de estos.

Convertidor USB-Puerto serie

Se utilizará, si es necesario, un dispositivo capaz de enviar información usando el método del puerto serie a un ordenador utilizando el puerto USB de este.

2.3.2 Control de calidad de los componentes electrónicos

En el ámbito del Real decreto 842/2002, el Reglamento Electrotécnico para Baja Tensión (REBT), las normas que se citan no son obligatorias en ningún caso, sino que otorgan "presunción de conformidad".

Directiva 2006/42/CE del parlamento europeo y del consejo relativa a las máquinas. Si se quisiera comercializar y poner en servicio, debe ir acompañada del certificado CE de conformidad y ostentar el marcado CE.

3. Condiciones de ejecución

3.1 Componentes mecánicos

3.1.1 Descripción del proceso de ejecución

El montaje y ensamblaje de los componentes mecánicos del robot se realizará mediante uniones mecánicas, la mayoría de ellas con tornillos.

Durante el proceso de montaje y ensamblaje no se realizará ningún proceso contaminante ni perjudicial para el medioambiente. Tampoco se utilizará ningún producto digno de mencionar que pueda afectar negativamente a la salud o al medio ambiente.

Para obtener un diseño óptimo en cuanto a robustez y funcionalidad se refiere, será necesario seguir las instrucciones.

3.1.2 Control de calidad

Las piezas inferiores del brazo robótico, es decir, las que conforman la base, deberán estar bien sujetas unas con otras y ser estables, ya que de ello dependerá la estabilidad del brazo robótico al completo.

Para obtener un movimiento fluido del movimiento se deberán anclar los motores y dejarlos bien sujetos a la estructura del robot al igual que los acoples de transmisión para evitar problemas en los ejes de transmisión de los motores.

3.2 Componentes electrónicos

3.2.1 Descripción del proceso de ejecución

Se deberán preparar y conectar los distintos componentes electrónicos, y para ello se deberá seguir una serie de pautas para mayor seguridad:

En primer lugar, se utilizarán guantes electroestáticos siempre que se tenga que manipular cualquier componente electrónico. También se deberá desconectar la alimentación mientras se esté manipulando la parte electrónica. Como se está trabajando con componentes electrónicos, es conveniente tener controladas la temperatura y la humedad para asegurar el correcto funcionamiento de estos.

3.2.2 Control de calidad

La conexión del adaptador USB-puerto serie con el controlador puede dar lugar a error. Siguiendo el mapa de conexiones de la hoja de especificaciones, se conectarán 4 pines. La tierra del controlador se conectará a la tierra del adaptador, al igual que el pin 3V3 del adaptador se conectará directamente al conector 3V3 del controlador. Donde puede generarse confusión es en las últimas dos conexiones, ya que el pin RXD del adaptador se conectará al conector TX de controlador, y el pin TXD con el RX.

Se deberá comprobar que los cables, sobre todo del tercer motor, puedan llegar al controlador sin que se genere demasiada tensión en su elongación. También se deberá revisar que el cableado de los motores no se enrede durante el funcionamiento del robot.

4. Prueba de servicio

Una vez montado el robot y realizadas todas las conexiones, se harán una serie de pruebas para comprobar el correcto funcionamiento del robot.

4.1 Funcionamiento mecánico

Se realizarán diversas pruebas de servicio para comprobar las características mecánicas del diseño.

En primer lugar, se deberá comprobar que la base principal, las patas y la base del primer motor están bien sujetos unos a otros, ya que de esto dependerá en gran medida la estabilidad del robot. Si las piezas no están correctamente conectadas, se generaría un riesgo de volcado del brazo.

En segundo lugar, se deberá comprobar que todos los acoples de los motores están bien sujetos al eje de transmisión y a su eslabón correspondiente, ya que, si no fuera así, podrían causarse desperfectos en los ejes de transmisión de los motores.

Por último, deberemos asegurarnos de que los tornillos sobre los que apoya el primer eslabón estén siempre ajustados a este No solo se debe comprobar en el montaje inicial, sino cada vez que se utilizará, de esta manera evitaremos desperfectos en el eje de transmisión del primer motor por un balanceo excesivo del eslabón.

4.2 Funcionamiento electrónico

A continuación, se comentarán las pruebas que se deberán realizar a los componentes electrónicos para asegurar la puesta en funcionamiento del robot.

En primer lugar, se comprobará que el controlador esté alimentado correctamente desde la fuente de alimentación. Al conectar la alimentación al controlador, si todo está correcto, se encenderá un led de color azul, en cualquier otro caso, deberá revisarse la conexión y el estado de los cables, ya que posiblemente no le llegue corriente a la placa.

También se deberá revisar el estado del cableado para evitar cables pelados o defectuosos.

Es importante revisar que los motores estén conectados en los canales correctos. Además, los fabricantes suelen proporcionar un software de control de motores donde se pueden mover los motores manualmente y comprobar que funcionan correctamente y que alcanzan todas las posiciones posibles.

Por último, se comprobará la correcta conexión del controlador con el ordenador mediante el puerto serie, para ello, desde el ordenador accederemos a la configuración de 'Administración de dispositivos', y nos aseguraremos que el software del ordenador lo ha detectado como un puerto COM.

5. Condiciones de entrega

Toda la documentación generada durante este proyecto quedará a la disposición de quien lo desee. Tanto la memoria, como los planos, como el pliego de condiciones estará disponible. Los archivos generados en SolidWorks del diseño del robot se subirán a la plataforma GrabCAD. Al tratarse de un proyecto con fines académicos, cualquier actualización que se realice, se plasmará en la documentación disponible para el profesorado que quiera utilizar este brazo robótico, y también para cualquiera que quiera aprender.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos.

IV. PRESUPUESTO

AUTOR: Vicente Ferrando Santamaría

TUTOR: Leopoldo Armesto Ángel

Curso Académico: 2019/2020

Índice presupuesto

1. Sobre el presupuesto

En el presente presupuesto se especifica el coste total del brazo robótico construido, no tiene en cuenta los diferentes prototipos generados durante el desarrollo del proyecto, tan solo el coste del diseño final.

El presupuesto se divide en 3 partes bien diferenciadas: los materiales, donde se encuentran los materiales mecánicos y electrónicos utilizados en la construcción del brazo robótico; la mano de obra, donde se reflejan las horas dedicadas al proyecto, separando las horas de trabajo de cada módulo (diseño, construcción, implementación,pruebas…); por último, también se reflejan los costes directos complementarios, los cuales no se pueden añadir a la descomposición del precio ya que son difíciles de cuantificar.

Este presupuesto debe ir adjunto al resto de documentos del proyecto (memoria, pliego de condiciones y planos) para que sea vinculante.

2. Materiales

Los costes de los materiales pueden cambiar ya que éstos no son en ningún caso precios fijos y el presupuesto puede variar en función de estos cambios. Si el precio de algún componente se modificara de manera excesiva, se podría sustituir por otro que tuviera sus mismas características o parecidas con un precio más reducido.

Para la elección de los materiales se ha optado por un perfil reducido de los costes para mantener el presupuesto lo más reducido posible. Para la construcción del brazo robótico tan solo se deberá tener en cuenta este apartado, ya que los alumnos que quieran construirlo tan solo deberán tener en cuenta el coste de los materiales.

Materiales							
Ref	**Ud**	**Descripción**	**Fabricante**	**Ref. fabricante**	**Cantidad**	**Precio**	**Total**
m1	ud	Bandeja de madera circular	Giftdecor	B07YSFY3RN	1	3,99 €	3,99 €
m2	ud	Placa de aluminio 150x250mm	Varios		0,344	14,45 €	4,97 €
m3	g.	Placa de aluminio con relleno de poliestireno 150x250mn	Varios		0,52	28,86 €	15,01 €
m4	ud	Triangulos de madera	Varios		3	0,30 €	0,90 €
m5	ud	Servo motor MG996R	AZ Delivery	B07PXDKLGH	3	7,66 €	22,98 €
m6	ud	Controlador de servo motores	WitMotion	B06Y3ZV4B8	1	22,59 €	22,59 €
m7	ud	Adaptador USB-puerto serie	WitMotion	B07VDSJPKX	1	15,42 €	15,42 €
m8	ud	Porta pilas	GTIWUNG	B07WJ3HFSP	1	1,50 €	1,50 €
m9	ud	Pack 4 pilas AA	Energizer	B00W5C13Q6	1	1,59 €	1,59 €
m10	ud	40pcs 20cm 2,54 mm Dupont Macho a Hembra	Neuftech	B00M9XOCV4	1	3,99	3,99 €
m11	ud	Tornillos, tuercas, escuadras (±10% error)	Varios		1	10,50 €	10,50 €
						SUBTOTAL	**103,44 €**

3. Mano de obra

Para el cálculo del presupuesto de la mano de obra, se ha tenido en cuenta que todo el proyecto ha sido realizado por un ingeniero técnico. El presupuesto de la mano de obra se ha dividido en 6 subapartados, los cuales se describen a continuación:

Mano de obra				
Ud	**Descripción**	**Cantidad**	**Precio**	**Total**
h	MOOE 8a Ingeniero Técnico	215	22,57 €/h	4.853 €
	1. Preproyecto - Bocetos y preparación de material	10	22,57 €/h	226 €
	2. Diseño y ensamblaje de piezas con software CAD 3D	32	22,57 €/h	722 €
	3. Construcción del robot	38	22,57 €/h	858 €
	4. Configuración de funcionamiento en software de simulación	43	22,57 €/h	971 €
	5. Desarrollo código de funcionamiento en simulación	77	22,57 €/h	1.738 €
	6. Pruebas de servicio	15	22,57 €/h	339 €
			SUBTOTAL	**4.853 €**

4. Costes directos complementarios

Los costes de este apartado corresponden a todos aquellos gastos generados durante el proyecto que no pueden ser cuantificados. En este caso, se han calculado aplicando un 15% a la suma de los costes de mano de obra y de los materiales.

Costes directos complementarios				
Ud	**Descripción**	**Cantidad**	**Precio**	**Total**
%	Medios auxiliares sobre costos directos	15%	4.955,99 €	743,40 €
			SUBTOTAL	**743,40 €**

5. Resumen

A continuación, se resumirán los costes totales del proyecto teniendo en cuenta los 3 apartados anteriores:

Resumen		
Ud	**Descripción**	**Total**
€	Materiales	103,44 €
€	Mano de obra	4.853 €
€	Costes directos complementarios	743,40 €
	TOTAL	**5.699,39 €**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño, montaje, implementación y simulación de un brazo robótico con 3 grados de libertad con fines educativos.

V. ANEXO

AUTOR: Vicente Ferrando Santamaría

TUTOR: Leopoldo Armesto Ángel

Curso Académico: 2019/2020

Índice del anexo 1. Código

Índice de figuras del anexo 1:

1. Código para la generación del espacio de trabajo:

1.1 CoppeliaSim:

```
function jointSweep(Q,i,N,q)
    if (i<N) then
        if (steps[i]>0) then
            for ii=0,steps[i],1 do
                qi=qLim[i][2]*ii/steps[i]+qLim[i][1]
                if (cyclic[i]) then
                    qi=2*math.pi*ii/steps[i]-math.pi
                end
                q[i]=qi
                Q=jointSweep(Q,i+1,N,q)
            end
        else
            Q=jointSweep(Q,i+1,N,q)
        end
    else
        if (steps[i]>0) then
            for ii=0,steps[i],1 do
                qi=qLim[i][2]*ii/steps[i]+qLim[i][1]
                if (cyclic[i]) then
                    qi=2*math.pi*ii/steps[i]-math.pi
                end
                q[i]=qi
                qq={}
                for j=1,N,1 do
                    qq[j]=q[j]
                end
                table.insert(Q,qq)
            end
        else
            qq={}
            for j=1,N,1 do
                qq[j]=q[j]
            end
            table.insert(Q,qq)
        end
    end
    return Q
end

function sysCall_init()

    file = io.open('robot_workspace.xyz','w')
    io.output(file)
    jointHandles={-1,-1,-1}
    cyclic = {-1,-1,-1}
    qLim={}
    N=3
    Qwrist={}

    for i=1,N,1 do
        jointHandles[i] = sim.getObjectHandle('Rev_Joint_M'..i)
        sim.setJointPosition(jointHandles[i],0)
        cyclic[i],qLim[i] = sim.getJointInterval(jointHandles[i])
    end
    wrist = sim.getObjectHandle('tipRobot')
    coll_check0 = sim.getCollisionHandle('RobotCollision')
```

Fig. 1 Código Lua para espacio de trabajo. Parte 1. Fuente propia.

```
    --coll_check1 = sim.getCollisionHandle('BrazoCollision')
    steps={0,100,100}
    q={0,0,0}
    Qwrist = jointSweep(Qwrist,2,3,q)
    maxPositions = #Qwrist
    print(maxPositions)
    counterWrist = 1

end
function sysCall_actuation()
    -- Put your main ACTUATION code here
    if (counterWrist<maxPositions) then
        for j=1,3,1 do
            sim.setJointPosition(jointHandles[j],Qwrist[counterWrist][j])
        end

        c0 = sim.readCollision(coll_check0)
        --c1 = sim.readCollision(coll_check1)

        if ( (c0==0) ) then
            position=sim.getObjectPosition(wrist,-1)
            io.write(table.concat(position,"\t"))
            io.write("\t")
            io.write(table.concat(Qwrist[counterWrist],"\t"))
            io.write("\n")
        end
        counterWrist = counterWrist+1
    else
    sim.stopSimulation()
    end
end

function sysCall_sensing()
    -- Put your main SENSING code here
end

function sysCall_cleanup()
    -- Put some restoration code here
    io.close(file)
end
```

Fig. 2 Código Lua para espacio de trabajo. Parte 2. Fuente propia.

1.2 Matlab:

```
XYZ=load('robot_workspace.xyz');
x=XYZ(:,1);
y=XYZ(:,3);
shp=alphaShape(x,y);
shp.Alpha=0.01;
[bf,xz]=boundaryFacets(shp);
figure();
plot(xz(:,1),xz(:,2));
title('Workspace 2D')
T = delaunay(xz(:,1),xz(:,2));
tri=triangulation(T,xz(:,1),xz(:,2));
stlwrite(tri,'brazo_workspace.stl');
```

Fig. 3 Código Matlab para espacio de trabajo. Fuente propia

2. Código principal de CoppeliaSim:

```
function onXChange(uiHandle, id, newValue)
    mov_path = false
    px = ((newValue*0.55)/200) + 0.1439
    position[1]=px
    sim.setObjectPosition(ref_point,-1,position)
end

function onYChange(uiHandle, id, newValue)

    mov_path = false
    py = ((newValue*0.6)/200) + 3.2926e-02
    position[2]=py
    sim.setObjectPosition(ref_point,-1,position)
end

function onZChange(uiHandle, id, newValue)

    mov_path = false
    pz = (0.49/100)*newValue + 0.4944
    position[3]=pz
    sim.setObjectPosition(ref_point,-1,position)
end

function IrAPunto()
    sim.setPathTargetNominalVelocity(path,0.1)
    mov_path = true
end


function DEC_HEX(IN)
    local B,K,OUT,I,D=16,"0123456789ABCDEF","",0
    while IN>0 do
        I=I+1
        IN,D=math.floor(IN/B),math.mod(IN,B)+1
        OUT=string.sub(K,D,D)..OUT
        OUT_String = tostring(OUT)
    end
    return OUT_String
end

function serialPort()

    posM1 = sim.getJointPosition(m1)
    posM2 = math.pi - sim.getJointPosition(m2)
    posM3 = sim.getJointPosition(m3)
    posM3_inv = posM3

    convM1 = 500 + (2000/math.pi)*(posM1)
    convM2 = 500 + (2000/math.pi)*(posM2)
    convM3 = 500 + (2000/math.pi)*(posM3_inv)

    --Comunicacion del motor 1
    if(string.len(DEC_HEX(convM1)) == 3) then
        dataL_M1 = string.sub(DEC_HEX(convM1), 2, 3)
        dataH_M1 = string.sub(DEC_HEX(convM1), 1, 1)
    end
```

Fig. 4 Código de programa principal en Lua. Parte 1. Fuente propia

```
57
58      if(string.len(DEC_HEX(convM1)) == 4) then
59          dataL_M1 = string.sub(DEC_HEX(convM1), 3, 4)
60          dataH_M1 = string.sub(DEC_HEX(convM1), 1, 2)
61      end
62
63      intL_M1 = tonumber(dataL_M1,16)
64      intH_M1 = tonumber(dataH_M1,16)
65      sendM1 = sim.serialSend(port,string.char(255)..string.char(2)..
66      string.char(0)..string.char(intL_M1)..string.char(intH_M1))
67
68      --Comunicacion del motor 2
69      if(string.len(DEC_HEX(convM2)) == 3) then
70          dataL_M2 = string.sub(DEC_HEX(convM2), 2, 3)
71          dataH_M2 = string.sub(DEC_HEX(convM2), 1, 1)
72      end
73
74      if(string.len(DEC_HEX(convM2)) == 4) then
75          dataL_M2 = string.sub(DEC_HEX(convM2), 3, 4)
76          dataH_M2 = string.sub(DEC_HEX(convM2), 1, 2)
77      end
78
79      intL_M2 = tonumber(dataL_M2,16)
80      intH_M2 = tonumber(dataH_M2,16)
81      sendM2 = sim.serialSend(port,string.char(255)..string.char(2)..
82      string.char(3)..string.char(intL_M2)..string.char(intH_M2))
83
84      --Comunicacion del motor 3
85      if(string.len(DEC_HEX(convM3)) == 3) then
86          dataL_M3 = string.sub(DEC_HEX(convM3), 2, 3)
87          dataH_M3 = string.sub(DEC_HEX(convM3), 1, 1)
88      end
89
90      if(string.len(DEC_HEX(convM3)) == 4) then
91          dataL_M3 = string.sub(DEC_HEX(convM3), 3, 4)
92          dataH_M3 = string.sub(DEC_HEX(convM3), 1, 2)
93      end
94
95      intL_M3 = tonumber(dataL_M3,16)
96      intH_M3 = tonumber(dataH_M3,16)
97      sendM3 = sim.serialSend(port,string.char(255)..string.char(2)..
98      string.char(4)..string.char(intL_M3)..string.char(intH_M3))
99  end
100
101
102 function sysCall_init()
103
104     m1 = sim.getObjectHandle("Rev_Joint_M1")
105     m2 = sim.getObjectHandle("Rev_Joint_M2")
106     m3 = sim.getObjectHandle("Rev_Joint_M3")
107     path = sim.getObjectHandle('Path')
108     targetPath = sim.getObjectHandle('TargetPath')
109     ref_point=sim.getObjectHandle('TargetMovement')
110
```

Fig. 5 Código de programa principal en Lua. Parte 2. Fuente propia

```
    position=sim.getObjectPosition(ref_point,-1)
    mov_path = false
    BaudRate = 9600
    port = sim.serialOpen("\\\\.\\COM12",BaudRate)

    ui=simUI.create('<ui enabled="true" modal="false" title="Brazo Robotico" closeable="true" layout="vbox" placement="relative" position="20,20">' ..
    '<label enabled="true" text="X"></label>' ..
    '<hslider enabled="true" minimum="-100" maximum="100" on-change="onXChange"></hslider>' ..
    '<label enabled="true" text="Y"></label>' ..
    '<hslider enabled="true" minimum="-100" maximum="100" on-change="onYChange"></hslider>' ..
    '<label enabled="true" text="Z"></label>' ..
    '<hslider enabled="true" minimum="-100" maximum="0" on-change="onZChange"></hslider>' ..
    '<button enabled="true" text="Trayectoria" on-click="IrAPunto"></button>' ..
    '</ui>')
end
```

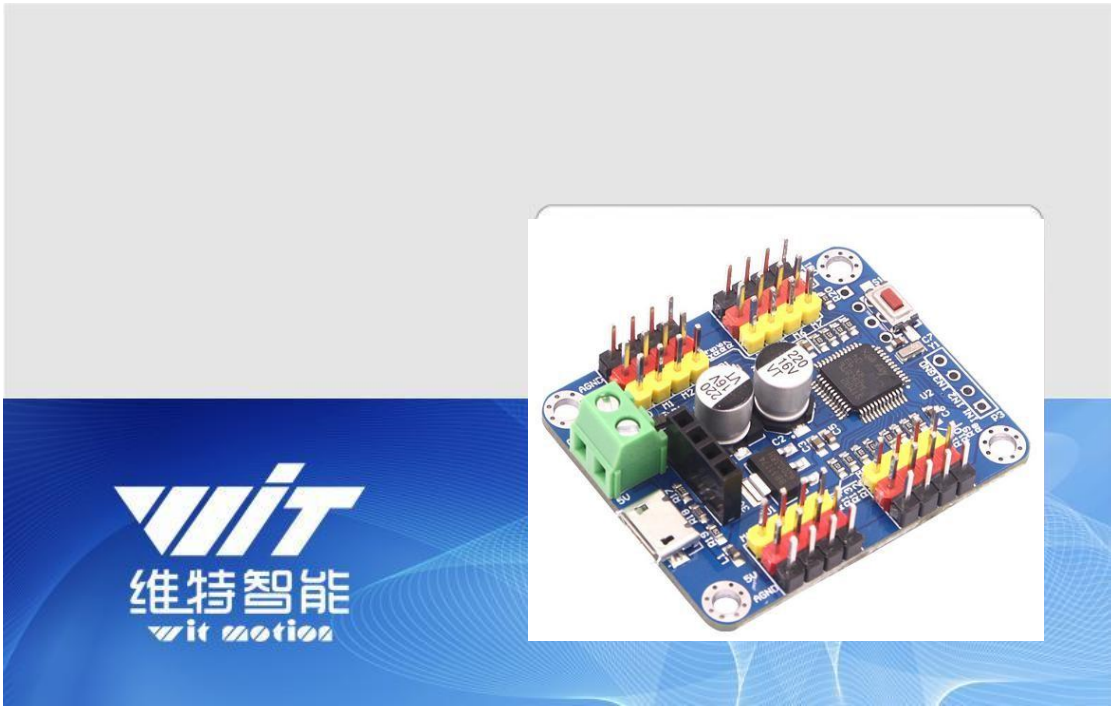
Fig. 6 Código de programa principal en Lua. Parte 3. Fuente propia

```
function sysCall_actuation()
    -- put your actuation code here
    position=sim.getObjectPosition(ref_point,-1)
    print(position)
    if (mov_path) then
        posPath = sim.getObjectPosition(targetPath,-1)
        sim.setObjectPosition(ref_point,-1,posPath)
    end

    serialPort()
   -- cinematicaInversa()

end

function sysCall_sensing()
    -- put your sensing code here
end

function sysCall_cleanup()
    -- do some clean-up here
end
```

Fig. 7 Código de programa principal en Lua. Parte 4. Fuente propia

Índice del anexo 2. Hojas de especificaciones

16 Channels Servo Board



Model： WT Servo 16

Description： Servo motor controller driver board 16 channels

Production Standard

Enterprise quality system standard: ISO9001:2016

Catalog

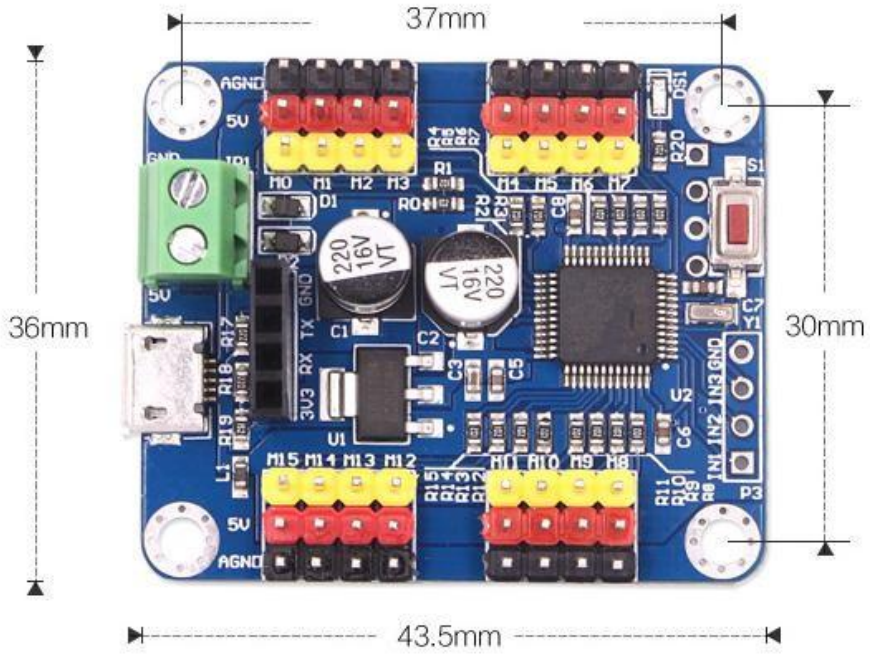
1. Features

- 16 channels servo motor control.
- USB/ TTL/ Bluetooth (optional Bluetooth module) connection.
- Action state setting, Simplify the action process.
- Online motor program editor, custom action sequences.
- Excellent PC control software, online control, offline operation.
- USB power supply (for writing program), direct plug, no need to drive.

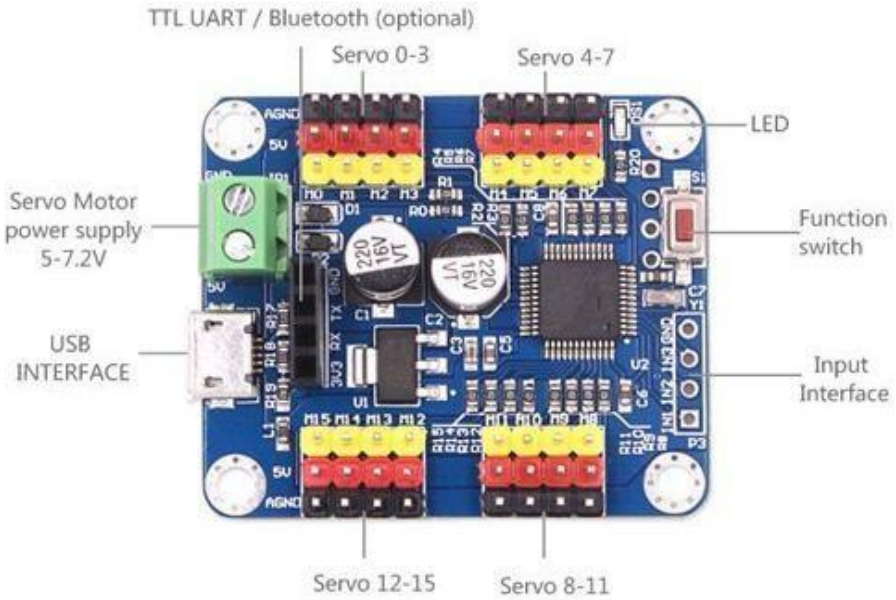
2. Product Parameters

- Master chip: STM32 series.
- Frequency: 48MHZ.
- Size: 3.5mm×36mm×12mm.
- Mounting hole position: 37×30mm, Aperture: 3mm.
- Operating temperature: -40～80℃.
- Operating voltage: USB: 5V
- TTL: 3.3V(two choose one).
- Servo motor supply: 5～7.2V. The JP1 needs to supply the servo motor separately.
- Channels: 16.
- PWM accuracy: 0.1us.
- The minimum step of the servo motor: 1us.
- Communication interface: USB/ TTL UART(optional Bluetooth).
- Baud rate: 9600Kps
- PC software: Yes.
- Number of action groups: Group 16.
- Store the number of action: 8192.

3. Product Display



4. Interface Description



5. Operating Instruction

5.1 Dual Power Supply

1. The USB port/ serial port is supplied to the servo motor control board. it is also used as the interface of PC communication. Voltage is 5V
2. The Servo Motor power port(JP1) supplies to the steering gear, the supplied voltage is 5-7.2V.

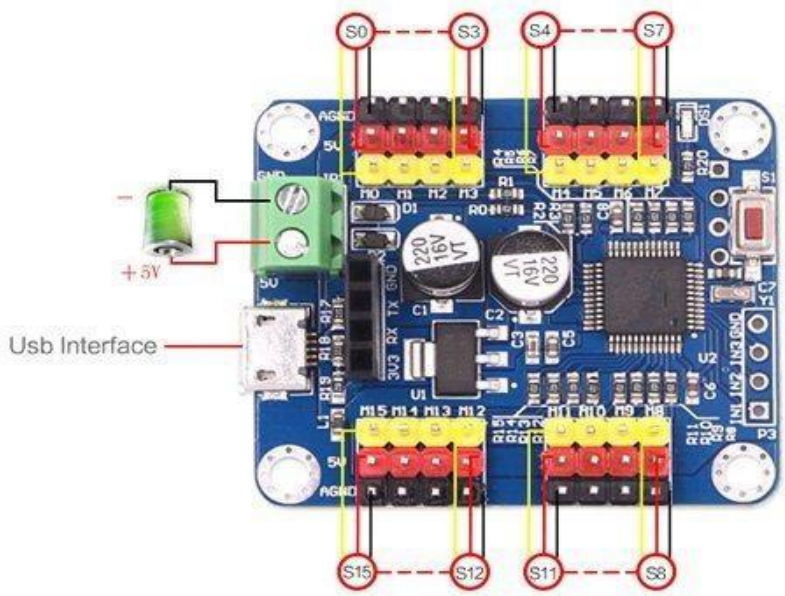
Reminder: The PC software can be programmed directly through the USB port/ serial port. If there is a connection to the steering gear. You must use the JP1 port to power the servo motor, otherwise the servo motor can not be turned. In the case of two-way power supply, the PC software operation is also correct, nor can not control the servo motor. Please measure the voltage of the JP1 port. This phenomenon is generally caused by the instability of power supply.

5.2 USB Connection

Connecting PC requires only one USB cable to control the board, connect to the PC software and test the basic function of the control panel.

5.2.1 Hardware Connection

When the USB cable is connected to the servo motor control panel, the light(DS1) of control panel is always on, which means having been connected successfully. Refer to the hardware connection diagram.

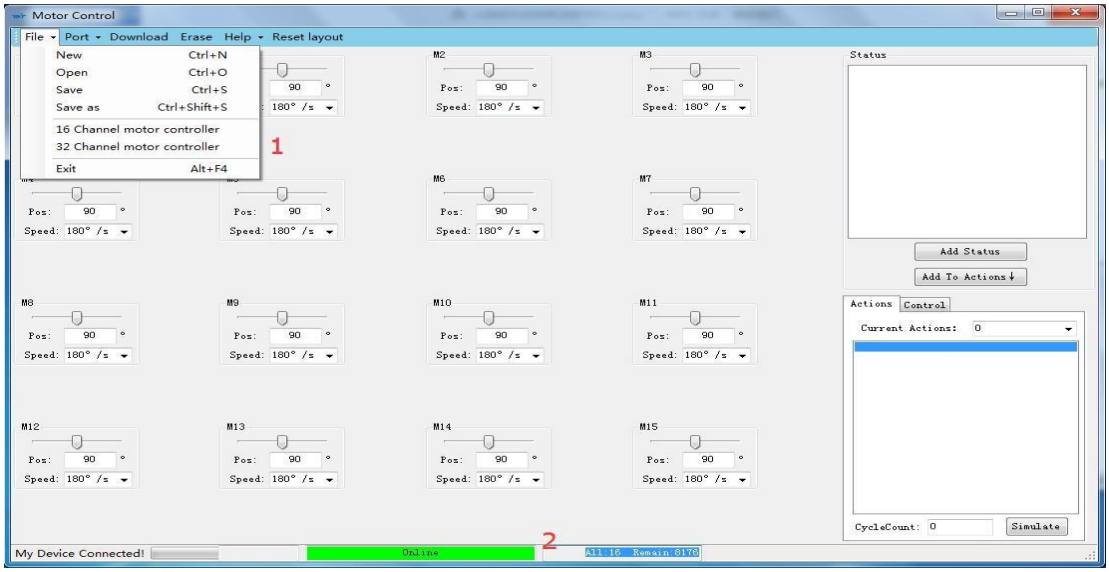


5.2.2 Software Connection

（1） Double-click on the PC software icon MotorControl.exe and open it.

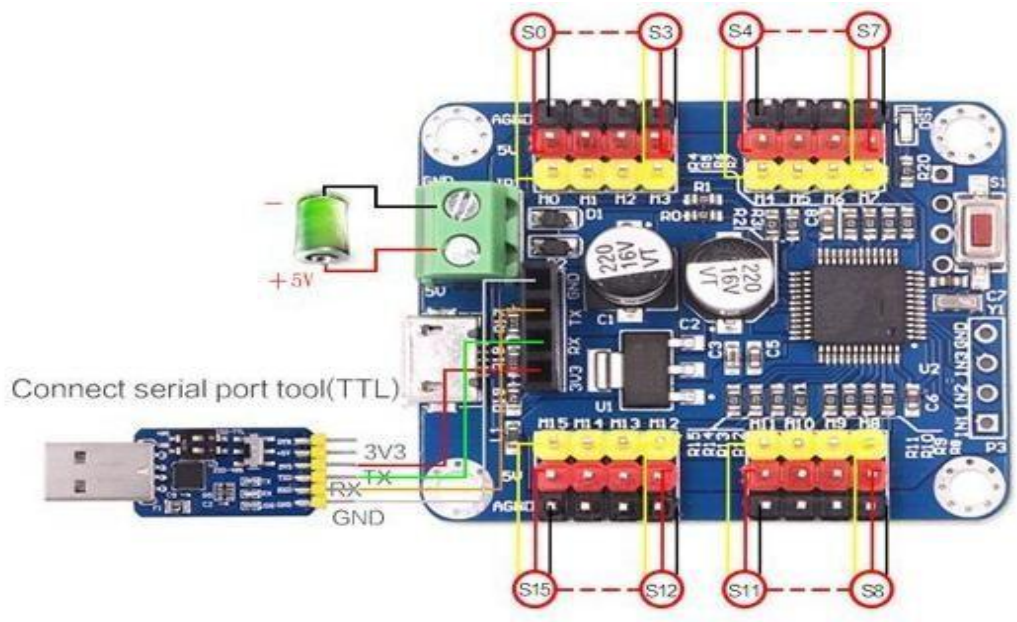
（2） Click “File ” and choose “16 Channels controller” .

When the status of the PC software connection is displayed green, that is the online state. At this time, the indicator light(DS1) of the control panel blinks slowly. If not online, the state bar is yellow.



5.3 Instruction

5.3.1 Hardware Connection



① Power the board

The supplied voltage of control panel is 3.3V

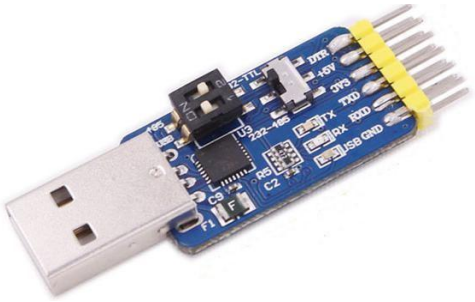
② Power the Servo Motor

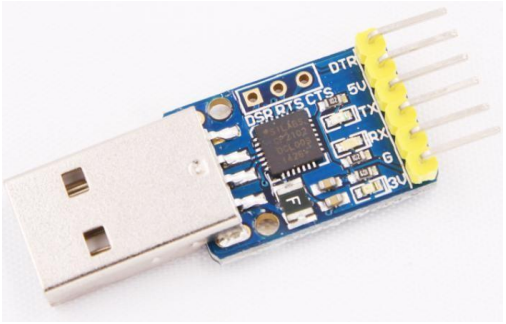
The Servo Motor power port(JP1) supplies to the steering gear, the supplied voltage is 5-7.2V.

Reminder：The voltage of control panel can not exceed 3.3V. Otherwise, the servo motor control panel will be damaged.

③

the following two module.



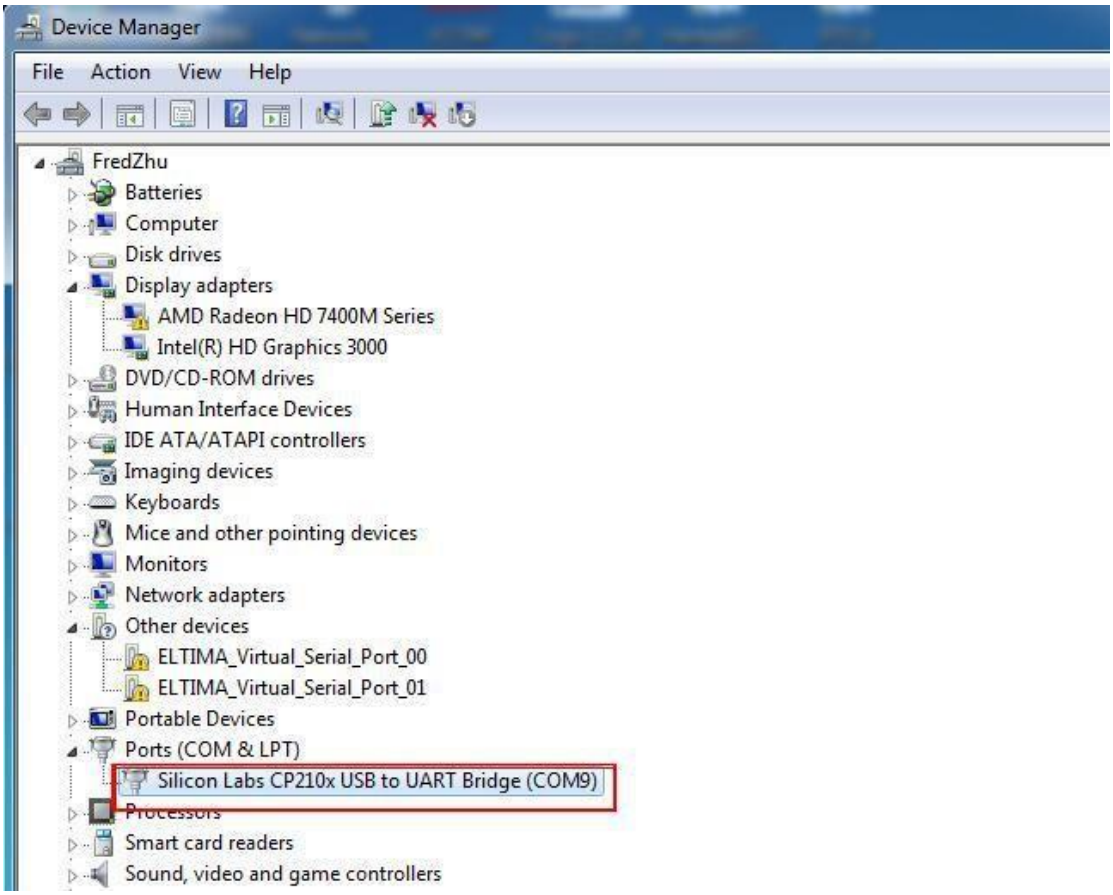


When connected to the PC software, you need a USB -TTL serial module. Recommend

The ways to connect USB module to the control panel are: 3.3V of the USB, TXD, RXD are connected to 3.3V, RX, TX, GND.

Reminder: Insert the module USB- TTL, and make sure the connection is correct, then you should install the drive first. Drive download address;

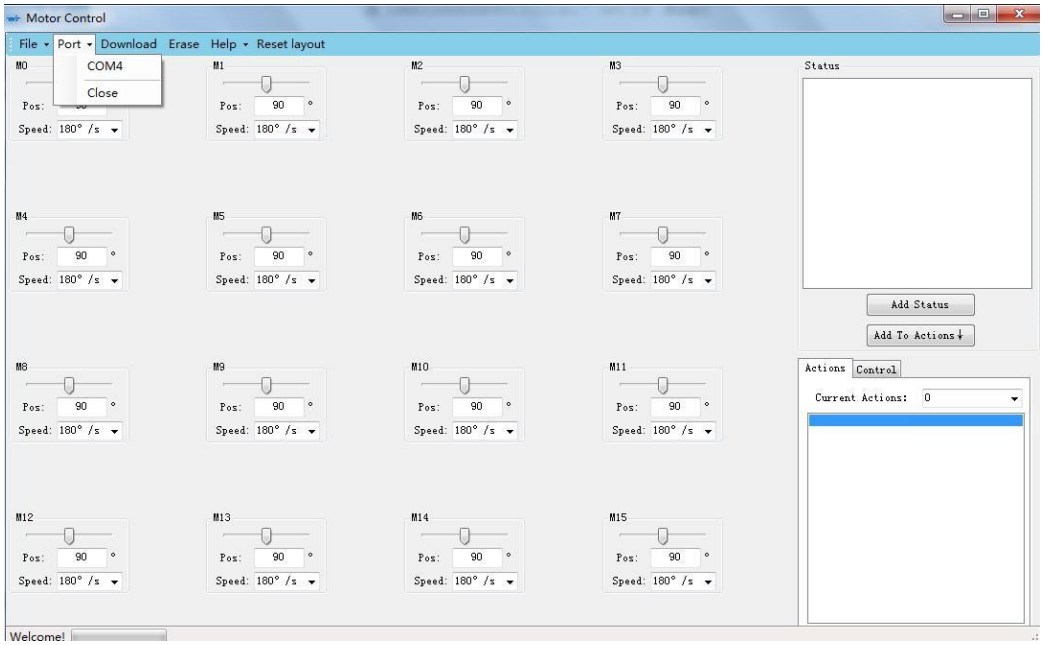
https://wiki.wit-motion.com/english/doku.php?id=communication_module_1



5.3.2 PC Software Connection

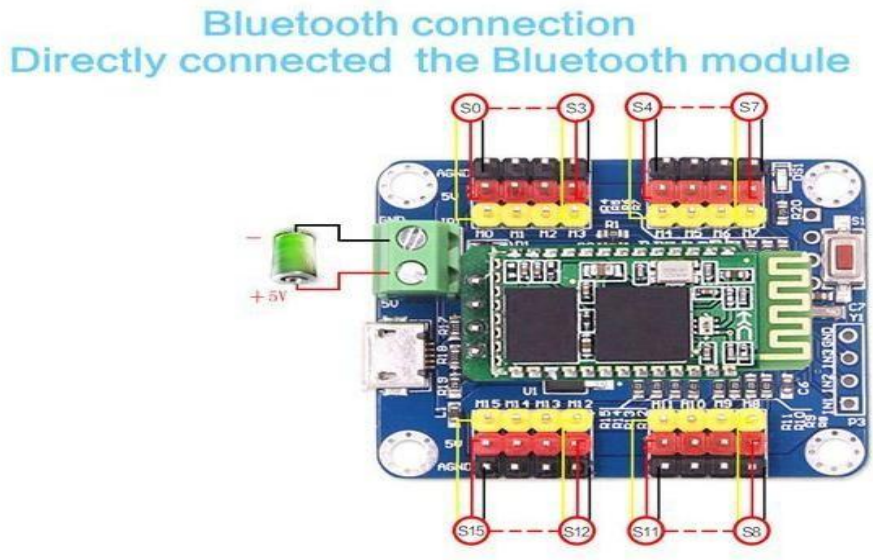
1. Click “File” and choose the servo motor control panel.
2. Click “Port” and choose the corresponding port number.

When the status of the PC software connection is displayed green,, that is the online state, at this time, the control pane indicator light(DS1) blinks slowly. If not on line, The state display column is yellow.



5.4 Bluetooth Connection（just for phone）

5.4.1 Hardware Connection



a. Power supply

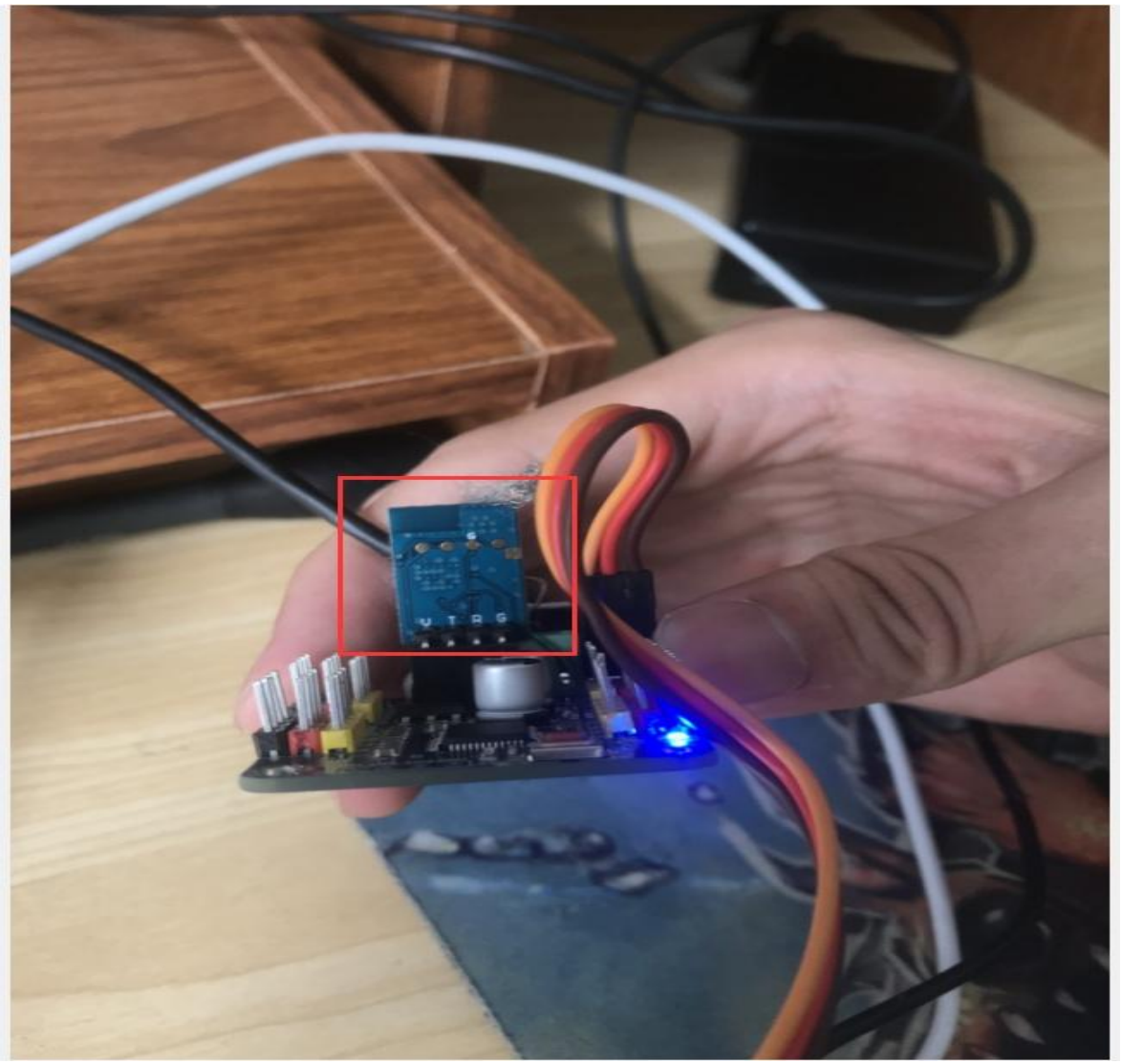
The power supply of the servo motor is 5V- 7.2V, refer to the hardware connection diagram .

b. Recommended Bluetooth module

Insert the Bluetooth module into the Bluetooth port, refer to the hardware connection diagram. c. Search the PC software for Bluetooth device and match it.

Windows system does not support for Bluetooth connection very well sometimes. Recommended a software Bluesoleil:

Use like this

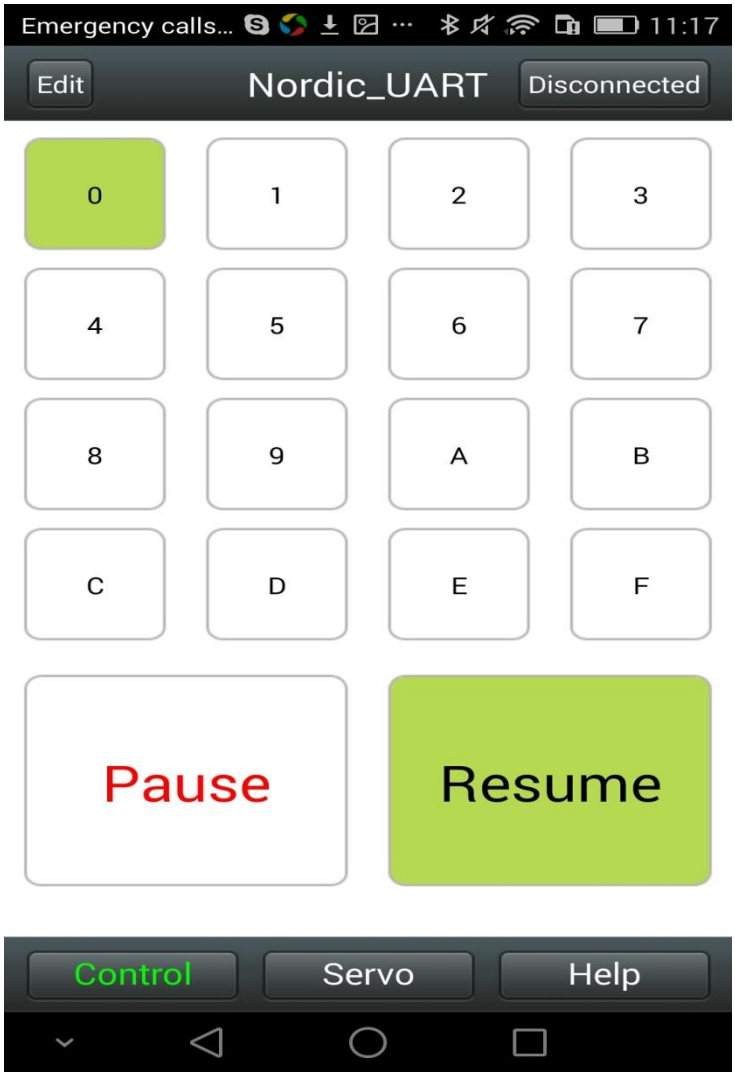


1）Open Bluetooth on phone

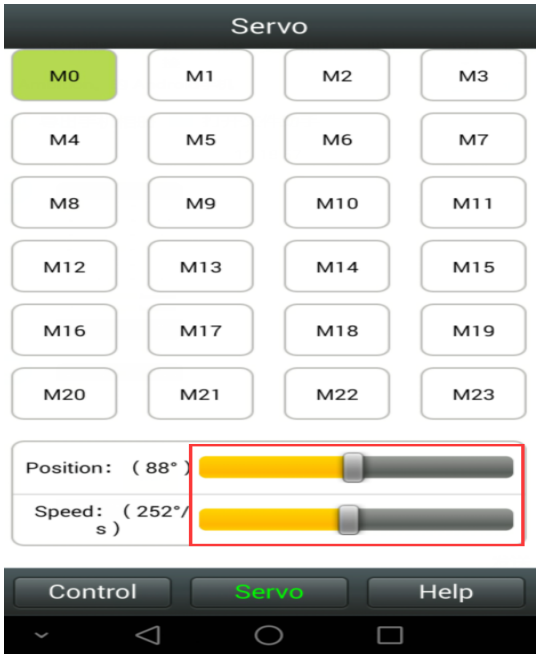
2）Then turn on the “mini IMU” APP on the phone，then click connect to find product

3）When the blue light on the Bluetooth module is lit, the connection is successful.

4）this page shows Action group which you need set on PC software, you can use action group to connect Multiple servos.

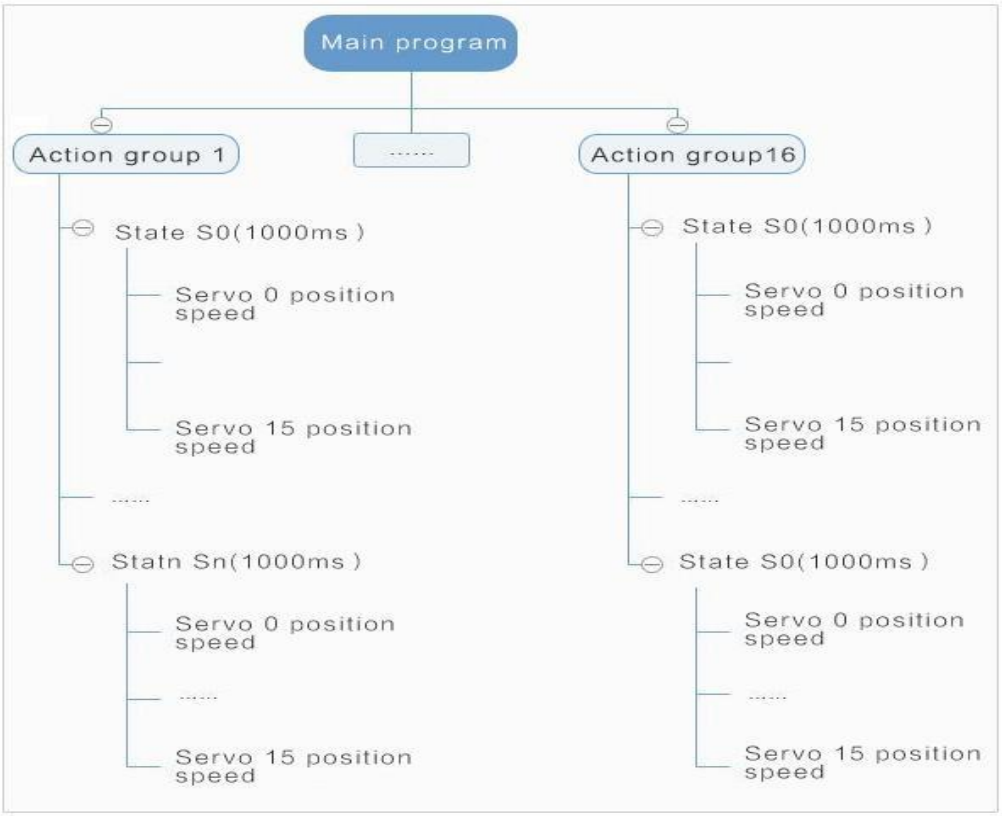


5) This page shows the name of the gusset on a single servo connection, you can click it to change the position and speed



5.5 Software Operation

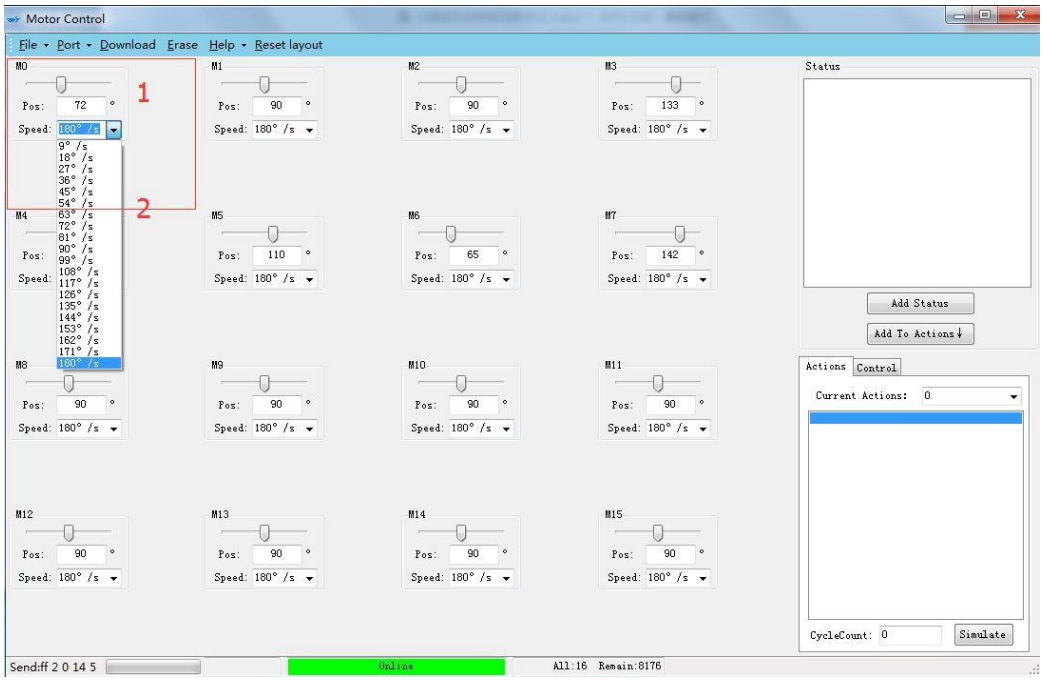
Highlights: only need to define the final state of the servo motor, without detailed calculation of the details of the servo motor operation, breaking the traditional control of the servo motor control board.

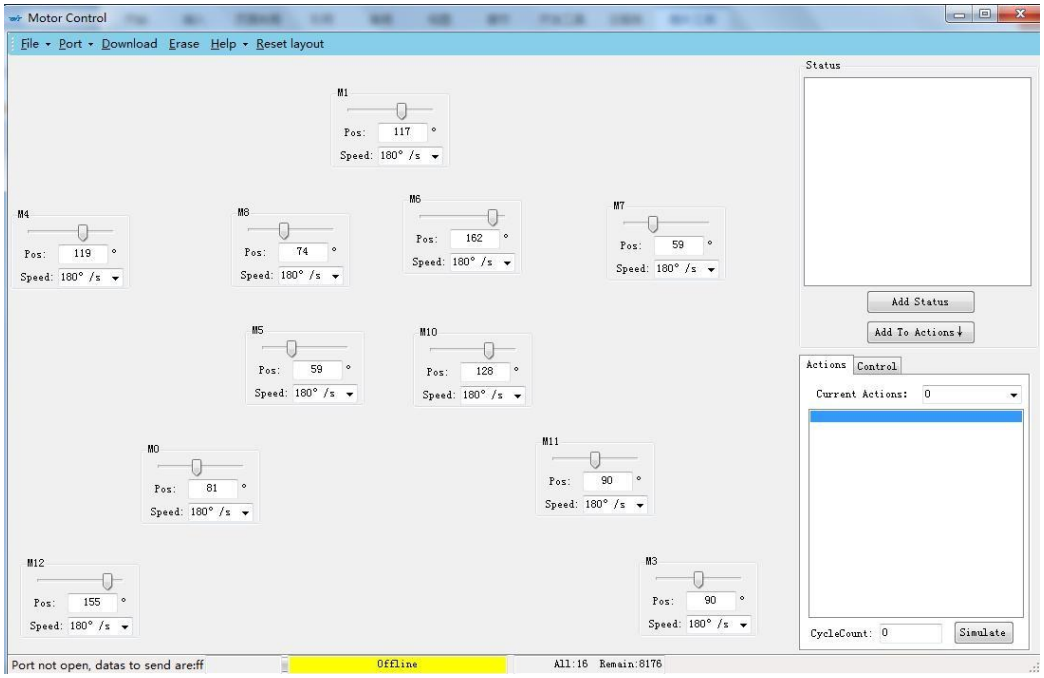


5.5.1 Control the Servo motor

（1） In the on line state, drag the progress bar of the servo motor and change the speed of the servo motor.

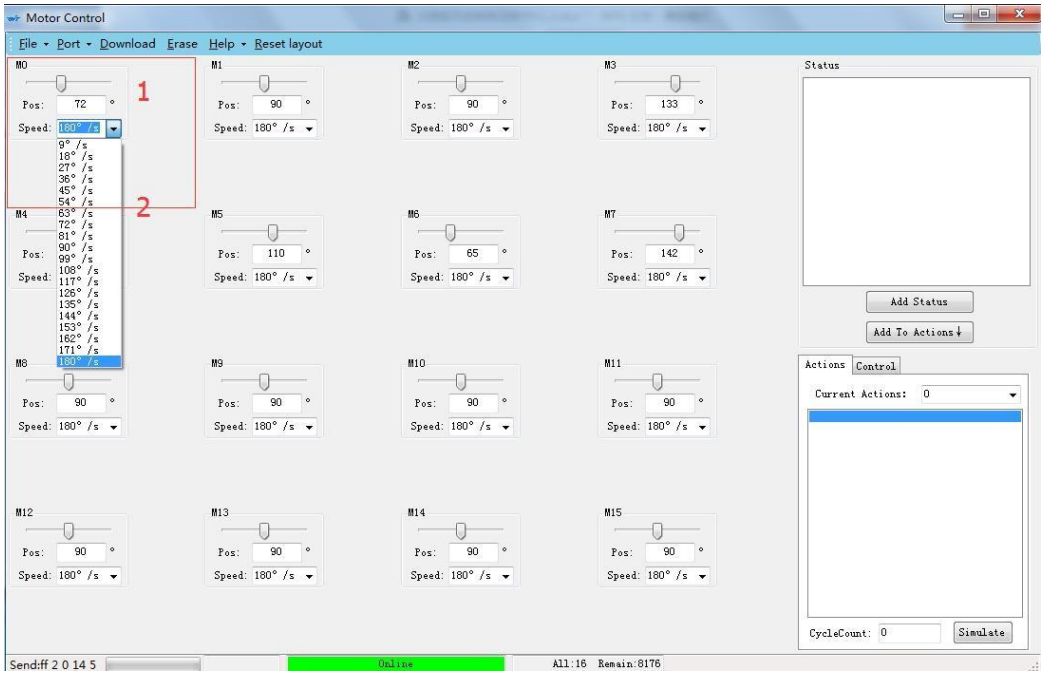
（2） Click”Speed” change the speed of the servo motor, speed is 9°/S-180 °/S optional .





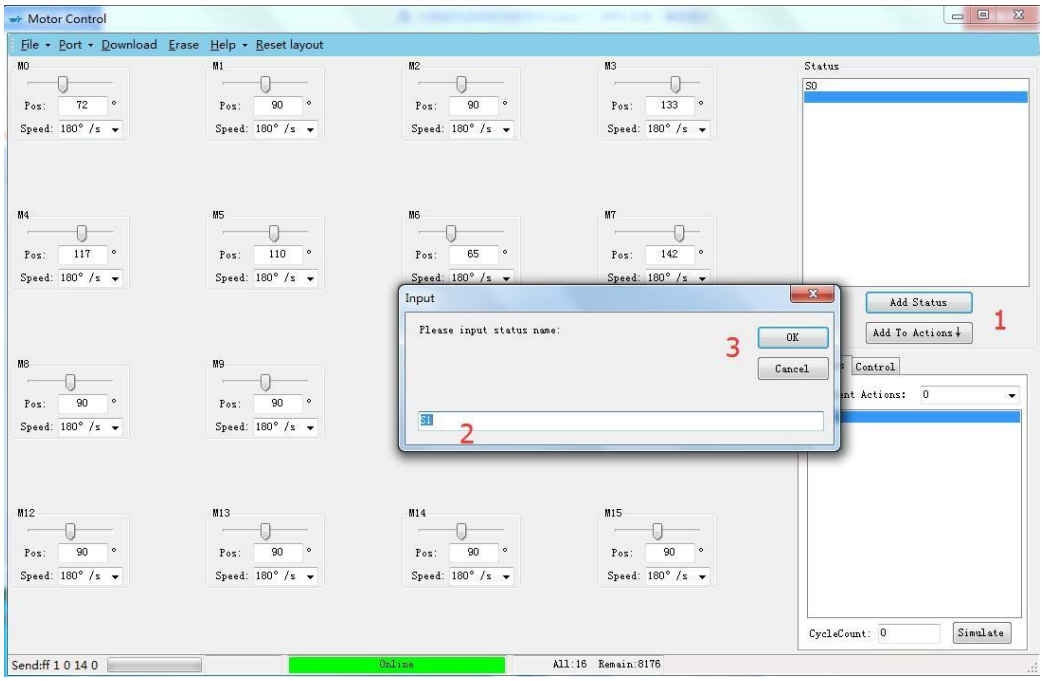
5.5.2 Add State

(1) Set the parameter of the servo motor(speed/ time)



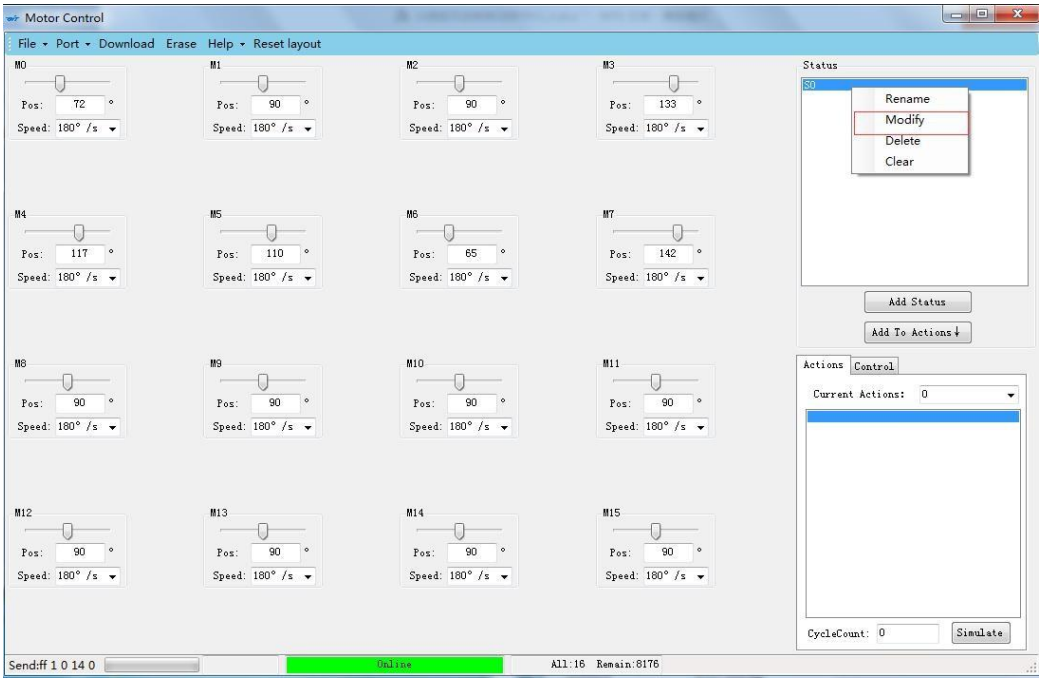
(2) Add the new state in the status, and input the name of it.

(3) Click the “OK”, the action will be added to the actions.



Repeat the steps above and you can add many different state

(4) Click the name of the action and modify the position and speed of the servo motor. Click the “Modify ” and finish it.



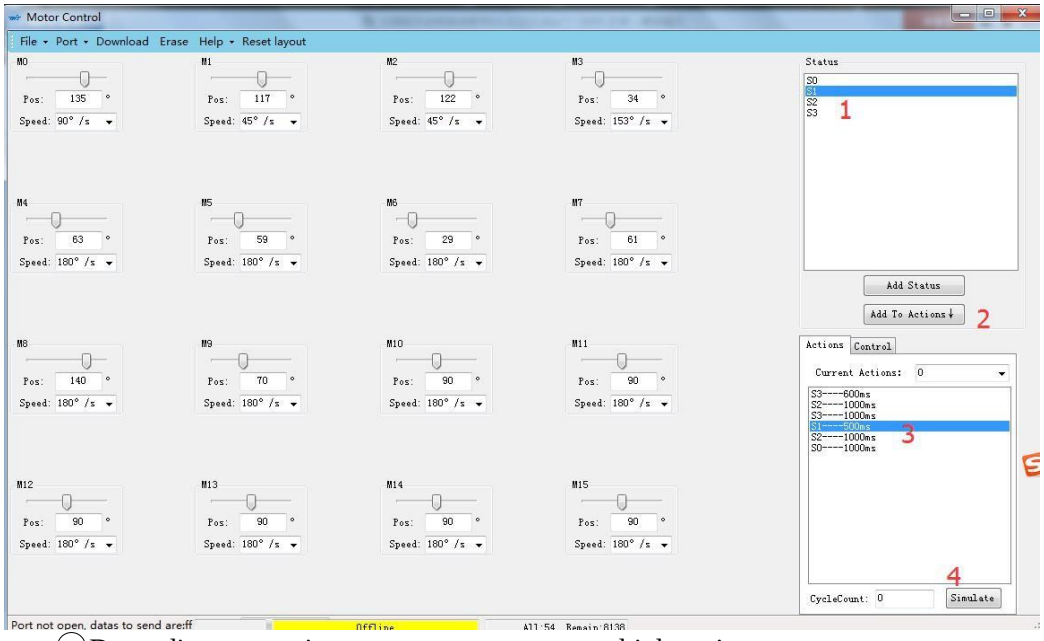
5.5.3 Set the Action Group

① Select the status name.

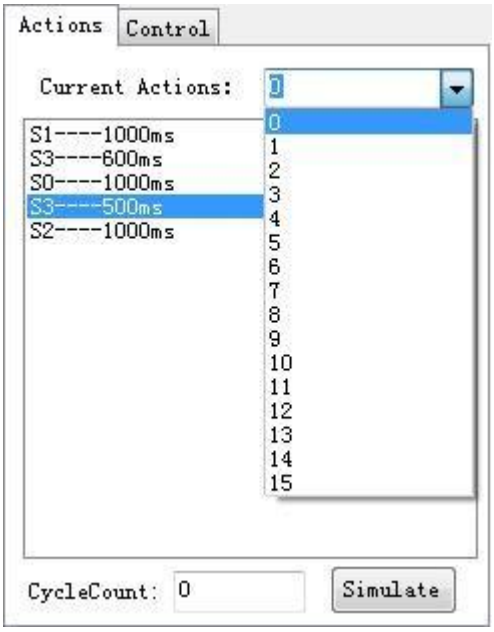
② Click the "Add to action"

③ You can find the status has been added, and set the time about the action group.

④ Add status to the action group in sequence, and click "simulate", the servo motor will do simulation operation.

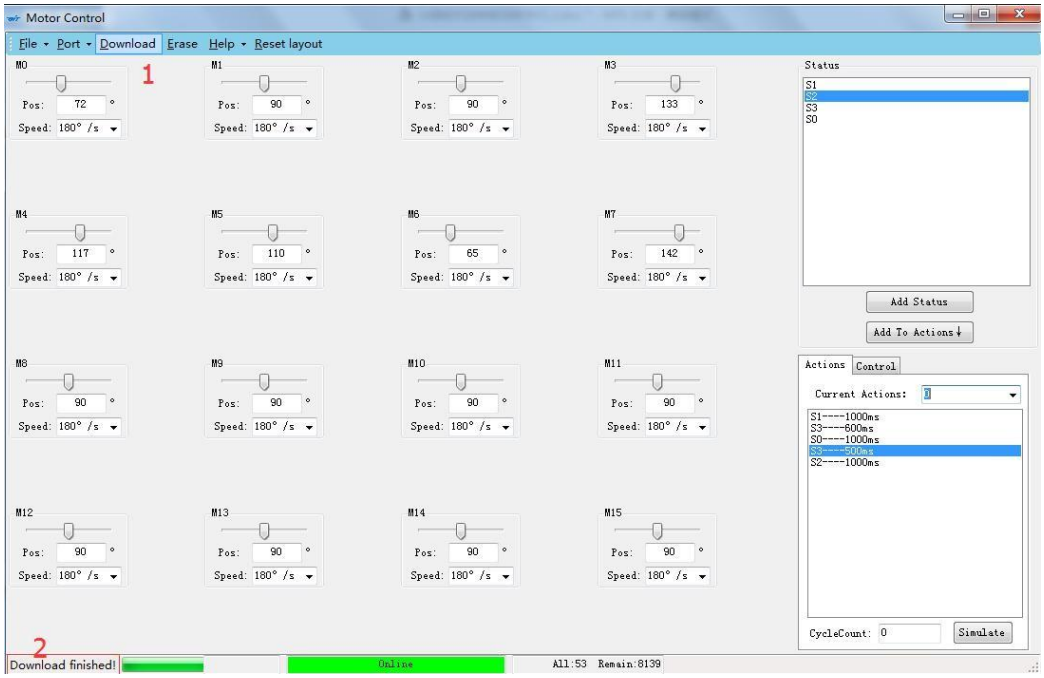


⑤Depending on requirements, you can set up multiple action groups.

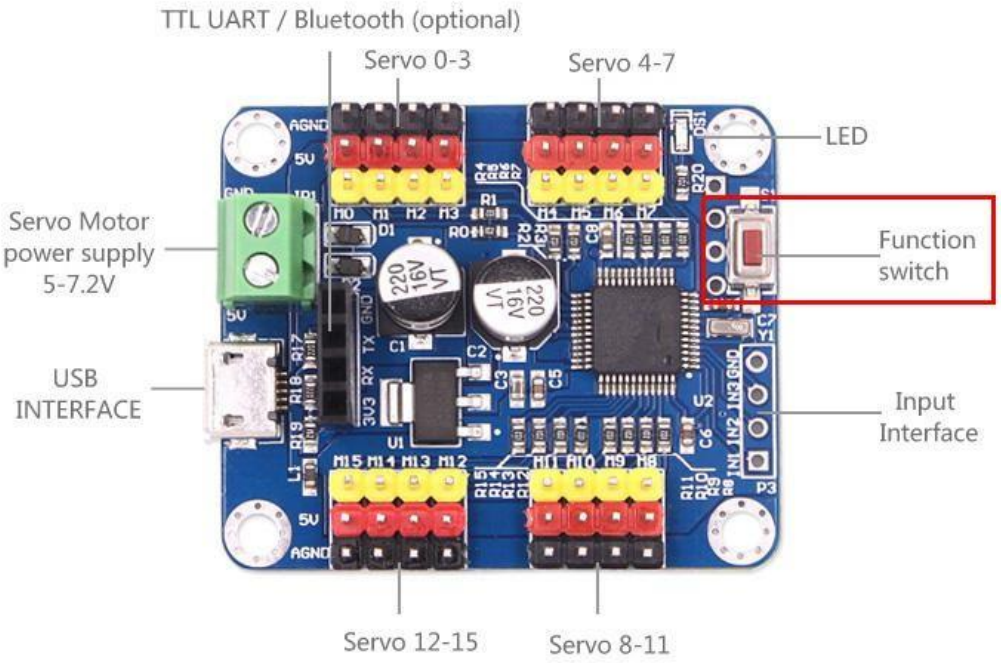


5.5.4 Download

(1) Click the “Download”, The signal on the control panel is always on. The bar of the downloading displayed”Download finished ”.

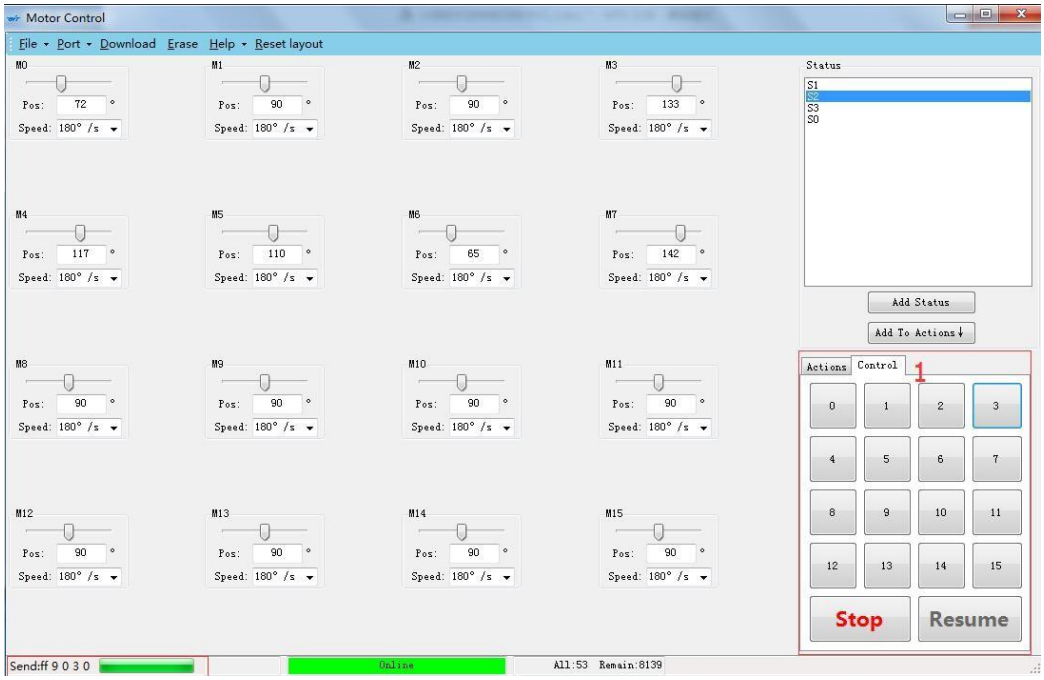


(2) Press the function switch, you can start action group 0, run the program offline.

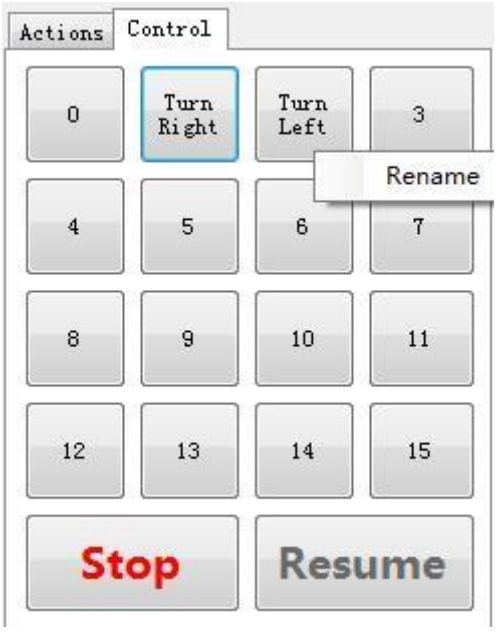


5.5.5 Control State

1. When all the action groups are downloaded, you can control arbitrarily action group through the control state.



Reminder: The action groups can be renamed, check the name of the action group and input the new name.



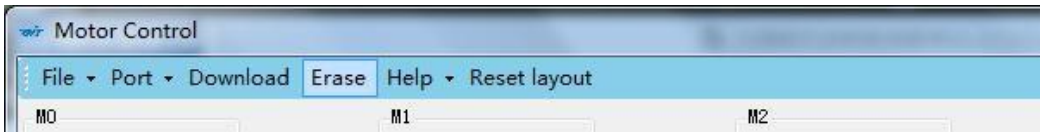
2. Emergency stop and resume

When you need an emergency stop, click on the emergency stop, the servo motor will stop.

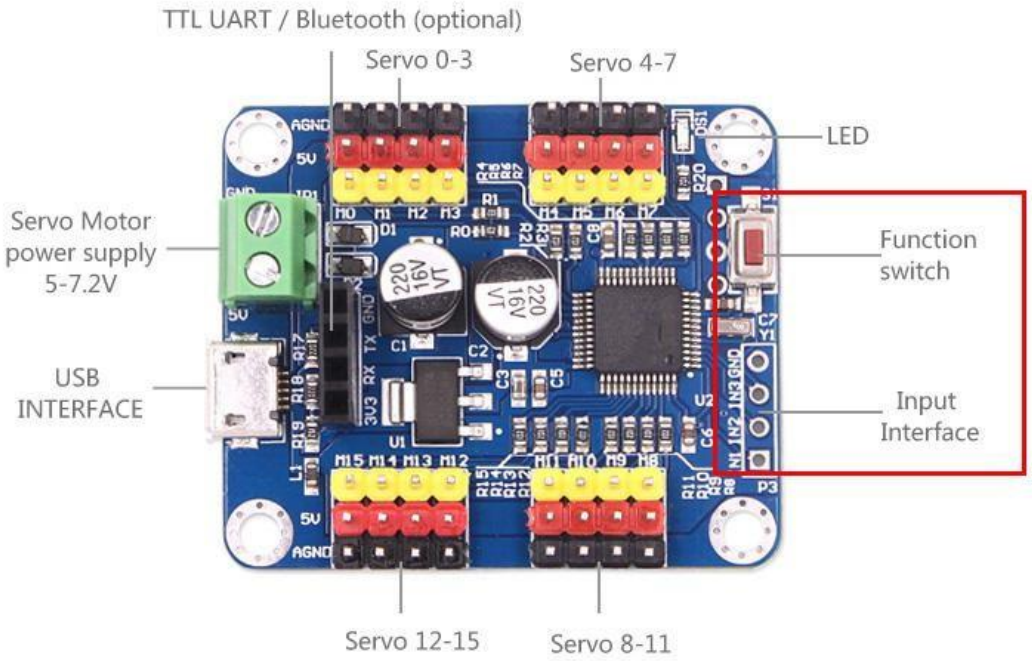
When you need to restore, click Restore, click will resume action group.

5.5.6 Chip Erase

When you need get a chip erase, you should just click “Erase” and all the action group will be deleted.



5.5.7 External Signal Input



Enter	operating	Features
S1	click it once	Start / stop action group 0
IN3	IN3 --- GND shorted	Start / stop action group 0
IN2	IN2---GND shorted	Start / stop action group 1
IN1	IN1---GND shorted	Start / stop action group 2

6.Serial Communication Protocol

Level: The TTL level(if the module is connected to the RS232 level, it may cause damage to module)

Baud rate: 9600 Stop bit: 1 Check bit: 0

6.1 PC Software to Control Panel

Reminder:

1. The factory default setting uses a serial port with a baud rate of 9600. Configuration can be configured through the host computer software, because all configurations are power-down save, so only need to configure once on the line.
2. Data format

0xFF	CMD	CH	DataL	DataH

Every Packet contains 5 bytes. Note that data is sent in hexadecimal not ASCLL mode. The first byte is the start code 0XFF, the second byte is CMD, the third byte is channel number, which controls the data of the corresponding channel. The fourth and fifth bits are the low and high bytes of the data, respectively.

CMD instruction table

Instruction	Function
0x01	Speed control
0x02	Position control
0x09	Action group start
0x0b	Emergency stop/ recovery

6.1.1 Speed Control of Servo motor

0xFF	0x01	CH	DataL	DataH

CH: Servo motor channel number, value 0- 15.

DataL, DataH: The and high bytes of the data, The two are combined into a short type of data with a symbol- Data.

Data indicates the speed of the servo motor, the unit (9 ° / s), the value of 1 to 20.

DataL=Data&0xff； DataH=Data>>8. Example:

1. Servo motor no.1, rotational speed: 10(90° / s) : Sent instructions: 0xFF 0x01 0x01 0x0a 0x00，where 0x000a is the decimal 10
2. Servo motor no.1, rotational speed: 20(180° / s) : Sent instructions: 0xFF 0x01 0x01 0x14 0x00, 0x0014 is the decimal 20

6.1.2 Position Control of Servo motor

0xFF	0x02	CH	DataL	DataH

CH：Servo channel number, the value of 0 ~ 15.

DataL，DataH：Data of the low byte and high byte, the two combined into a signed short type of data Data, DataL = Data & 0xff; DataH = Data >> 8. Data indicates the position of the steering gear, unit us (0.09 °), the value is 500 ~ 2500, the control pulse width of the steering gear is 500us ~ 2500us, the corresponding angle 0 °~ 180 degrees。

Example：

1) Servo 0, position 1500us (90 °): send command 0xFF 0x02 0x00 0xdc 0x05, where 0x05dc is the decimal 1500。

2) Servo 1, speed 500us (0 °): send command 0xFF 0x02 0x01 0xf4 0x01, where 0x01f4 is the decimal 500。

6.1.3 Action Group Settings

0xFF	0x09	0x00	DataL	DataH

Data: The number of the action group. Value:1- 15

Execute the action group, you can sent the instructions to the control panel through the serial port.

1． Action group1: sent instruction 0xFF 0x09 0x00 0x00 0x00, 0x0000 represent 0 of 10decimal system.

2． Action group2: sent instruction 0xFF 0x09 0x00 0x01 0x00, 0x0010 represent 0 of 10decimal system.

3． Action group3: sent instruction 0xFF 0x09 0x00 0x02 0x00, 0x0020 represent 0 of 10decimal system.

4． Action group4: sent instruction 0xFF 0x09 0x00 0x03 0x00, 0x0030 represent 0 of 10decimal system.

5． Action group5: sent instruction 0xFF 0x09 0x00 0x04 0x00, 0x0040 represent 0 of 10decimal system.

6． Action group6: sent instruction 0xFF 0x09 0x00 0x05 0x00, 0x0050 represent 0 of 10decimal system.

7． Action group7: sent instruction 0xFF 0x09 0x00 0x06 0x00, 0x0060 represent 0 of 10decimal system.

8． Action group8: sent instruction 0xFF 0x09 0x00 0x07 0x00, 0x0070 represent 0 of 10decimal system.

9． Action group9: sent instruction 0xFF 0x09 0x00 0x08 0x00, 0x0080 represent 0 of 10decimal system.

10． Action group10: sent instruction 0xFF 0x09 0x00 0x09 0x00, 0x0090 represent 0 of 10decimal system.

11． Action group11: sent instruction 0xFF 0x09 0x00 0x0a 0x00, 0x00a0 represent 0 of 10decimal system.

12． Action group12: sent instruction 0xFF 0x09 0x00 0x0b 0x00, 0x00b0 represent 0 of 10decimal system.

13． Action group13: sent instruction 0xFF 0x09 0x00 0x0c 0x00, 0x00c0 represent 0 of 10decimal system.

14． Action group14: sent instruction 0xFF 0x09 0x00 0x0d 0x00, 0x00d0 represent 0 of 10decimal system.

15． Action group15: sent instruction 0xFF 0x09 0x00 0x0e 0x00, 0x000e0 represent 0 of 10decimal system.

16． Action group16: sent instruction 0xFF 0x09 0x00 0x0f 0x00, 0x00f0 represent 0 of 10decimal system.

6.1.4 Emergency Stop and Recovery

0xFF	0x0b	0x00	DataL	DataH

DataL, DataH: low byte and high byte of data, the two are combined into a signed short type of data Data.

DataL=Data&0xff；DataH=Data>>8。Data That emergency stop or recovery, the value of 0 to

1.

1 for emergency stop, 0 for recovery

Emergency stop: send command 0xFF 0x0b 0x00 0x01 0x00

Recovery: Send command 0xFF 0x0b 0x00 0x00 0x00

7. Appendix

DS1 state：

DS1	Control panel state
Normally on	Off-line
Slow blinking（1S per time）	On line
Fast blinking（0.2S per time）	Action group operation

深圳维特智能科技有限公司

WitMotion ShenZhen Co., Ltd

Servo motor controller drive board 16channels

E-mail :wit@wit-motion.com
Website :www.wit-motion.com/english.php
Address :Honghai building 1405 Songgang town Baoan District Shenzhen Guangdong Province China
AliExpress: https://witmotion.aliexpress.com/store/2029054

Three Functions USB-Serial Port Module SPECIFICATION

Model ：USB-Serial Port Module

Description ： USB-Serial Port Module USB-TTL USB-232 USB-485

Production Standard

Enterprise quality system standard: ISO9001:2016

Tilt switch production standard：GB/T191SJ 20873-2016

Criterion of detection：GB/T191SJ 20873-2016

Revision date：2018.08.22

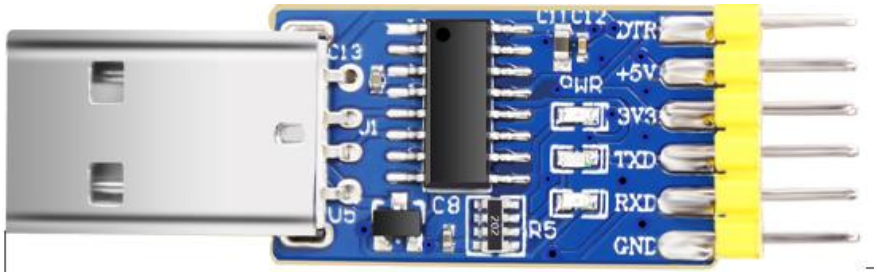
Version	Update content	Author	Date
V4.0	Release	Sharlene	20180821

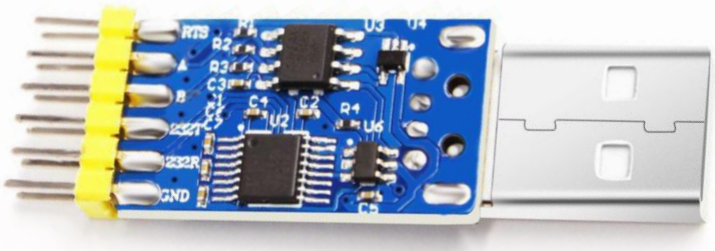
Catalog

1 Description

(1) Three functions, support USB-TTL USB-232 USB-485.
(2) Compatible Windows Linux Wince operating system.
(3) With a 500mA fuse protection circuit to prevent shorting and over current burning.
(4) The indicator lights indicate the working status of the device in red, yellow and green. COM port selects USB (yellow light) is always on, data reception RX (red light) flashes, data transmission TX (green light) flashes.
Small size, high stability, easy to carry.
(5) Small size, high stability, easy to carry.
(6) Compatible with 3.3V/5V voltage input and output, can supply power to the MCU.
(7) Intelligent identification switch serial port mode, no need to manually switch, easy and convenient.

2 Pin Description





Name	Function
+5V	Module power, 5V input, output
3V3	Module power, 3.3V output
RX	Serial data input, TTL level
TX	Serial data output, TTLlevel
232R	Serial data input, 232 level
232T	Serial data output, 232level
A	RS485 Signal line A
B	RS485 Signal line B
GND	GND
DTR	Data terminal preparation/control flow output
RTS	Request to send

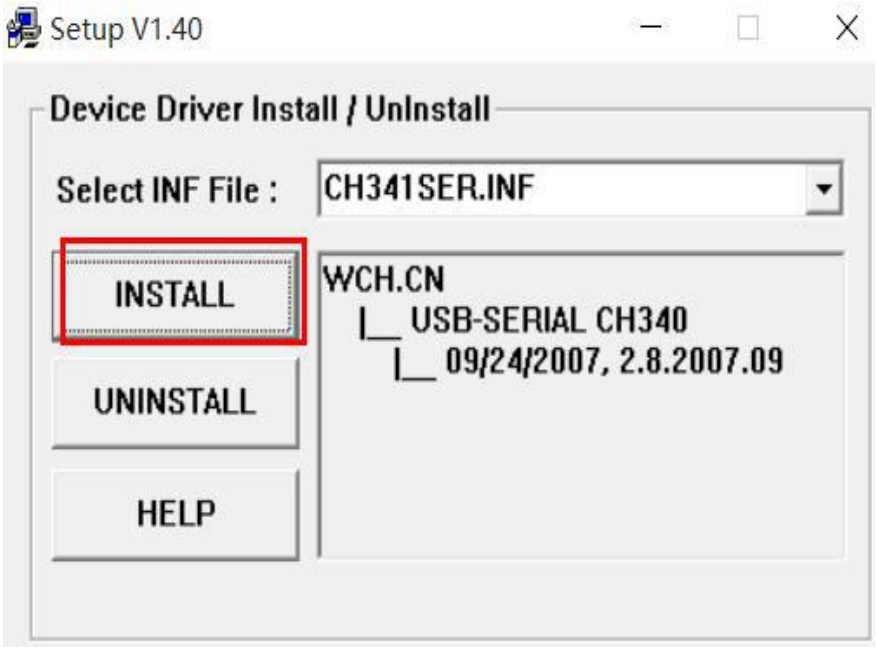
3 Method

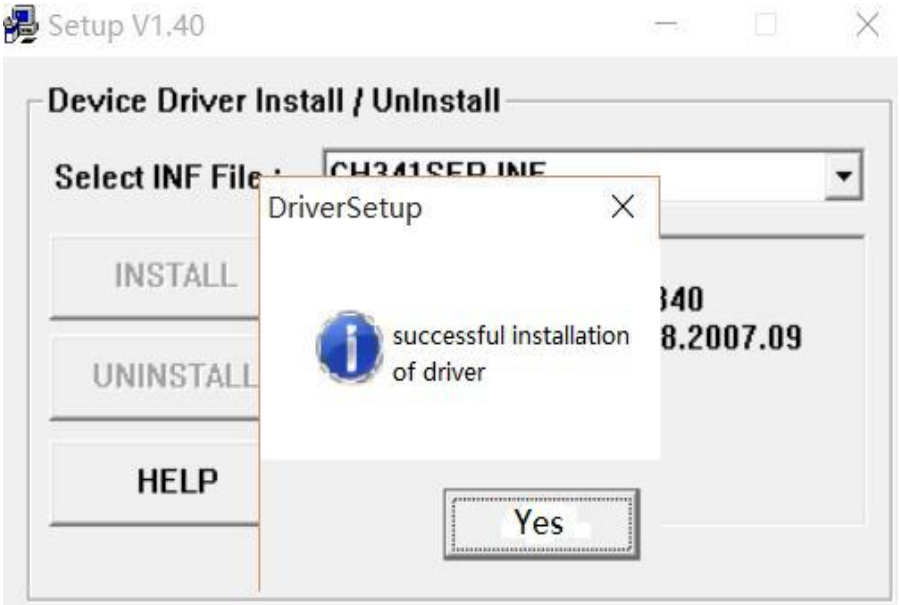
3.1 Install driver

First, the module is connected via USB-TTL module to the computer, open the package of CH340 and install the module driver. Open CH341SER.exe, click INSTALL and then it will output a COM.

CH340 drive:

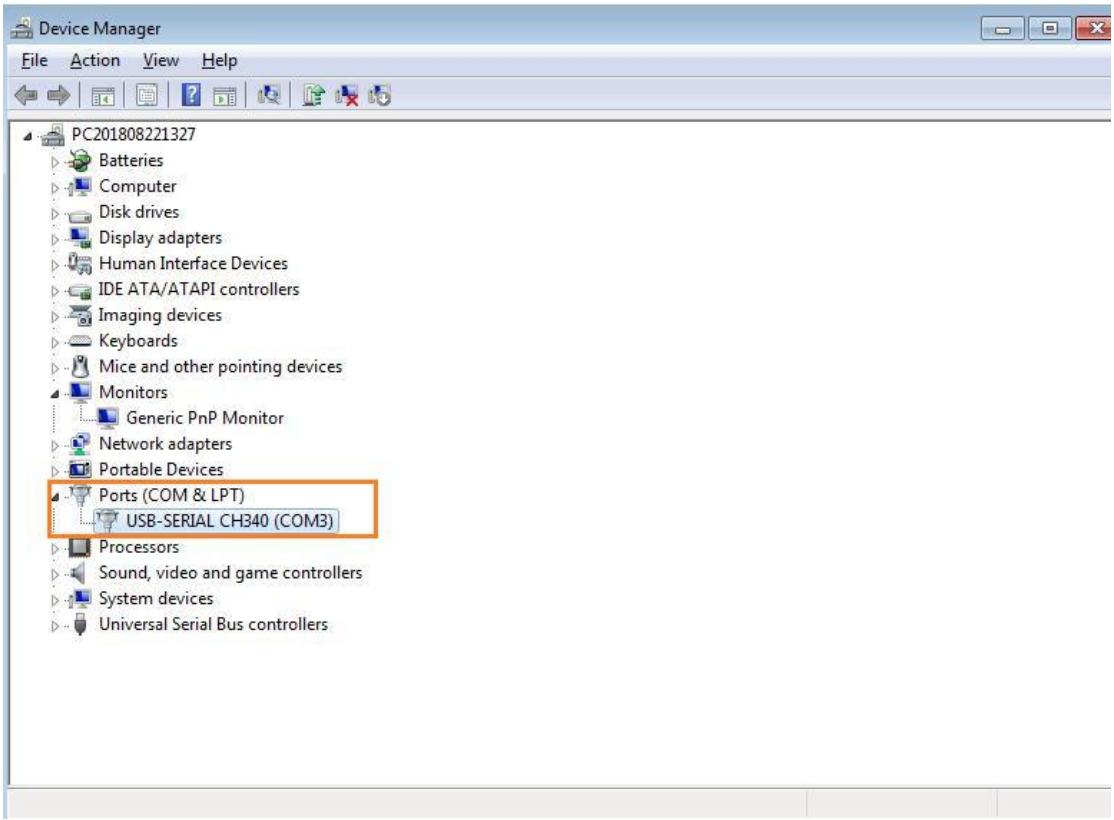
https://wiki.wit-motion.com/english/doku.php?id=communication_module





3.2 Check port number

Right click my computer\management\Device manager\Port（COM&LPT）. You can see the generated port number, the port number is COM3.

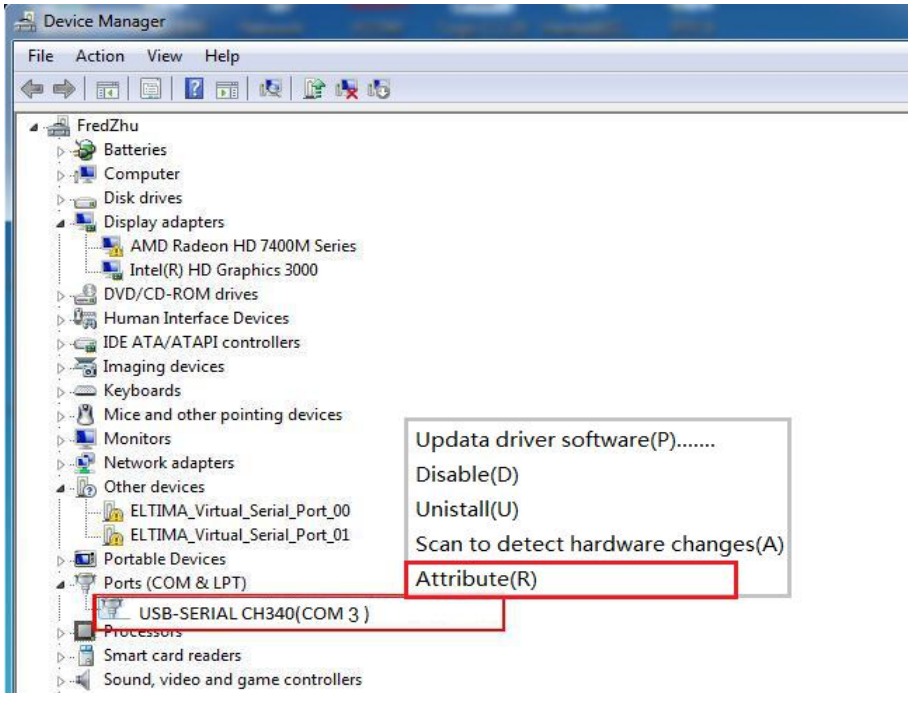


Change port number

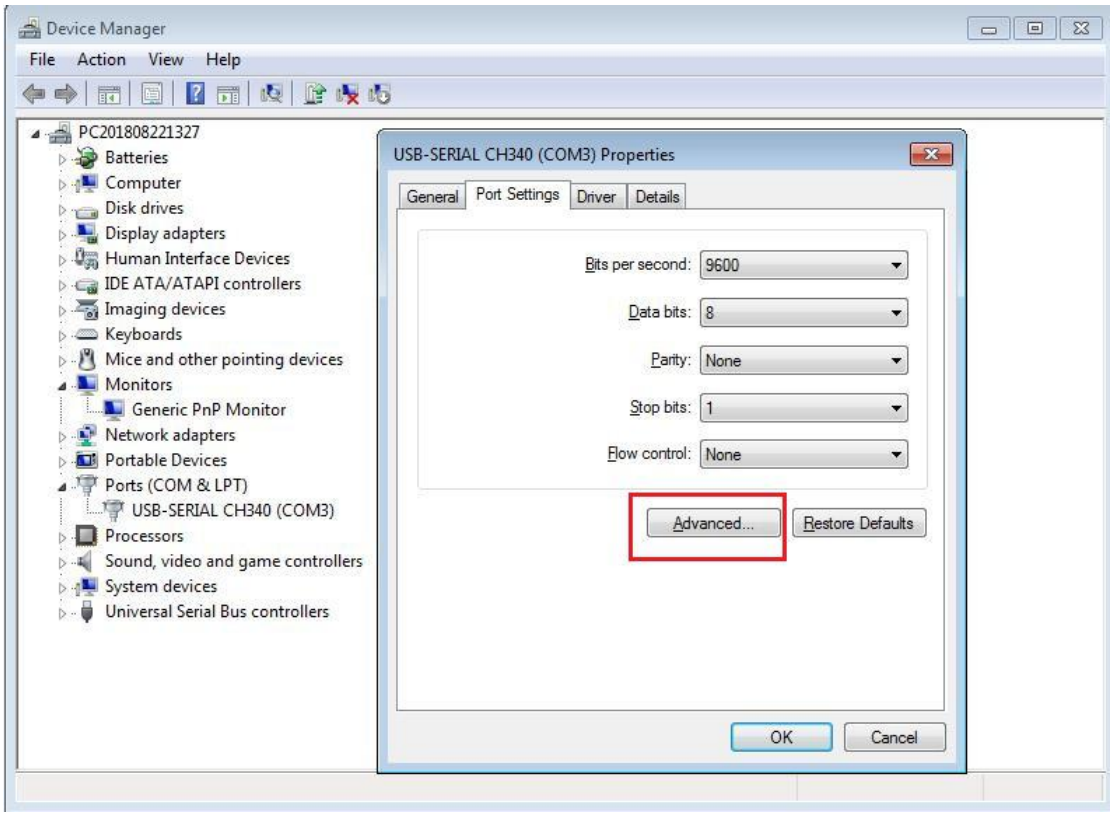
Sometimes we will insert multiple USB serial ports into the computer. Sometimes we want the serial port number to be assigned in the way we expected, so we need to manually adjust the serial port number. For example, change the serial port 16 above to a serial port.

The steps are as follows:

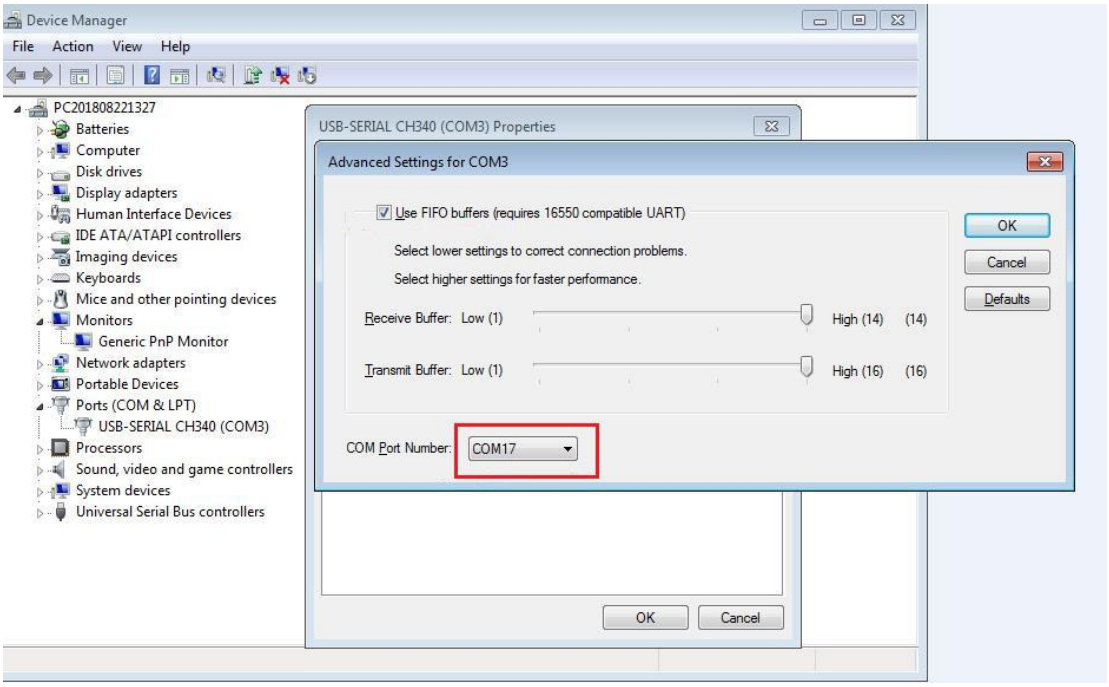
Open the device manager, right click on the USB-SERIAL CH340 and select Attribute



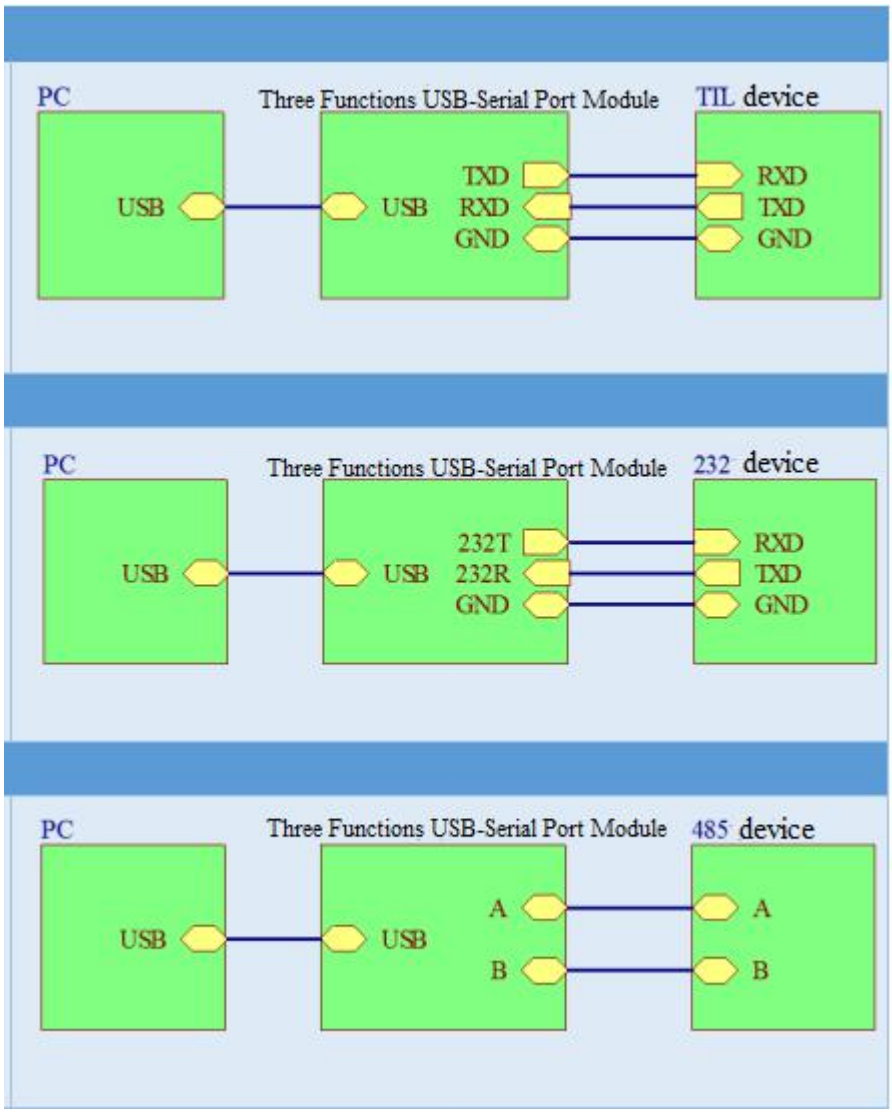
Click on Port Settings, then click on "Advanced Settings".



Change the COM port and click OK.



4 Connect Description



5 Function test

Product functionality can be verified by USB to TTL self-loop test and USB to 232 self-loop test. Methods as below:

5.1 USB-TTL

Connect the TXD and RXD of the module to the DuPont line. Dial the corresponding DIP

switch according to the instructions in the function selection, then insert the module into the computer and use the serial debugging assistant to send data to see if there is corresponding data to return. If you can receive the data sent, the module is functional.

5.2 USB-232

Connect the module's 232T and 232R with DuPont line, dial the corresponding DIP switch according to the instructions in the function selection, then insert the module into the computer and use the serial port debugging assistant to send data to see if there is corresponding data to return. If you can receive the data sent, prove that the module is functioning properly.

5.3 USB-485

(This mode test needs to be combined with other 485 devices, such as two three-in-one modules)

Take two three-in-one serial port modules, connect A and B of the two modules with DuPont line, A to A and B to B. Insert the module into the computer, open two serial port debugging assistants, and select the serial port numbers corresponding to the two three-in-one serial port modules. Use one of them to send and see if another serial port can receive the corresponding data. If you can receive the data sent, the module is functional.

深圳维特智能科技有限公司

WitMotion ShenZhen Co., Ltd

Three Functions USB-Serial Port Module

TEL : (+86) 755-33185882
E-mail : wit@wit-motion.com
Website : www.wit-motion.com
Aliexpress : https://witmotion.aliexpress.com
Alibaba : https://witmotion.en.alibaba.com
Wit-wiki : https://wiki.wit-motion.com/english
Address : Honghai building 1306 Songgang town Baoan District Shenzhen Guangdong Province China