UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO, ESTUDIO Y SIMULACIÓN DE UN ROBOT BÍPEDO EN COPPELIASIM (V-REP) Y CONTRUCCIÓN DE UN PROTOTIPO EN IMPRESIÓN 3D

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Guillermo López Valero

TUTORIZADO POR

Leopoldo Armesto Ángel

CURSO ACADÉMICO: 2019/2020

Agradecimientos,

A mi familia y amigos por apoyarme y ayudarme durante el grado y la realización del tfg.

A mi tutor Leopoldo Armesto Ángel por aconsejarme siempre.

RESUMEN

Este proyecto consta de una primera parte en la que se crea el diseño de un robot bípedo con 4 servos por pata, lo que supone un total de 8 grados de libertad. Para este diseño se ha utilizado un software CAD (SolidWorks) para modelado mecánico en 2D y 3D en el que se han creado todas las piezas del robot y con el que se ha simulado su ensamblaje completo.

La segunda parte del proyecto se centra en el estudio y simulación del movimiento de este robot que hemos creado, programándolo en un entorno similar a C con el objetivo de que camine de forma estable, además de realizar otros movimientos. Para esta parte se ha utilizado un simulador de robots muy completo llamado CoppeliaSim.

Por último, se ha impreso con la ayuda de una impresora en 3D y su respectivo software de impresión, un prototipo del robot diseñado, y se han utilizado servos y placas similares a arduino para el m--ontaje de este, con la finalidad de que, en un posterior proyecto, se programe y controle con una App para smartphones creada con App Inventor.

PALABRAS CLAVE

Robot, Robótica, Bípedo, Programación, Caminante, Diseño, Impresión 3D, Electrónica.

ABSTRACT

This project consists of a first part in which the design of a biped robot with 4 servos per leg is created, representing a total of 8 degrees of freedom. CAD (SolidWorks) software for 2D and 3D mechanical modeling has been used for this design, in which all the robot parts have been created and where its complete assembly has been simulated.

The second part of the project focuses on the study and simulation of the movement of this robot that we have created, programming it in an environment similar to C in order to make it walk stably, in addition to performing other movements. For this part, a very complete robot simulator called CoppeliaSim has been used.

Finally, a prototype of the designed robot has been printed with the help of a 3D printer and its respective printing software, and servos and boards similar to arduino have been used for the assembly of this, in order to in a later project, program it and control it with an app for smartphones created with App Inventor.

KEYWORDS

Robot, Robotics, Bipedal, Programming, Walking, Design, 3D Printing, Electronics.

RESUM

Aquest projecte consta d'una primera part en la qual es crea el disseny d'un robot bípede amb 4 servos cada cama, el que suposa un total de 8 graus de llibertat. Per a aquest disseny s'ha utilitzat un programa CAD (SolidWorks) per modelatge mecànic en 2D i 3D amb el que s'han creat totes les peces del robot i s'ha simulat el seu acoblament complet.

La segona part del projecte es centra en l'estudi i simulació del moviment d'aquest robot que hem creat, programant en un entorn similar a C amb l'objectiu de que camini de manera estable, a més de realitzar altres moviments. Per aquesta part s'ha utilitzat un simulador de robots molt complet anomenat CoppeliaSim.

Finalment, s'ha imprès amb l'ajuda d'una impressora en 3D i el seu respectiu programa d'impressió, un prototip del robot dissenyat, i s'han utilitzat servos i plaques similars a arduino per al muntatge i acoblament d'aquest, amb la finalitat que en un posterior projecte, poder programar-lo i controlar-lo amb una App per a smartphones creada amb App Inventor.

PARAULES CLAU

Robot, Robòtica, Bípede, Programació, Caminant, Disseny, Impressió 3D, Electrònica.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO, ESTUDIO Y SIMULACIÓN DE UN ROBOT BÍPEDO EN COPPELIASIM (V-REP) Y CONTRUCCIÓN DE UN PROTOTIPO EN IMPRESIÓN 3D

MEMORIA

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Guillermo López Valero

TUTORIZADO POR

Leopoldo Armesto Ángel

CURSO ACADÉMICO: 2019/2020

1.- OBJETO

El objetivo de este proyecto era utilizar las herramientas y aplicar los conocimientos obtenidos a lo largo de los cuatro años de la carrera de ingeniería, en este caso, centrándome más en la parte robótica e informática de esta.

Por un lado, se ha utilizado un programa de diseño en 3D visto en una optativa de la carrera para crear las piezas desde cero de un robot bípedo, con una estructura sencilla de 4 servos por pata.

La segunda parte del proyecto consiste en la implementación de estas piezas en otro programa de ordenador, para así hacer el ensamblaje del robot y poder simular el movimiento de este, con propiedades físicas y dinámicas simulando tal cuál sería el robot en la vida real, y calculando los ángulos en los que colocar los servos, para así conseguir que ande de forma estable.

Por último, con la ayuda de una impresora 3D y su respectivo programa de impresión, se han impreso las piezas del robot con material PLA, el material más respetuoso con el medio ambiente. De esta forma tenemos un prototipo del robot simulado en ordenador, no funcional de momento, ya que habría que comprar la electrónica y programarlo con Arduino.

1.1.- Contenido del proyecto

El trabajo o dossier final del proyecto surge de la unión de cuatro documentos distintos, los cuales forman el proyecto completo.

El primer documento, del cual ya forma parte este apartado, se trata de la MEMORIA del proyecto. Aquí se explica con detalle en qué ha consistido el trabajo, cuáles han sido las herramientas utilizadas, qué problemas ha habido y cuál ha sido el resultado final, además del proceso seguido para la creación del robot, bibliografía y demás apartados que están presentes en el índice del proyecto.

Después tenemos el segundo documento, PLANOS, en el que se encuentran todas las piezas creadas del robot, con sus respectivas vistas y cotas para así poder diseñarlas a partir de los planos. Además, también hay planos de conjunto donde se ven la unión de las partes del robot y su construcción, es por eso por lo que se ha dividido en planos de tipo “ensamblaje” y “piezas”, de esta forma se ve mejor estructurado.

El tercer documento será el PLIEGO DE CONDICIONES, un apartado meramente técnico y normativo donde se aclaran las condiciones para la realización el proyecto.

El cuarto y último documento es el de PRESUPUESTO, donde se analiza y calcula el coste que ha supuesto la realización de todo el trabajo desde dos puntos de vista distintos. Un primero cálculo en el que se tienen en cuenta los programas informáticos y equipo electrónico utilizado, ya que ha sido necesario un ordenador de sobremesa y tres programas informáticos principalmente, además de una impresora 3D. El segundo y último cálculo comprende la mano de obra y horas invertidas por el alumno y su tutor.

1.2.- Contexto y antecedentes

Este proyecto ha sido realizado desde el punto de vista de la ingeniería automática y robótica. Es por esto por lo que el contexto del trabajo se ve rodeado mayoritariamente por la historia de los robots y la programación, pero más concretamente los robots bípedos y autómatas (Ilustración 1).

Ilustración 1. Ejemplo de robot bípedo, en este caso de 2 servos por pierna. [1]

Hay mucha historia acerca de los robots y su evolución, ya que los orígenes de estos los podemos encontrar desde el mundo antiguo. Al principio no eran conocidos como robots, debido a que el término, *robot*, fue introducido para la gente a través de la obra R. U. R. (Robots Universales Rossum) del dramaturgo checo Karel Čapek (Ilustración 2), que se estrenó en 1920. Era una sátira donde los robots eran seres biológicos fabricados que realizaban todo el trabajo manual desagradable.

Ilustración 2. Karel Čapek, conocido por acuñar el concepto de robot. [2]

Podemos encontrar en la Wikipedia («Robot», 2020) la siguiente definición para la palabra Robot, *"Un robot es una entidad virtual o mecánica artificial. En la práctica, esto es por lo general un sistema electromecánico que, por su apariencia o sus movimientos, ofrece la sensación de tener un propósito propio. La independencia creada en sus movimientos hace que sus acciones sean la razón de un estudio razonable y profundo en el área de la ciencia y tecnología. La palabra robot puede referirse tanto a mecanismos físicos como a sistemas virtuales de software, aunque suele aludirse a los segundos con el término de bots".*

Por otro lado, La Real Academia Española (ASALE & RAE, s. f.) acoge cuatro definiciones distintas de la palabra *robot*, pero en nuestro caso nos quedaremos con estas dos: *"Máquina o ingenio electrónico programable que es capaz de manipular objetos y realizar diversas operaciones"* y *"robot que imita la figura y los movimientos de un ser animado".*

Las tres definiciones sirven para dar una idea general de qué se tratan los robots, y es que hoy en día podemos ver robots de muchas formas y tamaños, con funciones totalmente distintas, más complejos o sencillos, pero al final todos entran dentro de esas definiciones.

1.2.1.- Robots caminantes

Como he explicado antes, podemos encontrar robots de muchas maneras, pero unos de los más impactantes y que ha ganado terreno conforme la electrónica y las fuentes de energía han evolucionado, son los robots con características biológicas. Observar el movimiento de un autómata sobre sus piernas es más sorprendente y complejo de entender que el mismo sobre ruedas.

La mayoría de estos robots se tratan de estructuras diseñadas para desplazarse mediante comportamientos simples copiados de los animales. Es por eso por lo que el número de patas y las articulaciones de estas influyen de manera importante en la creación de estos robots.

Uno de los motivos por los que cada vez se buscan diseñar robots capaces de caminar y moverse como insectos, animales o humanos, son para la aplicación de estos en tareas que pueden considerarse peligrosas para los humanos. De este modo se consigue poner a salvo las vidas humanas e incluso hacer las tareas de forma más precisa y rápida.

Los robots hexápodos son un ejemplo de robots caminantes y que imitan a insectos, en este caso a las arañas. El tener 6 patas y diversas articulaciones en estas le hace a este tipo de robot uno de los más adaptables y que mejor se desenvuelven en algunos terrenos. Además, 6 patas otorgan una estabilidad al robot muy superior a otros tipos de robots, les permite controlar las patas prácticamente por individual, asegurándose que el robot mantendrá el equilibrio.

Otro tipo de robot caminante son los robots cuadrúpedos. Estos tienen 4 patas y al igual que los hexápodos, también tienen bastante estabilidad y son más rápidos que los bípedos. Dependiendo la estructura de las patas, encontramos cuadrúpedos que imitan a mamíferos, con las patas verticales a la base y que por lo tanto les permite la capacidad de andar y con menos cantidad de fuerza par en las articulaciones. Esta configuración también les permite andar por regiones estrechas debido a las pequeñas pezuñas que tienen.

La otra configuración de robot cuadrúpedo es la que imitan a los insectos, pero con 4 patas. Cada pata tiene una de las extremidades horizontal, y la otra vertical. Sería como

una araña pero con 4 patas. Las ventajas de este diseño es que tiene mucha más estabilidad que el anterior, y también les permite más rango del momento para apoyar el pie. También tienen el centro de gravedad más bajo y pueden girar o cambiar la dirección del movimiento más fácilmente.

Por último, en cuanto a robots caminantes, tenemos los robots bípedos. Estos están compuestos por 2 piernas, y la mayoría de las veces buscan simular los movimientos de los humanos. Esta esta una tarea compleja pues las articulaciones de los humanos no son rígidas a diferencia de los robots, y esto puede causar algunos problemas. Además, al tener solo 2 puntos de apoyo, mantener el equilibrio y centro de gravedad es una tarea compleja en la programación de estos robots. Es por eso por lo que se han necesitado años y la evolución de los diseños y electrónica para conseguir robots que imitan de forma muy similar el caminar de los humanos. No todos los diseños son igual de complejos, un robot con tan solo 2 servos en cada pierna puede considerarse también un robot bípedo, pero la diferencia es que los movimientos y las funciones de este serán mucho menores a otro con muchos más servos y por lo tanto más libertad de movimiento.

1.2.2.- Grados de libertad

Al hablar sobre robots, da igual de qué tipo, es importante conocer el término *grados de libertad*, ya que ayudan a entender la libertad de movimientos que puedes hacer con las articulaciones del robot y qué limitaciones tienes.

Llamamos *grado de libertad* (g.d.l.) a cada una de las coordenadas independientes que necesarias para describir la posición y la orientación en el espacio de los elementos del robot.

Si pensamos que tenemos un robot en un espacio tridimensional, cada par eslabón-articulación tiene un solo grado de libertad, de rotación o traslación, pero una articulación puede tener más de un g.d.l. si varios ejes cortan entre sí.

La capacidad de moverse de este a lo largo de los tres ejes de translación perpendiculares sumaría tres grados de libertad para el robot. Estos movimientos serían hacia delante/atrás, arriba/abajo e izquierda/derecha. En la imagen de abajo (Ilustración 3) correspondería a los ejes perpendiculares de color rojo, azul y amarillo respectivamente.

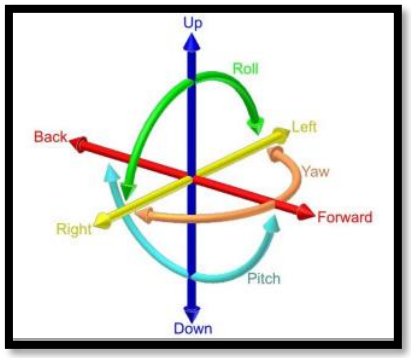


Ilustración 3. Todos los movimientos y grados de libertad posibles por una articulación. En total son seis grados de libertad: adelante/atrás, arriba/abajo, izquierda/derecha, cabecear, guiñar y rodar. [3]

Los últimos tres movimientos serían la rotación de los tres ejes perpendiculares, es decir, el cabeceo, el guiño y el rodar. En la Ilustración 3 los identificamos como el eje de rotación de color celeste, naranja y verde respectivamente.

En la siguiente imagen (Ilustración 4) vemos mejor representado lo que estamos explicando de los g.d.l. de un robot.

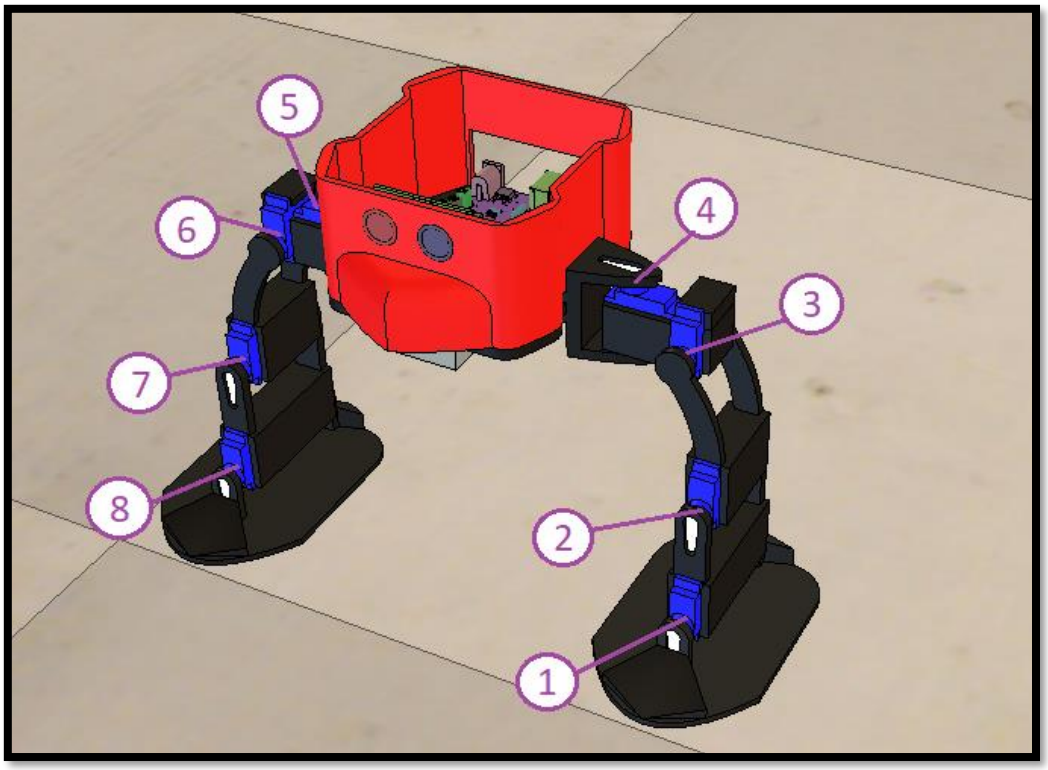


Ilustración 4. Representación de los ejes y grados de libertad del robot diseñado para el proyecto. Es una simulación en Solidworks y el robot tiene un total de 8 g.d.l.

Este es el robot diseñado del proyecto, y hemos introducido los *shapes* o modelos de las piezas en el programa Solidworks, para así ensamblar el robot y poder trabajar con las físicas y movimientos.

Cada número señala a un servo del robot, por lo tanto, a una articulación y por consiguiente suma un g.d.l. a nuestro robot. En total está compuesto por ocho servos, cuatro en cada pierna. El "1" y el "8" son los servos y articulaciones de los tobillos. El "2" y el "7" corresponden a las rodillas, ya tenemos 4 g.d.l. Y el "3", "4"; "5" y "6" forman las dos caderas de nuestro robot, lo que significa que cada cadera tiene dos servos y, en definitiva, 8 grados de libertad para nuestro robot

1.2.3.- Software de simulación de robótica

Como en muchos otros ámbitos además de en ingeniería, hay proyectos robóticos que cuestan tiempo y dinero y que, por lo tanto, no puedes fabricarlos hasta saber con más o menos certeza que va a funcionar. Es por eso por lo que se desarrollan herramientas y softwares de simulación, para que antes de poner en práctica lo diseñado, se haga una prueba en ordenador. Además, estos simuladores ofrecen la posibilidad de cambiar las variables, por lo que además de verificar el correcto funcionamiento del diseño, puedes modificarlo y jugar con los valores hasta que des con la solución correcta.

En el caso de softwares de simulación de robots podemos encontrar varios disponibles, ya sean gratuitos o de pago, y algunos más especializados en unos robots u otros.

- CoppeliaSim (V-Rep): actualmente conocido como CoppeliaSim, pero antes llamado V-Rep, este es uno de los softwares de simulación más completos y disponibles para su uso. Es el que se ha decidido usar en la simulación de este proyecto. Te permite simular robots que hayas diseñado por tu parte o sino utilizar algunos de los modelos que contiene el propio programa. Se pueden utilizar 6 lenguajes de programación distintos como C/C++, Python o Java. Además, contiene funciones de cálculo de la cinemática, de planeación de trayectorias, sensores de visión, de proximidad, etc. Este software está especializado en todo tipo de robots, ya sean caminantes, con ruedas, o robots utilizados en las industrias.

- RobotStudio: es el software de simulación y programación de la empresa de robots ABB, líder mundial en ingeniería eléctrica y automatización. Contiene réplicas digitales de los modelos de sus robots, además de los sistemas físicos para que se puede ver lo que suceden en las líneas de producción. Las herramientas de programación permiten crear, simular y probar instalaciones completas de robots en un entorno virtual 3D sin tener que perturbar la línea de producción real. Este simulador está especializado en los robots industriales o brazos robóticos.

- RoboDK: un software similar a RobotStudio, creado para simular y programar robots fuera de las líneas de fabricación. Se puede crear un ambiente virtual y acceder a bibliotecas con robots industriales de más de 30 fabricantes diferentes. Permite cargar un modelo en 3D del robot y controlar el movimiento de sus articulaciones, además de generar trayectorias.

2.- DESARROLLO DEL PROYECTO

En la asignatura optativa Robótica móvil, cursada en tercero de carrera, trabajamos por primera vez o al menos en mi opinión, con programas muy interesantes con los que simular, diseñar y programar todo tipo de robots. En esta optativa disfruté de cada clase y sobre todo con el proyecto final, donde diseñabas más o menos tu propio robot con ruedas, y tenías que programarlo y crear su propia aplicación para smartphones y así controlarlo. Era algo nuevo y divertido a la vez que requería trabajo.

En ese momento decidí enfocar mi TFG en algo relacionado con el diseño y programación de un robot. En este caso ha sido a raíz de ir descartando opciones o de surgir problemas, el diseño de las piezas de un robot bípedo con uno de los programas que explicaremos más adelante, *SolidWorks*.

Una vez creadas las piezas del robot y ver que el ensamblaje queda firme y tal cual se buscaba, pasé las piezas a otro programa de simulación y programación de robots, *CoppeliaSim.* Con esta herramienta consigues darle propiedades físicas al robot, en un entorno totalmente modificable y simulado como el mundo real. Aquí es donde se ha realizado la programación de los movimientos del robot como si se tratara de un robot físico programado por ejemplo con Arduino.

Por último, puesto que mi idea inicial era imprimir el robot diseñado, montarlo y programarlo físicamente con una electrónica que indicaré en los próximos apartados, decidí imprimir el robot para ver como quedaría realmente y ver que sería posible programarlo también como esperaba.

2.1.- Soluciones alternativas

En este apartado se detallan las soluciones variadas que tenía para este proyecto y que al final por unas cosas u otras han sido descartadas, ya sea desde el punto de vista del enfoque del proyecto, como también el diseño del robot y la electrónica que se iba a utilizar.

2.1.1.- Diseño

- 12 grados de libertad: en primer lugar, el diseño del robot iba a ser de uno con 6 servos en cada pierna, simulando lo más parecido posible a las articulaciones de las piernas humanas. Es decir, tendría una cadera con 2 servos, una rodilla con otros 2 grados de libertad, y por último el tobillo o movimiento del pie lo controlaría los 2 últimos servos. Como vemos en la siguiente imagen (ilustración 5), este sería el prototipo pensado con 6 servos en cada pierna, más complejo que el diseñado al final, pero también más común.
  Se descartó esta idea ya que había anteriormente un proyecto parecido en curso y se buscaba algo diferente.

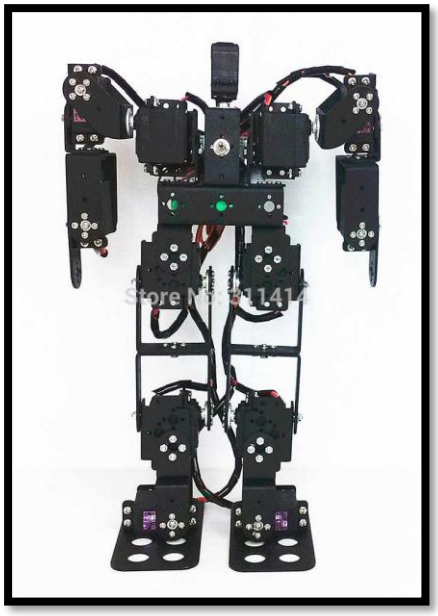


Ilustración 5. Primer diseño e idea pensada para la creación de un robot bípedo. Este robot tiene 12 grados de libertad en total. [4]

- 8 grados de libertad: una opción más barata en cuanto a servos, pero también con más limitaciones ya que dispones de menos grados de libertad y menos juego de articulaciones. El robot sería algo similar a lo que vemos en la imagen superior derecha (Ilustración 6), una pierna compuesta por 2 servos simulando el juego de cadera, un tercer servo para la rodilla y un último servo para mover el tobillo.

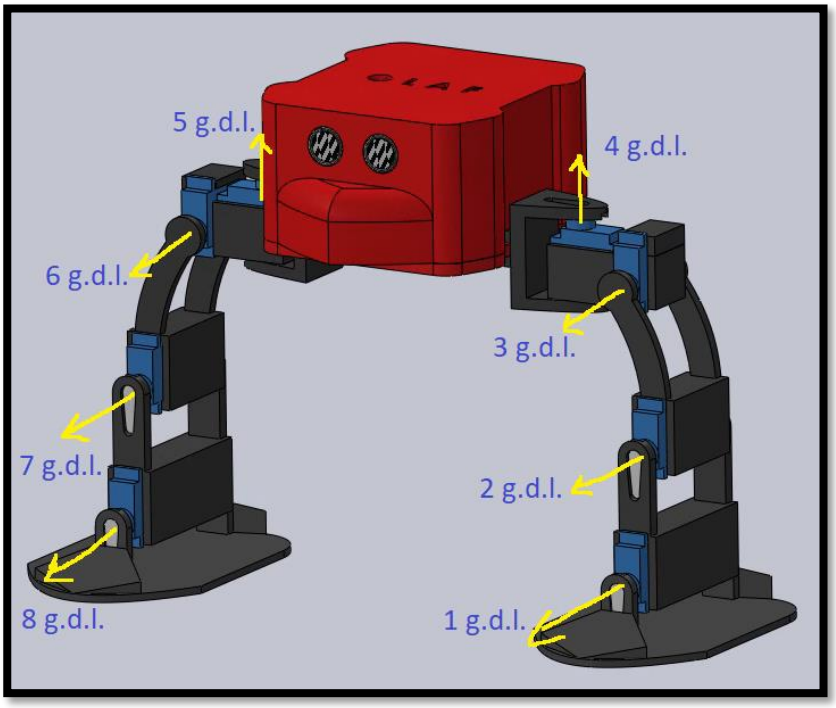


Ilustración 6. Diseño de un robot de 8 grados de libertad.

2.1.2.- Electrónica

Como ya he mencionado anteriormente, el proyecto al final se ha enfocado desde un punto de vista de diseño y de simulación. Eso no quiere decir que no se tuviese en cuenta qué electrónica utilizar en caso de montar el prototipo en 3D del robot diseñado y programarlo con Arduino. Este apartado es para mencionar qué electrónica se pensó y cuáles son las ventajas y desventajas de cada alternativa:

- Arduino Uno R3: es una de las placas más usadas y documentadas del mundo (Ilustración 7). Al ser de Arduino hay muchas bibliotecas disponibles en internet para poder programar con ellas. Tiene 14 pines de entrada/salida digitales y 6 entradas analógicas por lo que dispone de muchos pines para su uso, ya sea con servos u otra electrónica. No dispone de bluetooth integrado o wifi por lo que habría que comprar un bluetooth externo en caso de querer crear una aplicación para smartphone y enviar órdenes al Arduino.

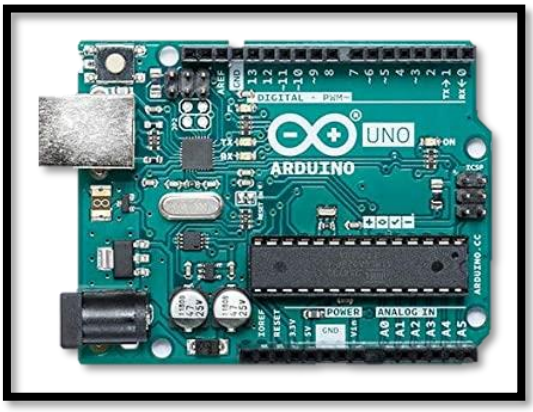


Ilustración 7. Placa de Arduino Uno R3. 16 pines digitales y 6 entradas analógicas. [5]

- Wemos ESP32 D1 R32: se trata de una placa con WiFi y Bluetooth integrados (Ilustración 8), por lo que no es necesario comprar un dispositivo de Bluetooth por separado. Además, es compatible con los *shields* de Arduino UNO (placa que se coloca sobre otra placa Arduino para ampliar las capacidades de esta). Tiene 6 entradas analógicas, igual que la de Arduino UNO, pero también 20 entradas/salidas digitales, lo cual dispone de más pines que la anterior. El procesador que utiliza es más rápido también tanto en memoria flash, RAM, etc.

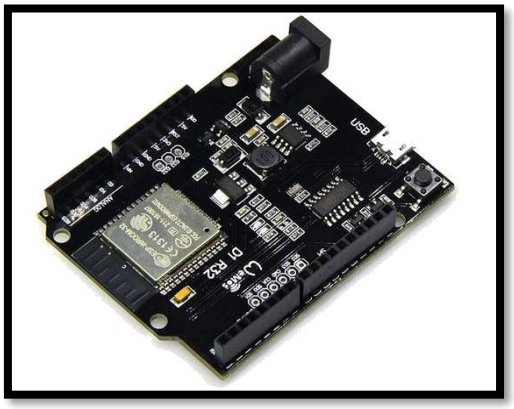


Ilustración 8. Placa Wemos ESP32 D1 R32 con Wifi y Bluetooth integrados. 20 pines digitales y 6 entradas analógicas. [6]

- Servomotor MG995: es un servo motor digital con piñonera metálica (Ilustración 9), que rota hasta 120º. El voltaje de operación es de 4.8 V a 7.2 V y de alta velocidad. Tiene un torque, es decir, la fuerza que tiene de giro, de 8.5 kgf·cm (4.8 V) y 10 kgf·cm (6 V). Muy potente para asegurarse que el servo no se queda sin fuerza par al intentar mover algo muy pesado. El problema es que el peso son 55 g y sus dimensiones de largo 40.7 mm, ancho 19.7 mm y altura 42.9 mm. Bastante grande y pesado por lo que el diseño del robot también tendría que ser grande. Además, es bastante caro y puesto que se necesitan varios de ellos no es la mejor opción, pero sí una para tener en cuenta.

Ilustración 9. Servomotor MG995 con sus accesorios y tornillería necesaria. [7]

- Servomotor SG90: este es uno de los servos más utilizados en la robótica para aprendizaje y prácticas con servos, ya que es de diminutas dimensiones y muy económico (Ilustración 10). Tan solo pesa 9 gramos y las dimensiones son casi la mitad que el MG995. El inconveniente es que el Par que tiene es de 1.2kc/cm a 4.8 V por lo que para mover ciertas piezas puede quedarse corto de fuerza. Es compatible con Arduino y otros microcontroladores.

Ilustración 10. Servomotor SG90. [8]

- **Batería Lipo:** se trata de una pila recargable que dispone de varias células, dependiendo la que se elija, y estas células equivalen a baterías pequeñas conectadas en paralelo para conseguir un aumento en la corriente de descarga. Las hay de muchos tipos y especificaciones, pero para este proyecto valdría con una batería Lipo de 1300 mAh de capacidad, con 2 celdas para tener un voltaje nominal de 3.7x2 V y 20C. Este último valor indica la capacidad de descarga de la batería, en este caso descargaría a 2.2x20 A (44 A) en 60/20 minutos. Es decir, en el caso de que el robot consuma 44 A, lo cual es muy por encima de lo que en realidad consumirá, la batería podrá otorgar esa corriente durante 3 minutos.

Ilustración 11. Batería Lipo de 2200 mAh y 2 celdas, voltaje nominal 7.4 V, y una capacidad de descarga de 20C (descarga a 20x2.2 A en 3 minutos). [9]

- **Baterías Li-ion 18650:** una solución más barata y menos delicada que las baterías Lipo. Simplemente se trata de alimentar la placa y los servos con 4 pilas de Li-ion de 3.7 V cada una y con una capacidad de 9800 mAh, no siempre real por lo que pueden ser de 2800 mAh en verdad. Las 4 pilas se pondrían en un soporte adaptado, que además de permitir cargas las baterías por micro-USB o tipo C, dispone de pines para proporcionar 3V/1A y 5V/3A. En la siguiente imagen vemos las pilas y el soporte comentado anteriormente (Ilustración 12).

Ilustración 12. A la izquierda soporte de 4 baterías 18650 para recargar o proporcionar corriente. A la derecha 2 baterías Li-ion 18650 de 3.7 V. [10] [11]

2.2.- Solución Adoptada y elección de los componentes

2.2.1.- Diseño

- **8 grados de libertad:** para el proyecto final se ha elegido realizar un robot bípedo de 8 grados de libertad, por lo que el robot tendría un total de 8 servos, 4 en cada pierna. Se descartó el de 12 servos ya que había un proyecto similar en curso, no terminado, y yo buscaba hacer algo distinto. Además, al necesitar menos servos, la elección de la electrónica sería más sencilla o al menos ya no sería necesario una placa expansora o controlador de servos, puesto que la placa elegida no disponía de tantas salidas ni podría suministrar tanta corriente.
  Además de los servos, el diseño se ha decidido hacer de la siguiente manera: 1 servo en el tobillo que movería el pie de lado a lado. Después 1 servo en la rodilla, que movería el tobillo y el pie también de lado a lado, no es una rodilla como la articulación de los humanos, sino que más bien como la de una araña. Por último, 2 servos en la cadera, uno que permite mover toda la pierna de arriba abajo, y otro para poder girar la cadera. En la siguiente imagen (Ilustración 13) se puede ver mejor de lo que estamos hablando.

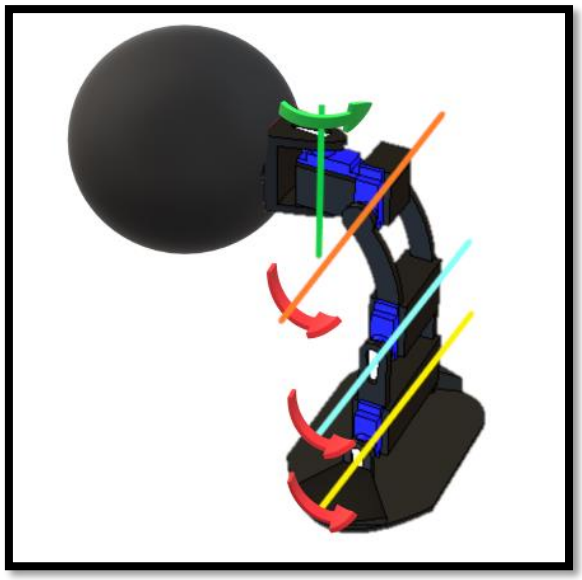


Ilustración 13. Ejes y giros de las articulaciones de la pierna izquierda del robot diseñado.

Esta imagen es la de la pierna izquierda del robot diseñado. La bola negra simula lo que sería el cuerpo del robot, y cada línea representa eje de giro de los distintos servos y articulaciones, las flechas muestran la dirección del giro, 3 de los servos giran de la misma manera, y el cuarto de otra distinta, perpendicular a los otros giros.

2.2.2.- Electrónica

- **Wemos ESP32 D1 R32**: la elección de esta placa en lugar de la Arduino Uno R3 ha sido mayormente debido a las mejores prestaciones que tenía esta, pero ambas hubiesen funcionado correctamente para el proyecto. Esta placa ya tiene el WiFi y el Bluetooth integrado, cosa que la de Arduino no, por lo que no hace falta comprar un módulo Bluetooth por separado. Además, el número de pines y el procesador de esta placa es ligeramente superior también, por lo que esta opción era la mejor para el proyecto.
  En la siguiente tabla (Tabla 1) podemos ver las especificaciones y limitaciones de la Wemos ESP32 D1 R32.

Especificaciones técnicas Wemos ESP32 D1 R32	
Tensión de alimentación	5 – 12 VDC
Corriente de funcionamiento	20 mA (media)
Corriente	250 mA (max)
Velocidad de Reloj	240 MHz
Pines Digitales (3.3 V)	20 entradas/salidas
Pines Analógicos	6 entradas
Memoria	520 Kb
Tamaño	68x53 mm

Tabla 1. Características generales de la placa Wemos ESP32 D1 R32. [6]

- Servomotor SG90: puesto que la estructura del robot no va a ser muy grande, utilizar otros servos como el MG995 sería demasiado. Obviamente funcionaría de igual manera, incluso sin preocuparse de quedarse sin fuerza, pero encarecería el diseño además de tener que adquirir unos controladores de servos adicionales. Los servos SG90 son suficientes con su par de 1.2 Kg·cm a 4.8 V. El diseño de las piezas se a hecho teniendo en cuenta las dimensiones de estos servos, y la simulación en CoppeliaSim también se ha realizado con la fuerza par máxima que pueden dar estos servos. En la siguiente tabla (Tabla 2) se pueden ver las especificaciones del servo SG90.

Especificaciones técnicas Servomotor SG90	
Tensión de alimentación	4 – 7.2 VDC
Velocidad	0.12 seg/60º
Par (4.8 V)	1.2 kg·cm
Peso total	10.6 g
Dimensiones (L x W x H)	22.0 x 11.5 x 27 mm
Giro máximo	180º

Tabla 2. Características generales Servomotor SG90. [8]

- Baterías Li-ion 18650: esta es la mejor opción sobre todo debido al precio, tamaño y peso de las baterías Lipo. En combinación con el soporte para pilas, conseguimos recargar las pilas de manera muy sencilla, detalle que en las baterías Lipo necesitas un adaptador especial. En la siguiente tabla (Tabla 3) tenemos las características de estas baterías.

Especificaciones técnicas Batería Li-ion 18650	
Voltaje	3.7 V
Tipo de batería	Li-ion
Capacidad	2800 mAh (generalmente)
Peso	70 g

Tabla 3. Características generales Batería Li-ion 18650. [11]

- Placa expansora de entradas y salidas: para poder realizar las conexiones necesarias de la placa Wemos a el resto de los componentes, se necesita de esta placa expansora para tener un acceso más sencillo a los pines de entrada y salida. En este caso, la placa Wemos utilizada es compatible con el shield de Arduino UNO. A continuación se muestran las características de esta placa expansora (Tabla 4).

Especificaciones técnicas Placa expansora	
Pines digitales	14 entradas/salidas
Pines analógicos	6 salidas
Compatibilidad	Arduino UNO y similares
Tensión de funcionamiento	5 V suministrados por la placa

Tabla 4. Características generales Placa expansora de pines. [12]

- Sensor Ultrasonidos HC-SR04: para la simulación del robot también se ha utilizado un sensor ultrasonidos con el fin de añadirle opciones al robot, ya sea la detección de obstáculos o aplicaciones similares. En la siguiente tabla (Tabla 5) encontramos las características de este sensor.

Especificaciones técnicas Sensor Ultrasonidos HC-SR04	
Tensión de alimentación	5 Vcc
Frecuencia de trabajo	40 KHz
Alcance máximo	4.5 m
Alcance mínimo	1.7 cm
Duración mínima pulso de disparo	10 µS
Duración pulso eco de salida	100-25000 µS
Tiempo mínimo de espera entre medidas	20 mS
Dimensiones	43 x 20 x 17 mm

Tabla 5. Características generales Sensor Ultrasonidos HC-SR04. [13]

2.3.- Herramientas utilizadas

2.3.1.- SolidWorks

Se trata de un software CAD, es decir, de diseño asistido por ordenador, para modelado mecánico en 2D y 3D. Su primera versión fue lanzada en 1995 pero ha ido y sigue actualizándose, incorporando nuevas opciones, haciendo más fácil su interfaz y mejorando sus algoritmos de cálculos.

Sirve para modelar y crear piezas en 3D desde cero, de las cuáles puedes sacar posteriormente sus planos técnicos y demás información necesaria para la producción.

Las soluciones de simulación para SolidWorks proporcionan una variedad de herramientas con las que analizar fácilmente y predecir el comportamiento físico del producto en condiciones reales.

2.3.1.1.- Interfaz

Al iniciar de primera el programa, aparece esta ventana (ilustración 11) en la que podemos crear una nueva pieza, ensamblaje o dibujo, o si ya disponemos de un archivo creado, a la derecha podemos darle a la carpeta “abrir” y seleccionar nuestro archivo para continuar su trabajo.

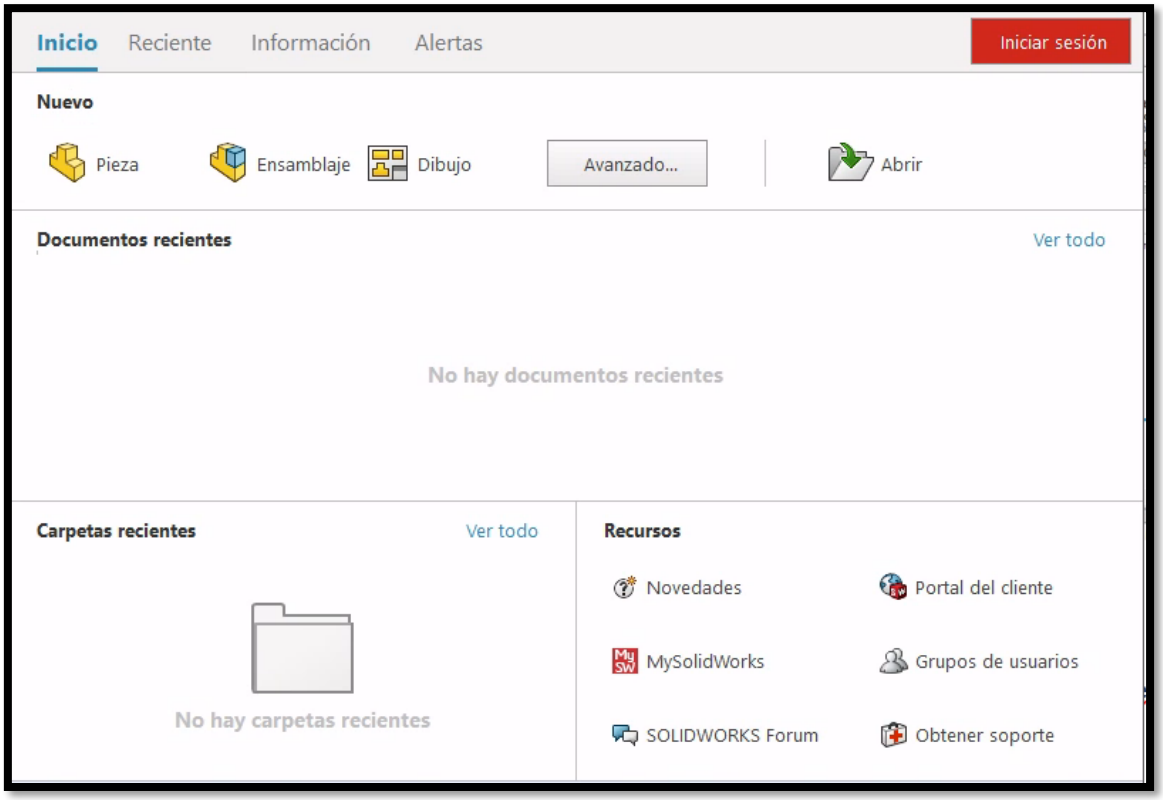


Ilustración 14. Ventana inicial del programa SolidWorks.

Como he mencionado anteriormente, podemos crear 3 tipos de archivo distintos en función de lo que busquemos:

- Pieza: seleccionaremos esta opción cuando queramos crear y representar en 3D una única pieza o componente.
- Ensamblaje: nos abrirá un escenario o plano vacío en el que podremos insertar distintas piezas en 3D y así proceder a su ensamblaje.
- Dibujo: siempre que queramos crear un dibujo técnico o plano en 2D de una pieza o ensamblaje ya existentes, podemos seleccionar esta opción.

Si por ejemplo queremos crear una pieza en 3D de nuestro robot, seleccionaremos donde pone "pieza", y nos aparecerá la siguiente ventana sobre la que trabajaremos (Ilustración 12).

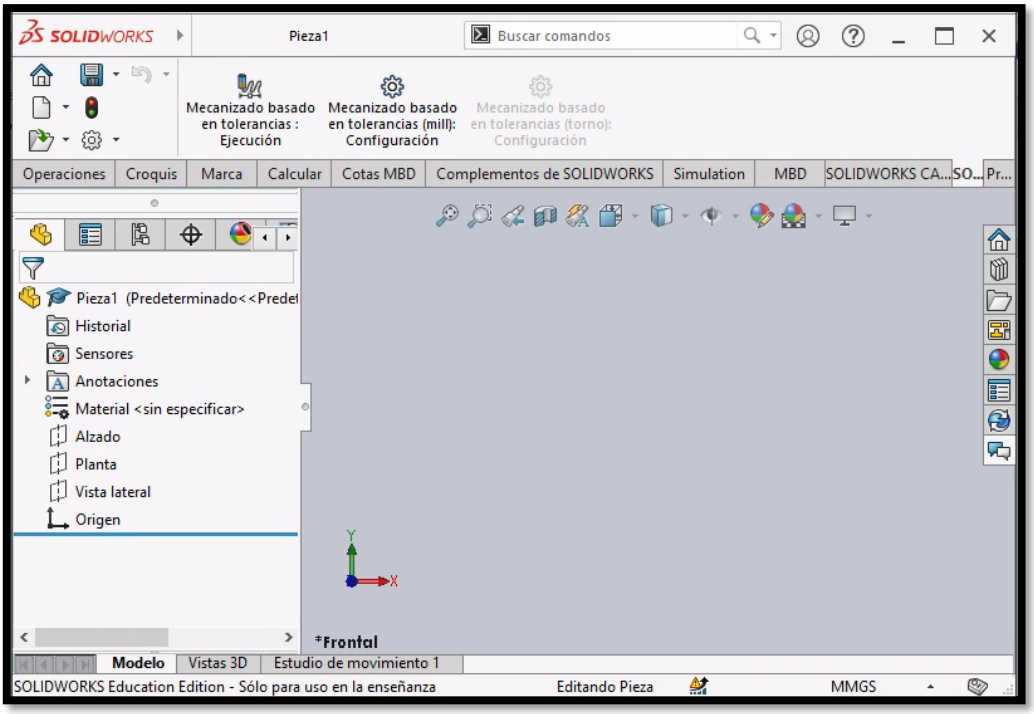


Ilustración 15. Escenario vacío en el que podemos empezar a crear nuestra pieza con un sistema de coordenadas cartesiano.

Se trata de un escenario vacío, con un sistema de coordenadas cartesiano. Lo primero que tendríamos que hacer es seleccionar el plano sobre el que queremos dibujar, en nuestro caso la "Planta", irnos a la opción de "Croquis" y seleccionar el icono de la "línea" para empezar a dibujar. En la siguiente imagen se ve el proceso (lustración 13).

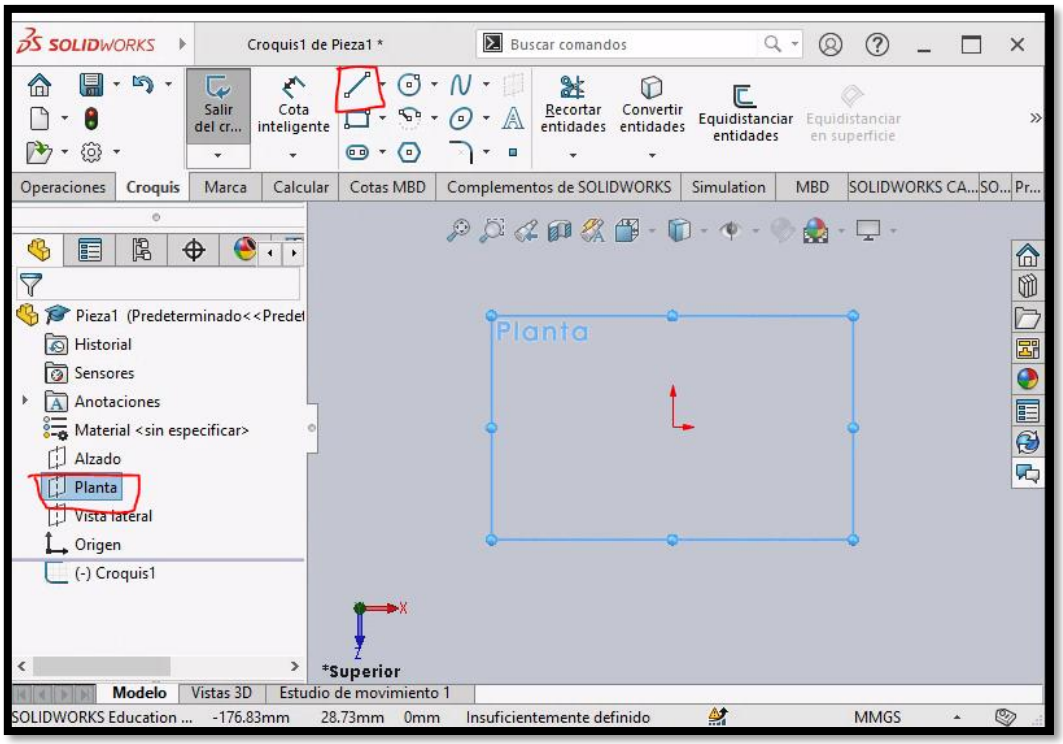


Ilustración 16. Seleccionando la Planta a la izquierda de la ventana, y posteriormente en Croquis la opción línea, podemos empezar a dibujar.

En la barra de herramientas superior hay muchas opciones para dibujar. Podemos utilizar las líneas, círculos, “splines” que son líneas curvas y otros polinomios.

Una vez seleccionado la “línea”, dibujamos la base de nuestra pieza sobre el plano “ Planta” de la forma en la que vemos en la siguiente imagen (Ilustración 14).

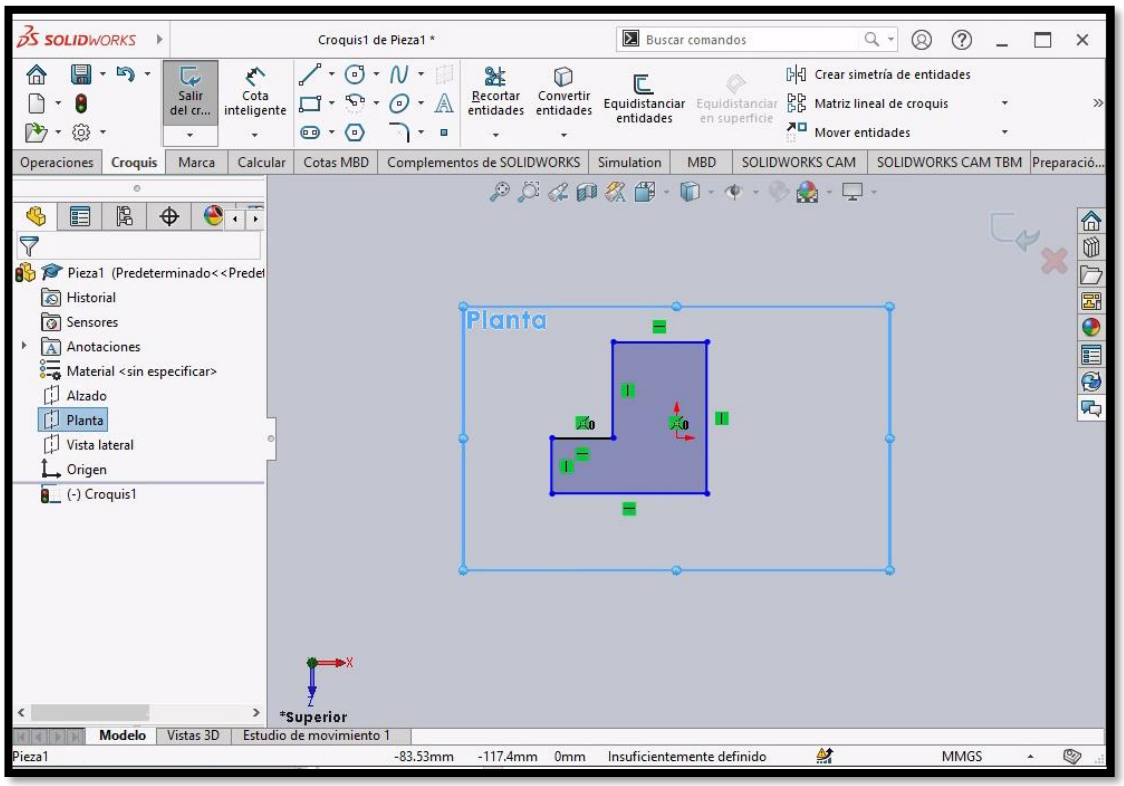


Ilustración 17. Figura plana dibujada con líneas sobre el plano "Planta".

Ya tenemos nuestra forma en 2D de la figura que queremos hacer en 3D. Ahora tenemos que pasar de algo plano a algo con volumen. Seleccionamos el croquis del dibujo, y nos vamos a la barra superior a la opción “operaciones”.

Este apartado contiene una gran variedad de herramientas con las que dar forma a nuestra pieza. Aquí elegimos la herramienta “Extruir” que nos permite dar volumen al dibujo y le indicamos cuanto queremos que mida la extrusión, por ejemplo “50 mm” y aceptamos la operación. Todo este se ve mejor en la imagen inferior (Ilustración 15).

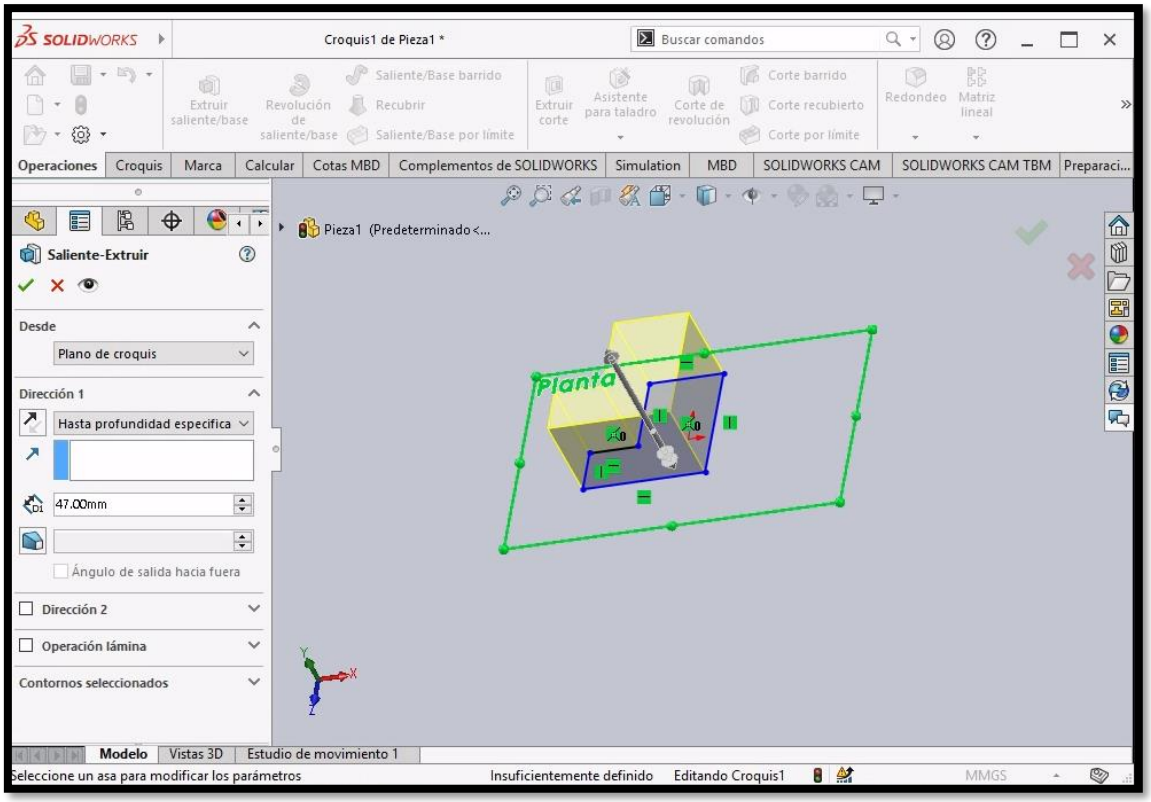


Ilustración 18. Extrusión de una figura plana para convertirla en una pieza con volumen. En la barra lateral elegimos lo que medirá la extrusión.

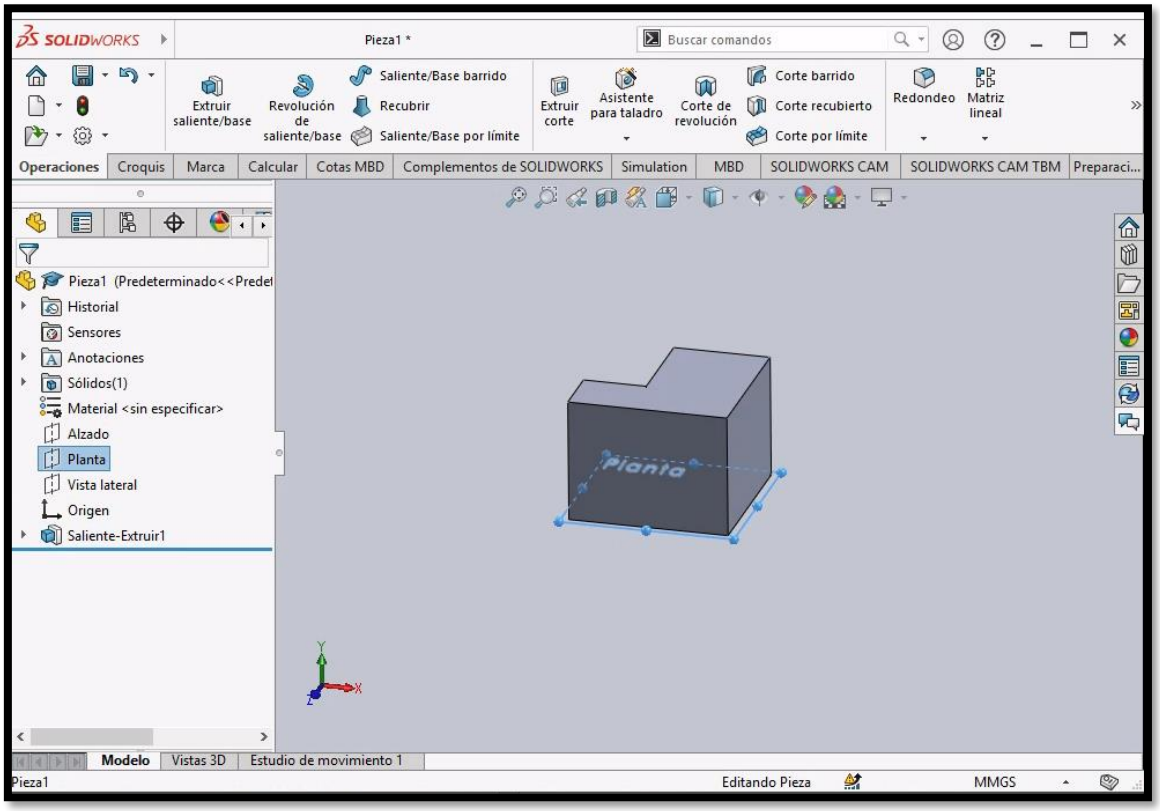


Ilustración 19. Resultado final del dibujo en 3D de una pieza en Solidworks.

Por último, como muestra esta imagen (Ilustración 16), este es el resultado final de nuestra pieza en 3D. Para diseñar todas las piezas del robot se han seguido estos procedimientos, obviamente editando más las formas finales y utilizando muchas de las opciones que dispone este programa.

2.3.2.- CoppeliaSim (V-REP)

El simulador de robots CoppeliaSim es uno de los programas más utilizados para la creación y simulación de cualquier robot. Posee un entorno de desarrollo integrado, basado en un control de arquitectura, digamos que cada objeto o modelo puede ser controlado individualmente a través de "scripts", los conocemos como secuencias de comandos que no se compilan a código máquina, sino que los ejecuta e interpreta el propio programa.

Además de los "scripts", se pueden controlar con "plugins" y demás opciones que nos da el programa. Esto hace que CoppeliaSim sea una herramienta muy versátil con la que trabajar e ideal para aplicaciones de robots.

Los controladores también tienen varios lenguajes que reconocen, pueden ser escritos utilizando C/C++, uno de los lenguajes más utilizados en robótica y que dispone de muchas librerías. Además, puedes utilizar Python, Java, Lua, Matlab y Octave.

En general, CoppeliaSim es utilizado para desarrollo de algoritmos rápidos, simulaciones y automatizaciones, muy importante también en las verificaciones de los prototipos robóticos relacionados con la educación y muchas aplicaciones más.

2.3.2.1.- Interfaz

CoppeliaSim dispone de una versión totalmente gratuita para todos que contiene todas las opciones de simulación, pero le faltan algunas de edición. Después una versión educativa con todo disponible pero solo para estudiantes y profesores de escuelas y universidades. Por último, una licencia pro con todo disponible y que también puedes utilizar para el uso comercial.

Dicho esto, al iniciar el programa se abre de forma predeterminada una escena (Ilustración 17). Este será el escenario sobre el que se trabajará, y además a los laterales y en la parte superior de la ventana nos aparecerán diversas herramientas y opciones que vamos a explicar de manera sencilla y general a continuación.

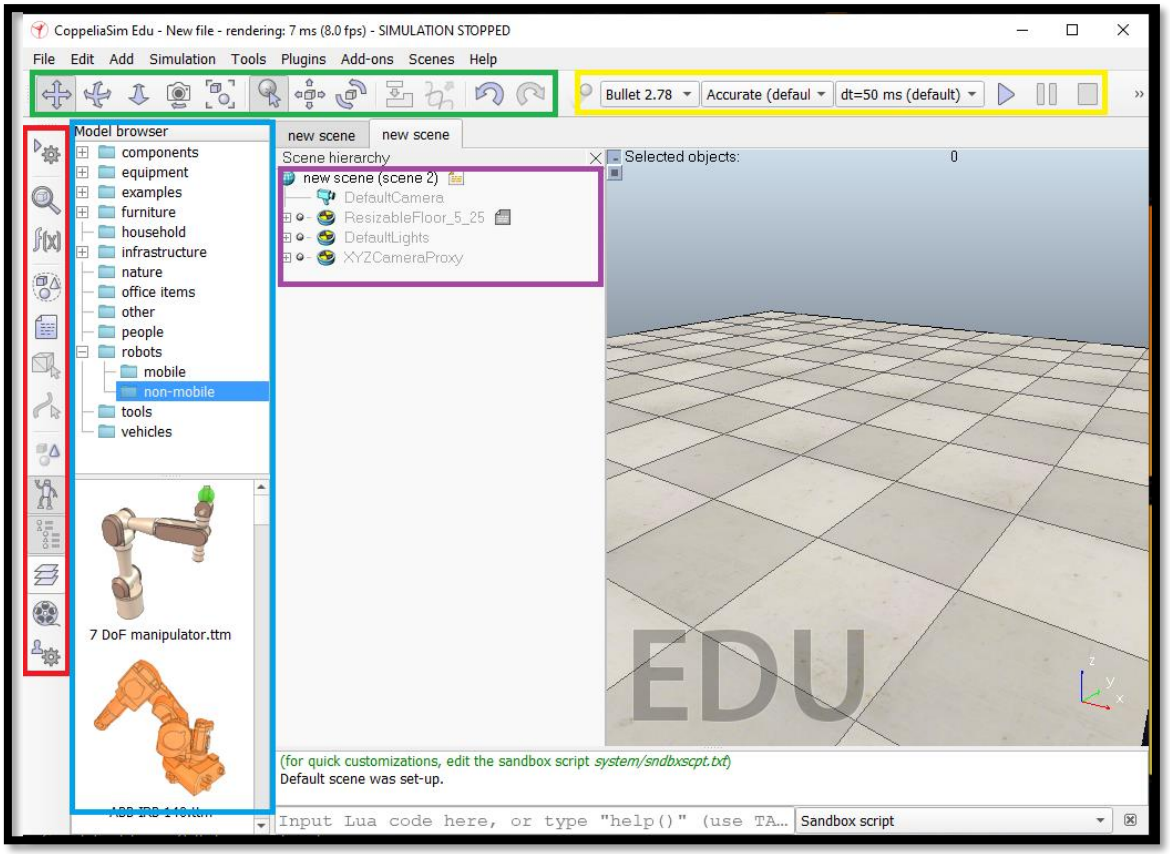


Ilustración 20. Interfaz de un escenario vacío en el programa CoppeliaSim.

- **Escenario:** es lo que podemos ver en la parte derecha de la ventana. Aquí es donde "soltaremos" nuestras piezas o robots para simular lo que se desee con ellos. Dispone de un eje de coordenadas cartesianas para saber en todo momento la orientación y coordenadas de cada uno de los elementos de la escena.

- **Barra de herramientas 1:** se encuentra en la parte superior de la ventana, en la Ilustración 17 está recuadrada en verde. De izquierda a derecha tenemos varios botones con los que nos podemos desplazar por la escena, además de con el ratón. También encontramos los botones de translación y rotación de objetos. Con estos, podemos mover y rotar los modelos a nuestro antojo. Y por último unas flechas para retroceder o avanzar en caso de que queramos corregir algo que hemos hecho recientemente.

- **Barra de herramientas 2:** situada verticalmente en el lateral izquierdo de la ventana, y recuadrada en rojo. De arriba abajo lo primero que encontramos es un botón para cambiar los parámetros de simulación, la frecuencia principalmente. Después otro nos sirve para ver las propiedades dinámicas de los objetos. El que tiene forma de papel es uno de los más importantes, con éste accedemos a los "scripts", que como he mencionado a lo largo del proyecto es donde se escribe el código para programar. Por último, otra opción a conocer es la antepenúltima, esas capas superpuestas. Como dice su nombre son las diferentes "layers" en las que estamos trabajando. CoppeliaSim permite trabajar en distintas capas para ocultar elementos que no queremos ver o nos molestan, pero sin eliminarlos de la escena.

- **Buscados de modelos:** recuadrado en azul, es una ventana en la que podemos arrastrar y soltar modelos ya creados de robots a la escena y trabajar con ellos, así como introducir tus propias "shapes" y modelos que hayas diseñado. En este caso para el proyecto se ha realizado esa segunda opción.

- **Escena de jerarquía:** identificado en la imagen con un recuadro morado, se trata de una ventana en la que se nos indica todos los elementos que se han introducido en el escenario. A la hora de crear jerarquías padre/hijo, para por ejemplo que cuando se mueva la pieza "padre", las piezas "hijo" le sigan, tendremos que usar esta ventana metiendo cada elemento "hijo" dentro del "padre". Además, desde esa ventana podemos acceder a las propiedades de cada objeto, así como modificar las dimensiones y propiedades de la escena.

- **Mando de simulación:** con esta barra en un recuadro amarillo, podemos controlar también los parámetros de tiempo de la simulación y así como pausar o reanudar esta.

2.3.3.- Cura (impresión 3D)

Es una de las aplicaciones más utilizadas a la hora de impresión de piezas en 3D. Este programa cuenta con una configuración recomendada con la que puedes imprimir el modelo 3D según los parámetros que considera mejor el programa, configurando solo la velocidad y la calidad con la que quieres la impresión.

Por otro lado, tienes la opción de modificar múltiples parámetros de impresión que quizá se adapten mejor a los que buscas, como por ejemplo el número de capas, que indicará la dureza y calidad de la impresión, el número soportes con los que quieres que se imprima para después separar la pieza más fácilmente, etc.

2.3.3.1.- Interfaz

En la siguiente imagen (Ilustración 18) se puede ver un ejemplo de un archivo abierto con el programa Cura y las muchas opciones que te dan a modificar en la parte derecha de la ventana.

En este caso la pieza se trata de la cadera diseñada para el robot del proyecto, es una vista de sección para poder ver el interior de la pieza. El programa la abre en una posición que no nos convenía por lo que con la barra lateral izquierda hemos movido y rotado la pieza como queríamos. En la parte derecha se modifican los parámetros de calidad, perímetro, relleno, etc. En rojo son las paredes externas de la pieza, en amarillo el relleno interno que habrá y en azul los soportes que imprimirá y después se quitarán.

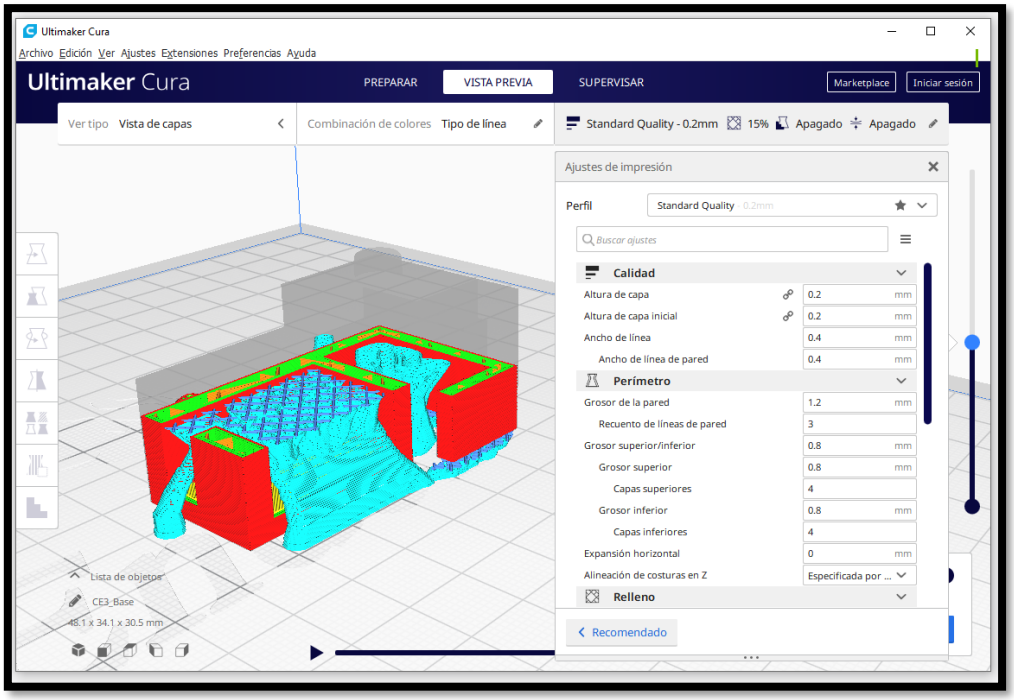


Ilustración 21. Archivo de impresión de la cadera del robot en el programa Cura.

2.4.- Diseño del robot (SolidWorks)

El diseño de cada una de las piezas del robot se ha realizado con el programa SolidWorks como se ha dicho a lo largo de la memoria. Este programa es muy completo para el diseño en 2D y 3D y también para ensamblajes y creación de planos. Ahora vamos a ver cada una de las piezas que componen el robot y por qué decidí hacerlas de esa manera, pero antes, en la siguiente imagen (Ilustración 19) está el diseño y ensamblaje final.

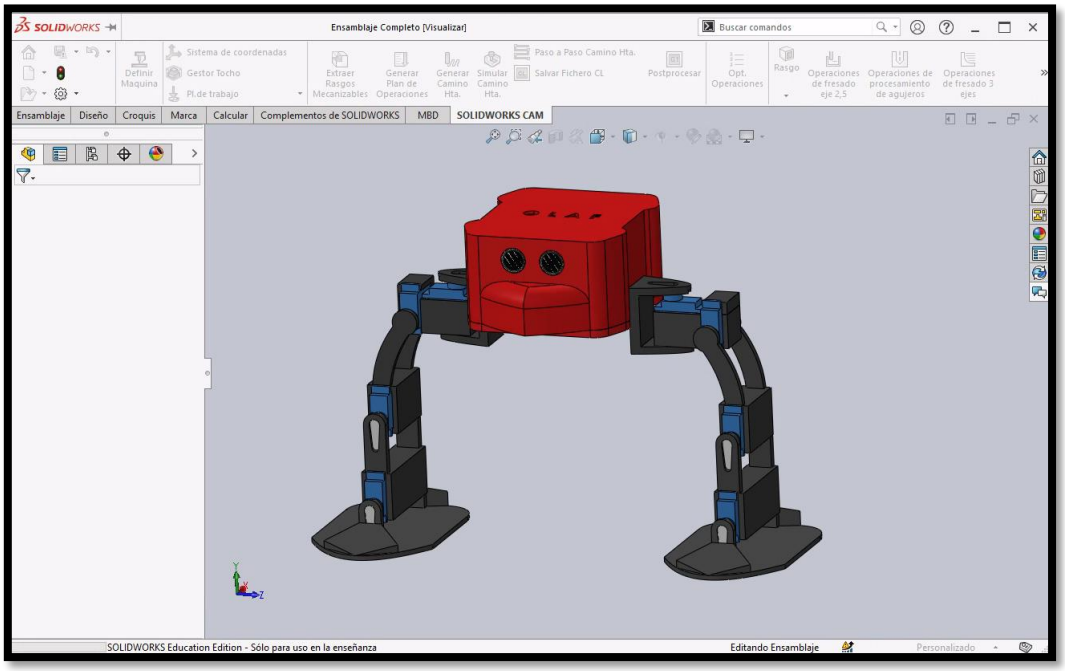


Ilustración 22. Diseño y ensamblaje en SolidWorks del robot.

2.4.1.- Base

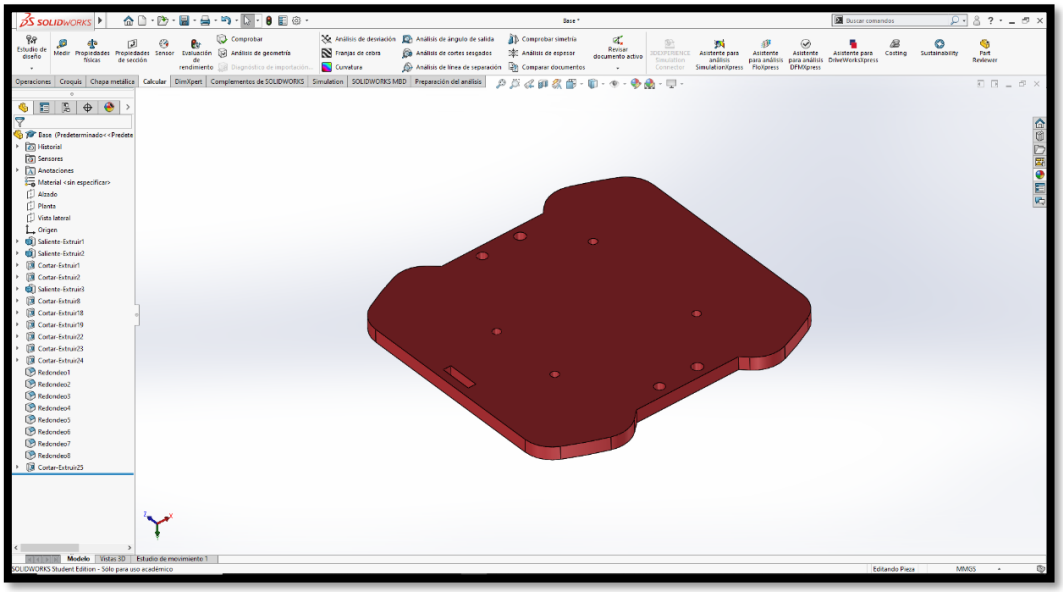


Ilustración 23. Diseño en SolidWorks de la base del robot.

En la imagen superior (Ilustración 20) podemos ver el diseño creado para la base de nuestro robot. La función de esta base es contener en ella la placa Wemos ESP32 D1 R32, así que le hice los 4 agujeros de en medio de la base para poder atornillar la placa. También es donde van a ir sujetas las piernas, por eso los 2 agujeros en cada lado de la placa, donde se atornillarán las 2 piezas "sujeción" que enseñaré más adelante. Por último, tiene una muesca en la parte delantera para que, al poner la cabeza, diseñada para tapar los componentes y sujetar el sensor ultrasonidos, esta no se mueva o se caiga. El diseño de la forma de la base es invención y no pretende seguir ninguna forma específica, era lo menos importante. Las dimensiones lado x lado son aproximadamente 110 x 110 mm.

2.4.2.- Cabeza

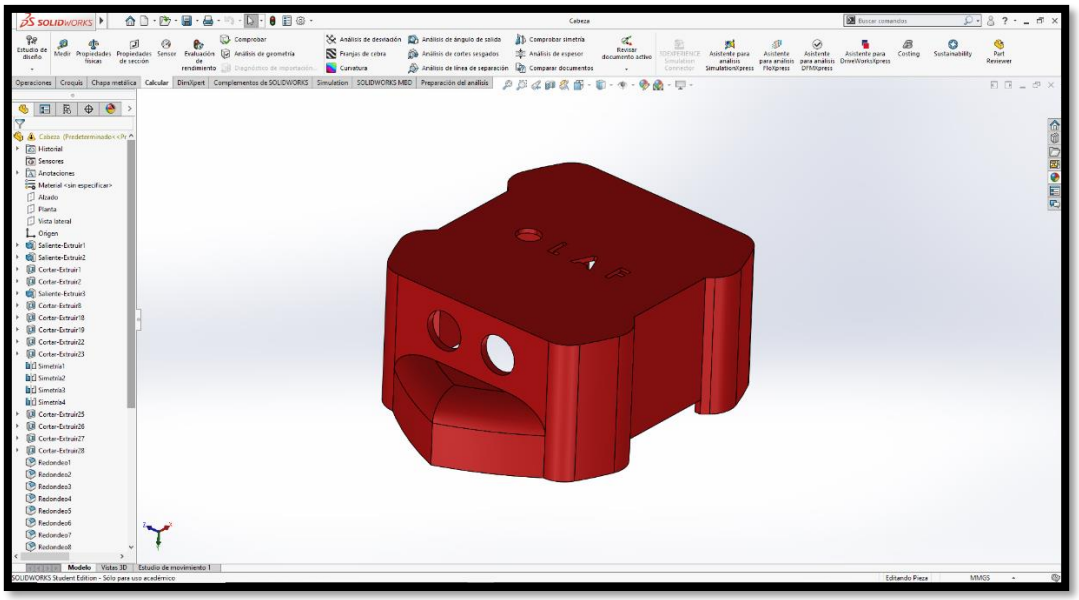


Ilustración 24. Diseño en SolidWorks de la cabeza del robot.

En la ilustración 21 podemos ver el diseño de la cabeza del robot. Tiene dos ojos con las medidas necesarias para que el sensor de Ultrasonidos encaje perfectamente. La base de la cabeza sigue la misma forma que la base del robot.

Las dimensiones son las mismas que la base 110 x 110 mm, y la altura es de 57 mm aproximadamente. La cabeza está hueca por dentro para que el peso del robot sea el mínimo, y además sirve para ocultar la placa y la electrónica utilizada.

2.4.3.- Sujeción

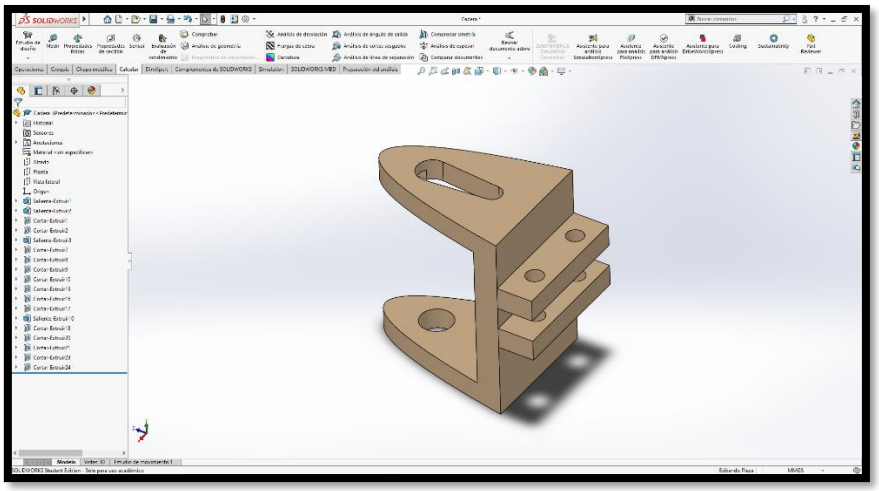


Ilustración 25. Diseño en SolidWorks de la sujeción del robot.

La pieza que vemos en la imagen superior (Ilustración 22) la he llamado sujeción y básicamente se encarga de mantener la cadera y por lo tanto la pierna del robot sujetada a la base. La base se mete entrar esos 2 salientes paralelos y con 2 tornillos nos aseguramos de que quede sujeto. El otro aguajero más grande es donde irá el "boloncho" de la cadera (Saliente redondo de algunas piezas para que giren al encajar en otras), y el hueco superior es donde irá la pieza de plástico del servo. Este montaje se ve mejor volviendo a la ilustración 19.

2.4.4.- Cadera

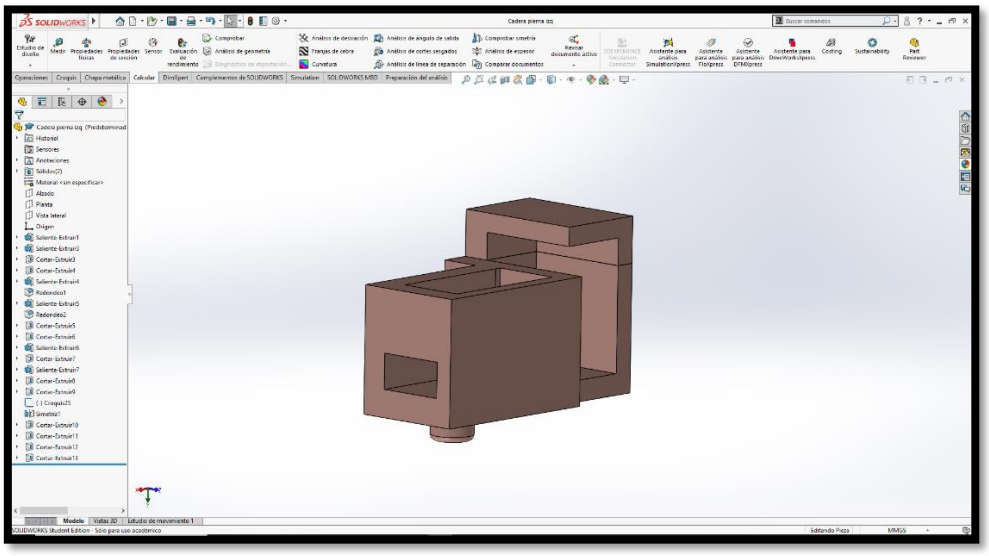


Ilustración 26. Diseño en SolidWorks de la cadera del robot.

La cadera del robot (Ilustración 23) se trata de una pieza con dos compartimentos para los 2 servos que contiene. Uno de ellos está en vertical, para hacer que la pierna suba y baje, y otro en horizontal para que la pierna puedo rotar hacia delante o hacia detrás. Tiene también los huecos necesarios para pasar los cables de los servos, y unos "bolonchos" para encajar la cadera con otras piezas y que tengan rotación. La pieza de la cadera derecha y la cadera izquierda son simétricas.

2.4.5.- Fémur

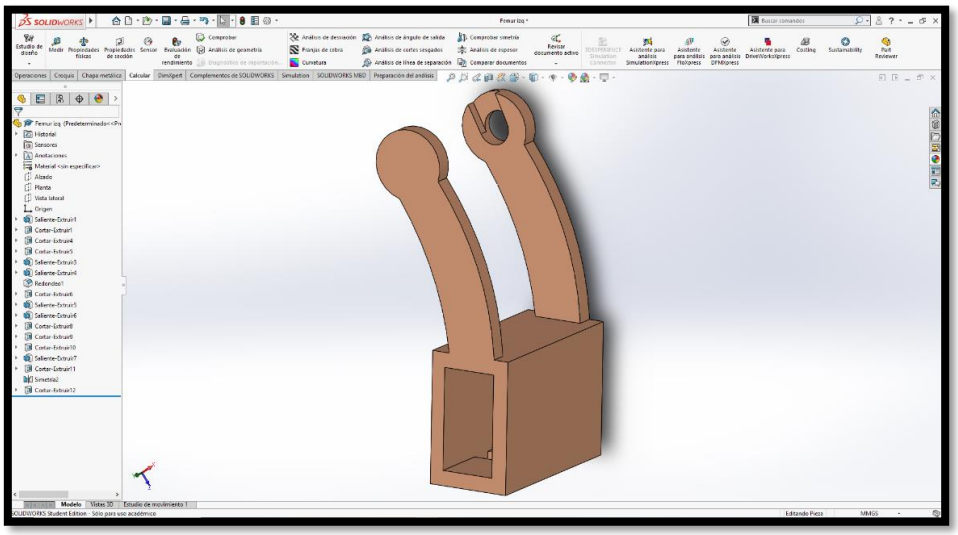


Ilustración 27. Diseño en SolidWorks del fémur del robot.

En esta imagen (Ilustración 25) se puede ver el diseño del fémur de la pierna izquierda del robot, el diseño del de la pierna derecha es el mismo, pero con una simetría vertical, como si fuera su espejo. Tiene el hueco para el servo de un tamaño de 24x11 mm aproximadamente, también el agujero para pasar los cables y un círculo con el que encajar el "boloncho" de la cadera. El diseño es curvado de este modo se reparte la fuerza que tiene que ejercer el servo de la cadera en el eje X e Y.

2.4.6.- Tobillo

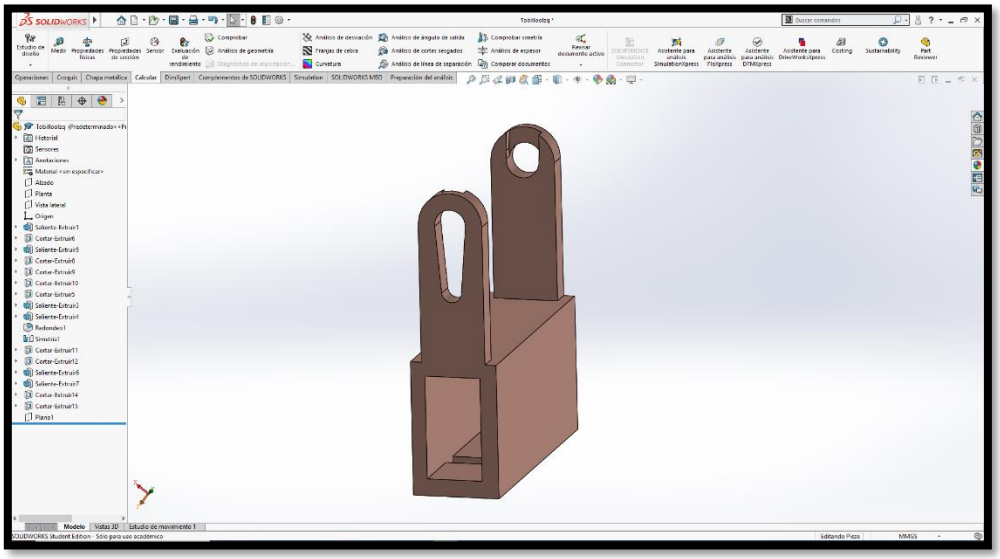


Ilustración 28. Diseño en SolidWorkd del tobillo del robot.

El diseño para el tobillo (Ilustración 26) es similar al fémur, pero en lugar de hacerlo curvado es recto, la fuerza del servo se verá dada en el eje Y. Tiene lo mismo que la otra pieza, espacio para el servo, para los cables y para el "boloncho" que se engancha al fémur. Solo ha hecho falta diseñar la pieza una vez ya que vale para ambas piernas.

2.4.7.- Pie

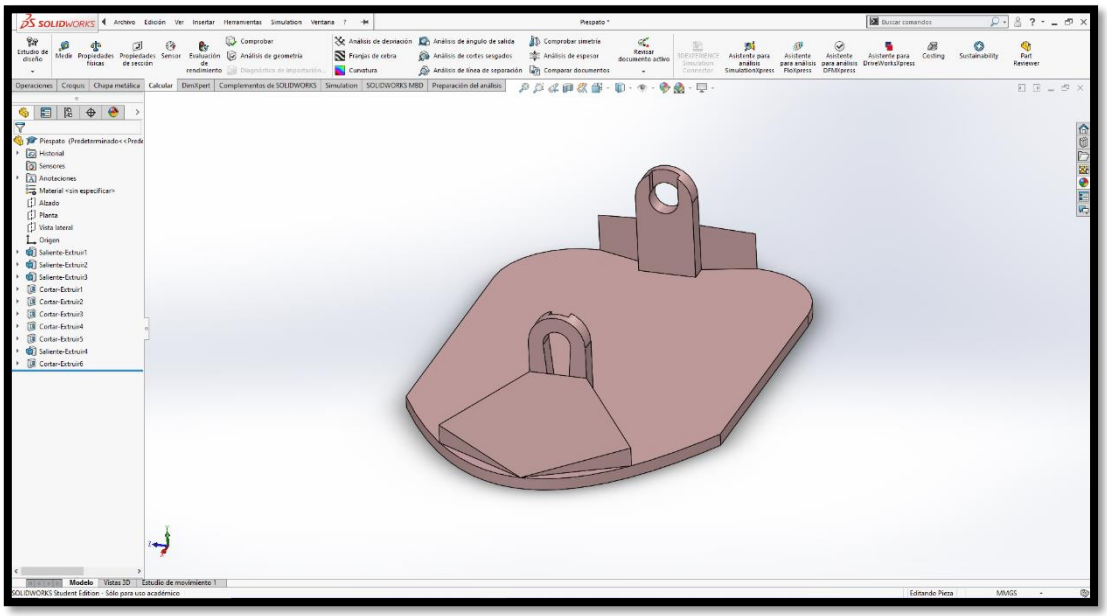


Ilustración 29. Diseño en SolidWorks del pie del robot.

Para el diseño del pie (Ilustración 27) había que pensar qué tamaño hacerlo ya que lo difícil de los robots bípedos es la estabilidad, así que para asegurarme de que el robot tenía suficiente plataforma de apoyo diseñé el pie con una superficie de estas medidas, 80 x 65 mm, que vemos en la siguiente imagen (Ilustración 28).

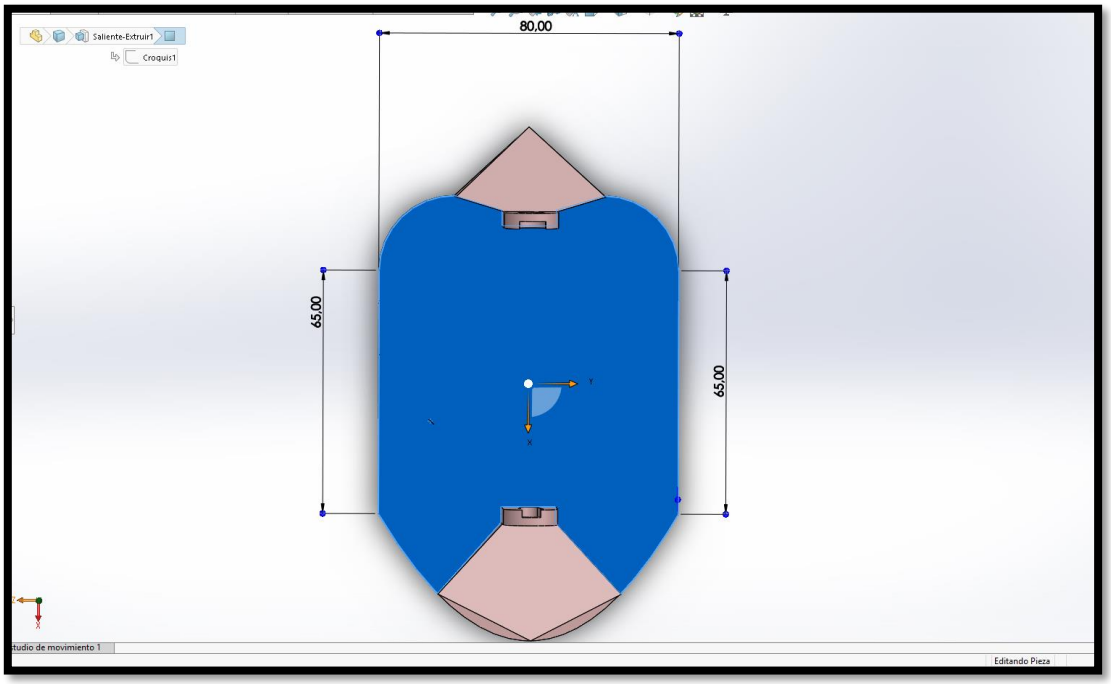


Ilustración 30. Medidas del pie diseñado en SoldWorks.

2.5.- Programación del robot (CoppeliaSim, V-REP)

2.5.1.- Importación de las piezas

Lo primero que hice para empezar con la programación del robot utilizando la herramienta CoppeliaSim fue importar las piezas 3D que había creado en SolidWorks.

Para ello fui a "File – Import – Mesh" y seleccioné todas las piezas que anteriormente había convertido a .stl desde SolidWorks, de este modo se introducen todas ya en el sitio correspondiente y el robot queda ensamblado.

Una vez importadas las piezas procedí a cambiarle el nombre a todas, ya que de serie vienen como "shape" y un número. De este modo las nombraba para saber en todo momento cuál estamos seleccionando, y a su vez le cambié el color para darle al robot un aspecto distinto y poder diferenciar las piezas y la electrónica.

En la siguiente imagen (Ilustración 29) se ve el resultado final de la importación de las piezas, el cambio de nombre y el cambio de color del robot.

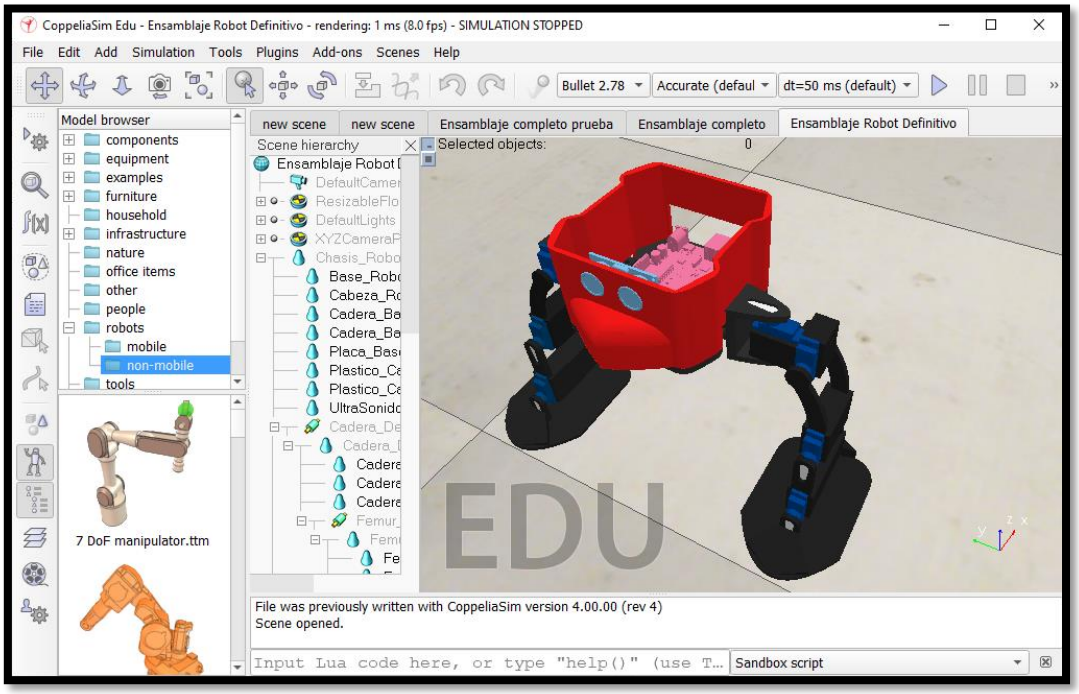


Ilustración 31. Importación, renombramiento y coloreado de los archivos .stl del robot en CoppeliaSim.

2.5.2.- Creación de las *shapes*

Algo importante a saber es que, hasta ahora, lo único que tenemos es el aspecto visual de las piezas del robot, pero estas no tienen propiedades físicas y dinámicas, por lo que para poder programar el robot como si fuera en la vida real, tenemos que asociar y crear para cada una de las piezas, una "shape" similar a ellas con las que sí que podremos trabajar.

Para ello seleccionamos una de las piezas, el pie por ejemplo y en la barra lateral izquierda clicamos en el icono del cubo. Aquí le damos a extraer un cuboide y ya tendríamos una de las "shapes" creadas. Hacemos lo mismo para el resto, en algunas será necesario hacer más de una "shape" como es el ejemplo del tobillo o del fémur.

Una vez creadas todas y viendo que tienen una forma similar a las piezas del robot, agrupamos las que formen una misma pieza y las pasamos todas a una segunda capa, ya que no queremos que sean visibles, solo sirven para que el programa trabaje y opere con ellas.

En la siguiente imagen (Ilustración 30) se ven las shapes creadas para el robot.

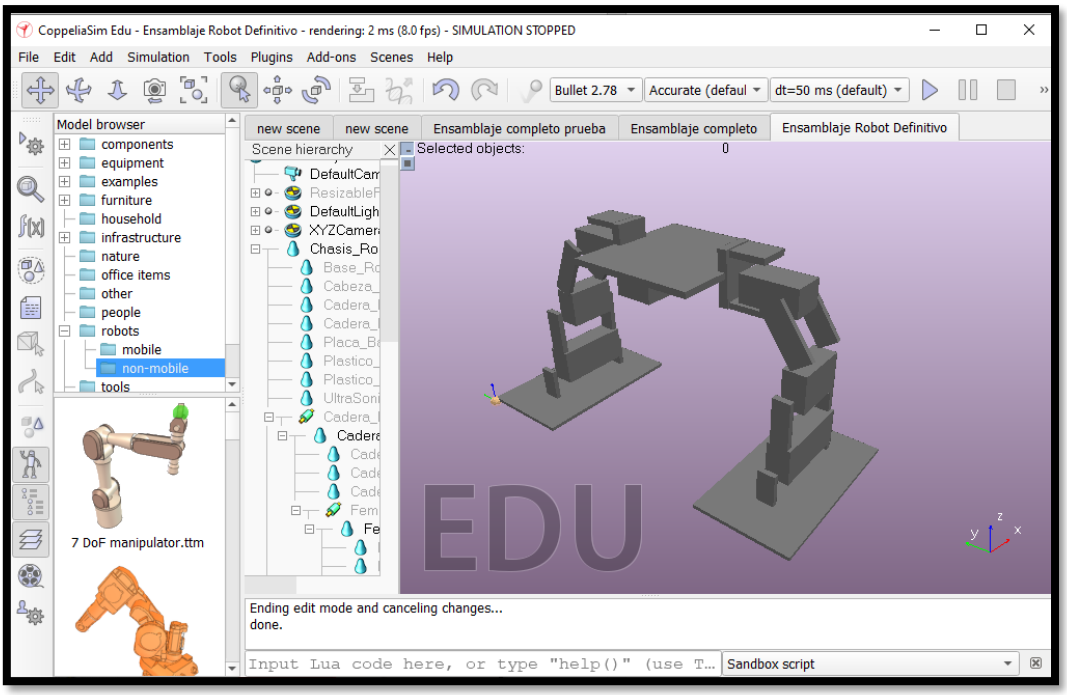


Ilustración 32. Creación de las shapes de cada una de las piezas en CoppeliaSim.

Como podemos observar, no he extraído "shapes" de la placa, o la cabeza, ya que no es algo que me interese y no va a influir apenas en el cálculo.

2.5.3.- Creación de articulaciones y servos

Ahora que hemos creado unas piezas con propiedades físicas y dinámicas, lo siguiente que necesitamos es simular los servos de nuestro robot. Para ello CoppeliaSim tiene una opción para introducir pares de revolución donde queramos, que en nuestro caso sería en cada uno de los servos.

El procedimiento que se ha seguido para todos los servos es el siguiente:

- Vamos a Add – Joint – Revolute, reducimos el tamaño para que sea más pequeño y fino, pues de serie es muy grande.
- Para poder colocar la articulación donde toca, extraemos una "shape" cilíndrica de la parte plástica del servo que gira, esto solo lo queremos para colocar la articulación en ese cilindro (Ilustración 31),
- Seleccionamos la articulación y manteniendo *Shift* clicamos en el cilindro, ahora vamos a la barra superior horizontal de herramientas y le damos al botón de trasladar la primera selección a las coordenadas de la segunda. Rotamos si es necesario el eje de revolución para que quede y gire como deseemos, y ya tendríamos un par de revolución en la posición del servo.

- Hacemos esto para cada uno de los servos y articulaciones. En la siguiente imagen (Ilustración 32) están introducidos todos los pares.

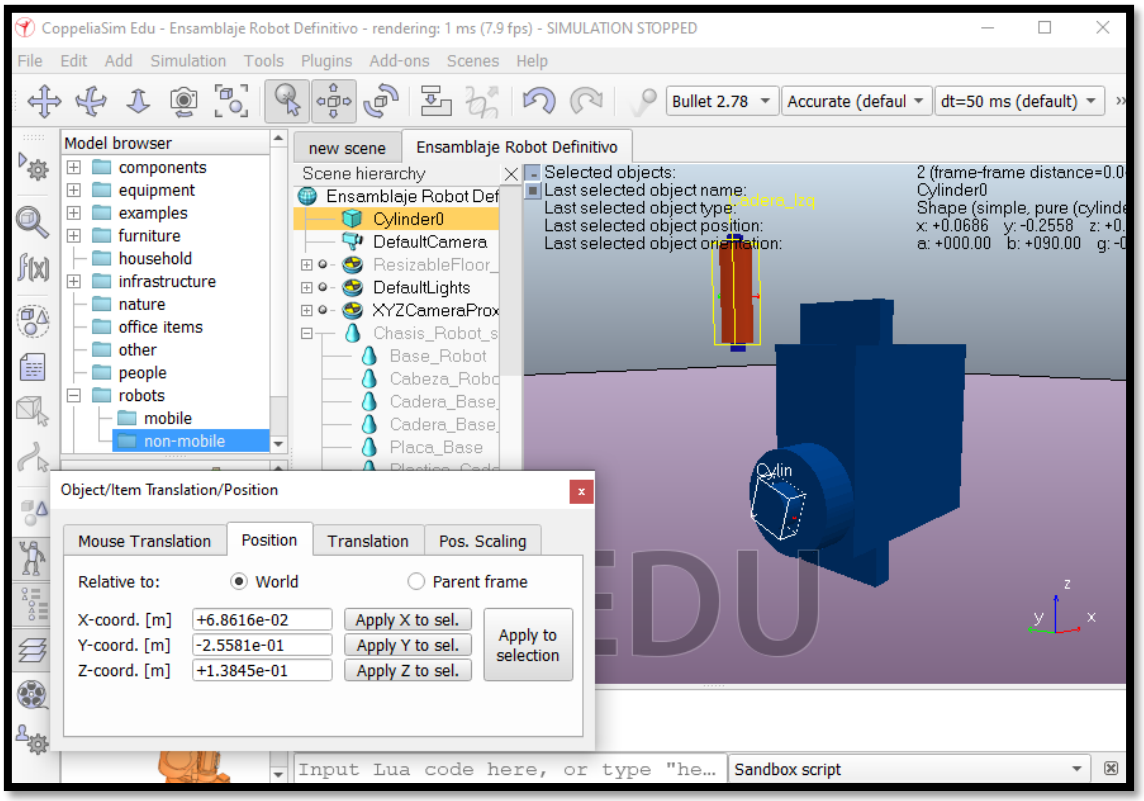


Ilustración 33. Posicionamiento de un eje de revolución sobre el servo en CoppeliaSim..

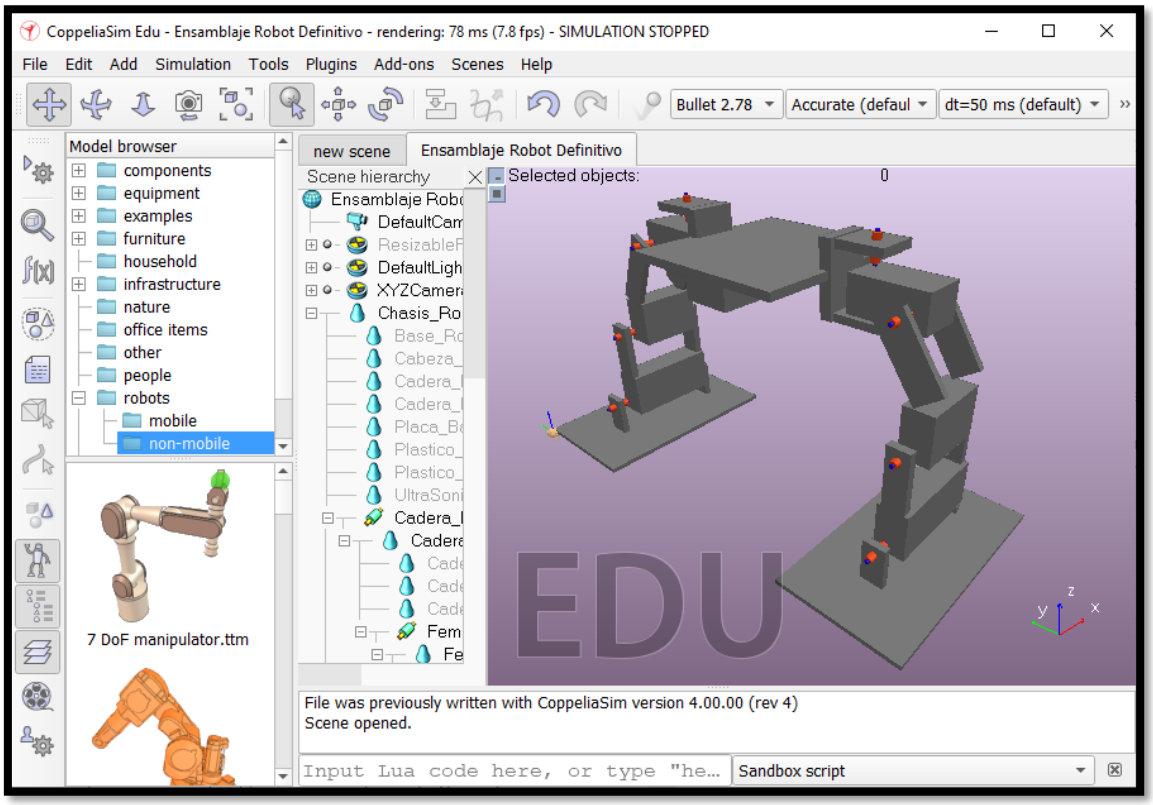


Ilustración 34. Colocación de las articulaciones y pares de revolución de nuestro robot en CoppeliaSim.

2.5.4.- Jerarquía del robot

Los archivos que había introducido y he dicho antes que eran solo el aspecto visual se quedan de pie y montados porque así es como estaban antes de pasarlos a .stl, pero las "shapes" creadas con el programa o los ejes de revolución no siguen ninguna jerarquía ni ninguna relación, si le diésemos a iniciar simulación se caerían. Por lo que ahora hay que crear una jerarquía de todas las "shapes" creadas, los ejes de revolución y las piezas de SolidWorks.

Con ayuda del *Scene hierarchy* vamos arrastrando las piezas que consideremos como "hijos" dentro de las piezas "padre", de esto modo las piezas hijo de moverán y seguirán a sus padres. En el caso del robot, la idea a la hora de programarlo es simular que uno de los pies, por ejemplo el pie derecho primero, es la base del robot, por lo tanto el padre de todos, y desde el pie derecho, subiendo por la pierna, pasando a la base y bajando hasta el pie izquierdo, irán el resto de los padres e hijos. Así pues, pensando que tenemos un brazo robótico, el pie derecho es la base de nuestro brazo, y el pie izquierdo es el "tip" o punta del robot, y este será el que querremos mover a la posición deseada.

En la ilustración 35 vemos la jerarquía creada para el caso mencionado. Posteriormente en la programación del robot modificaremos un poco la estructura y explicaremos por qué.

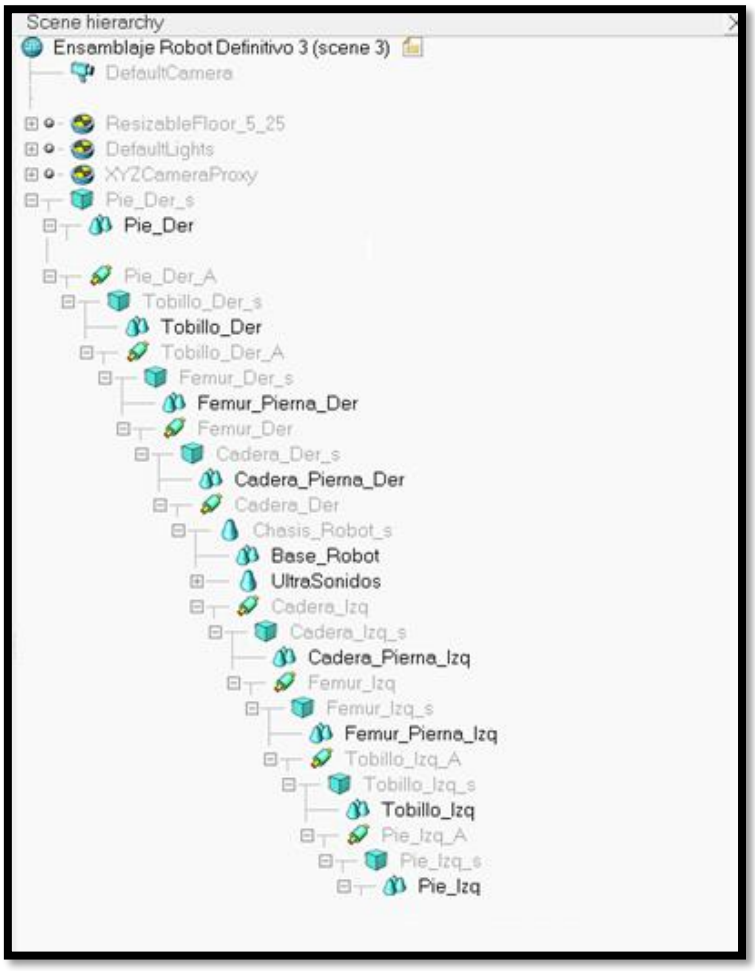


Ilustración 35. Jerarquía de todas las piezas y articulaciones del robot en CoppeliaSim.

2.5.6.- Programación del movimiento

El programa CoppeliaSim dispone de herramientas para el cálculo de la cinemática inversa de los robots, mediante las cuáles tú marcas al robot donde quieres que se mueva, y este te calcula los movimientos que debería hacer y escribe en los servos los ángulos necesarios para alcanzar esa posición.

Partimos de la idea como he mencionado anteriormente, de que nuestro robot bípedo ahora se trata de un brazo robótico, que tiene una base llamada "pie_der_s" y el resto del robot son las articulaciones y eslabones hasta llegar a la punta o *tip* del robot, que le hemos llamado "pie_izq_s".

Lo primero que haremos será añadir 2 *dummies* a nuestro robot, que serán los encargados de crear las cadenas cinemáticas del robot. Para ello vamos a la opción "Add" de la barra superior y seleccionamos *dummy*. Estos objetos siempre van en parejas, ya que utilizaremos una configuración "tip-target" en la que se necesitan tener una pareja de *dummies*.

Llamamos al primero de ellos "Dummy_pie_izq_target" y lo hacemos hijo de la base de nuestro robot (pie_der_s). El segundo de los *dummies* le ponemos el nombre de "Dummy_pie_izq_tip" y lo hacemos hijo de la punta de nuestro robot (pie_izq_s). Como indican los nombres, el primero de ellos se moverá libremente, mejor dicho, por un camino que estableceremos más adelante, y el segundo *dummy* situado en el *tip* seguirá al *target* para poder cerrar la cadena cinemática y resolver la posición y los ángulos que deben escribirse en los servos.

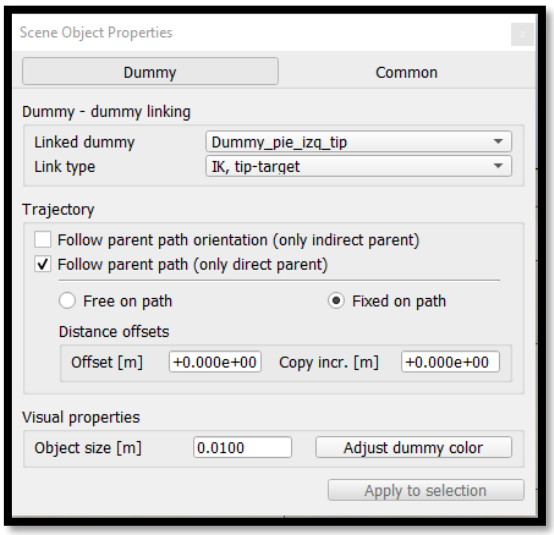


Ilustración 36. Propiedades Dummy target.

Ahora creamos un *path*, en la misma barra de herramientas para crear los *dummies.* Este *path* o camino será el que editaremos y marcaremos que siga el *dummy target* para hacer que nuestro robot se mueva.

Una vez hecho esto, entramos dentro de las propiedades del *dummy target (Ilustración 36)* y arriba seleccionamos la opción "Linked dummy" para unir este con el *tip.* También seleccionamos el tipo de link *"IK, tip-target",* y por último marcamos la casilla de "Follow parent path", para que siga al camino que editaremos tanto en posición como en orientación.

Hacemos los mismo con el otro *dummy, el tip*, pero esta vez vinculándolo con el otro, y en lugar de marcar la casilla de "Follow paren path", la dejamos desmarcada, ya que este no seguirá el *path* sino solo al *target*.

En la imagen de más abajo (Ilustración 37) se puede ver como ha quedado la jerarquía con la adición de los *dummies* y el *path*.

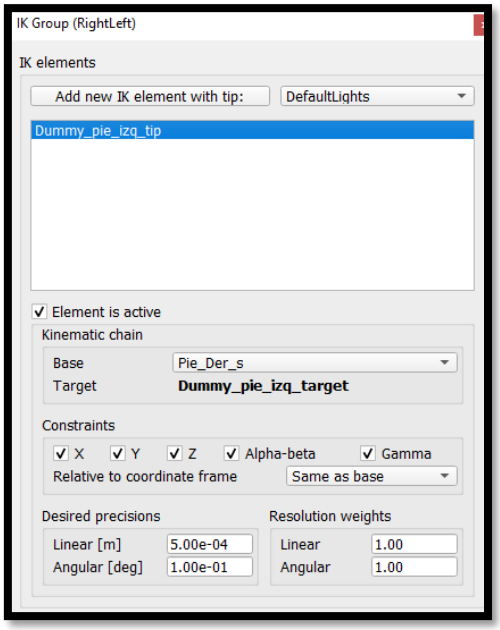


Ilustración 37. Configuración del IK module.

A continuación, una vez tenemos "linkeados" los 2 *dummies* y creado un *path* que después editaremos, tenemos que introducir un módulo de cálculo de la cinemática inversa, ya que sino no se realizarán las operaciones necesarias para el cálculo de los ángulos. Para ello vamos a "Tools – Calculation module properties" y después al partado "Kinematics". Aquí añadimos un nuevo grupo "IK" y le llamamos como queramos. Seleccionamos el método de cálculo "DLS" que, a pesar de ser más lento, tiene mayor precisión, y subimos el máximo de iteraciones a 50, por último el valor "damping" a 0.01.

Esta configuración la vemos en la ilustración 38. Ahora lo siguiente es ir a "Edit IK elements", que se encuentra en la parte inferior de la misma ventana.

En esta nueva ventana añadimos un nuevo "IK element with tip", y seleccionamos al *Dummy_pie_izq_tip.* Automáticamente asociará a este con su target ya creado antes, y ponemos como Base al *Pie_Der_s.* Por último marcamos todas las casillas de abajo para que así el *dummy tip* siga al *target* tanto en posición como en orientación, y así evitar que el pie de un paso y aterrice en el suelo con una orientación no deseada.

Estos últimos pasos los encontramos en la ilustración 39, donde se muestra la ventana sobre la que estábamos hablando.

Ahora lo siguiente a hacer es modificar ese *path* que habíamos creado anteriormente. Basta con irnos a sus propiedades, y añadir y mover tantos puntos como deseemos que

Ilustración 39. Jerarquía y adición del Path y Dummies.

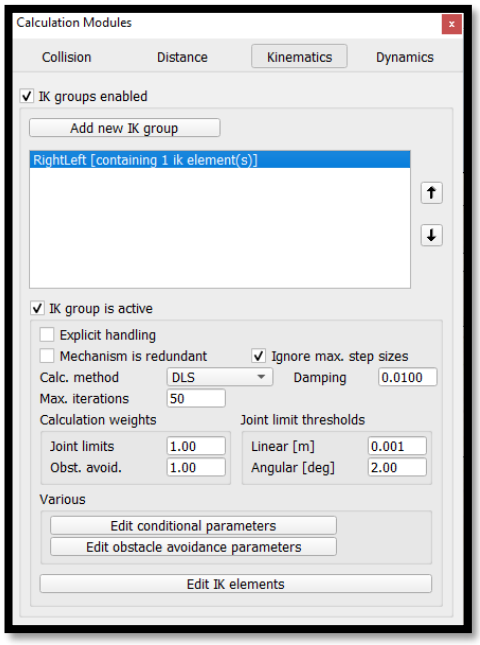


Ilustración 38. Configuración del IK element.

contenga nuestro camino, y dar la forma y trayectoria que queramos. Además es importante ver que la posición y orientación que el *dummy target* es igual que la de los puntos del *path,* para que así el robot le siga tal y como queremos. En la imagen de arriba (Ilustración 40) vemos cuál es el camino diseñado para que haga el primer paso, simplemente describe una curva hacia el interior del robot, y después acaba sobre el mismo eje, pero un poco más adelante, para así dar un paso en recto.

Por último, para que los *dummies* sepan como seguir al *path*, debemos indicárselo mediante *scripts* y funciones. Para ello añadimos un nuevo script de tipo "thread", y utilizamos funciones tales como *sim.getObjectHandle(),* que nos permite sacar el manejador de un objeto y así poder utilizarlo en todo el *script.* Además, usamos funciones para extraer las propiedades del *path* y así saber en qué posición nos encontramos. Estas son *sim.getPathLength*(), sim.getPositionOnPath() y *sim.getOrientationOnPath()*, como indican sus nombres nos devuelven la longitud del camino, y la posición y orientación en la que está el punto. Con estos valores y las funciones *sim.setObjectPosition()* y sim.setObjectOrientation(), podemos escribir continuamente en el *dummy objetivo* para que siga al *path*.

El código utilizado lo podéis encontrar en el documento "ANEJO 1: Código", y la explicación de este no requiere más de lo ya comentado. Para saber cómo usar las funciones, el programa dispone de una web, en la que están explicadas todas las funciones de las que disponemos.

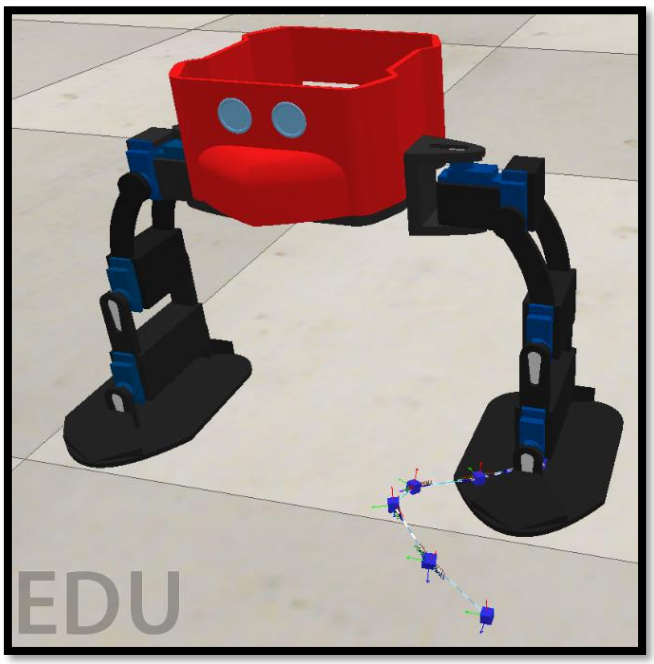


Ilustración 40. Creación de un primer Path para el pie izquierdo.

Al darle a iniciar simulación el robot mueve el pie izquierdo hacia el final del camino, consiguiendo dar un paso y moviendo cada unos de los servos correspondientes. El problema es que la jerarquía es unidireccional, por lo que para mover el pie derecho que es la base, tenemos que hacer que deje de ser la base. Pero no podemos estar cambiando la jerarquía cada vez que queramos mover un pie u otro.

La solución que he encontrado es añadir un cubo cualquiera y hacerlo padre de todo, es decir, metiendo la rama jerárquica que tenemos dentro del cubo, de este modo el pie

derecho dejará de ser la base de todo el robot, tan solo lo será para mover el pie izquierdo, y para cuando toque mover el pie derecho, este se moverá en torno al cubo.

Para que se mueva añadimos 3 ejes prismáticos y los ponemos en perpendicular entre ellos, además en la jerarquía los metemos dentro del cubo y uno dentro de otro. Por debajo del último eje prismático colocamos el resto del robot. En la siguiente imagen (Ilustración 41) se puede ver la nueva jerarquía, con 2 *dummies* nuevos que ahora explicaré. También se muestra los 3 ejes prismáticos creados y el cubo.

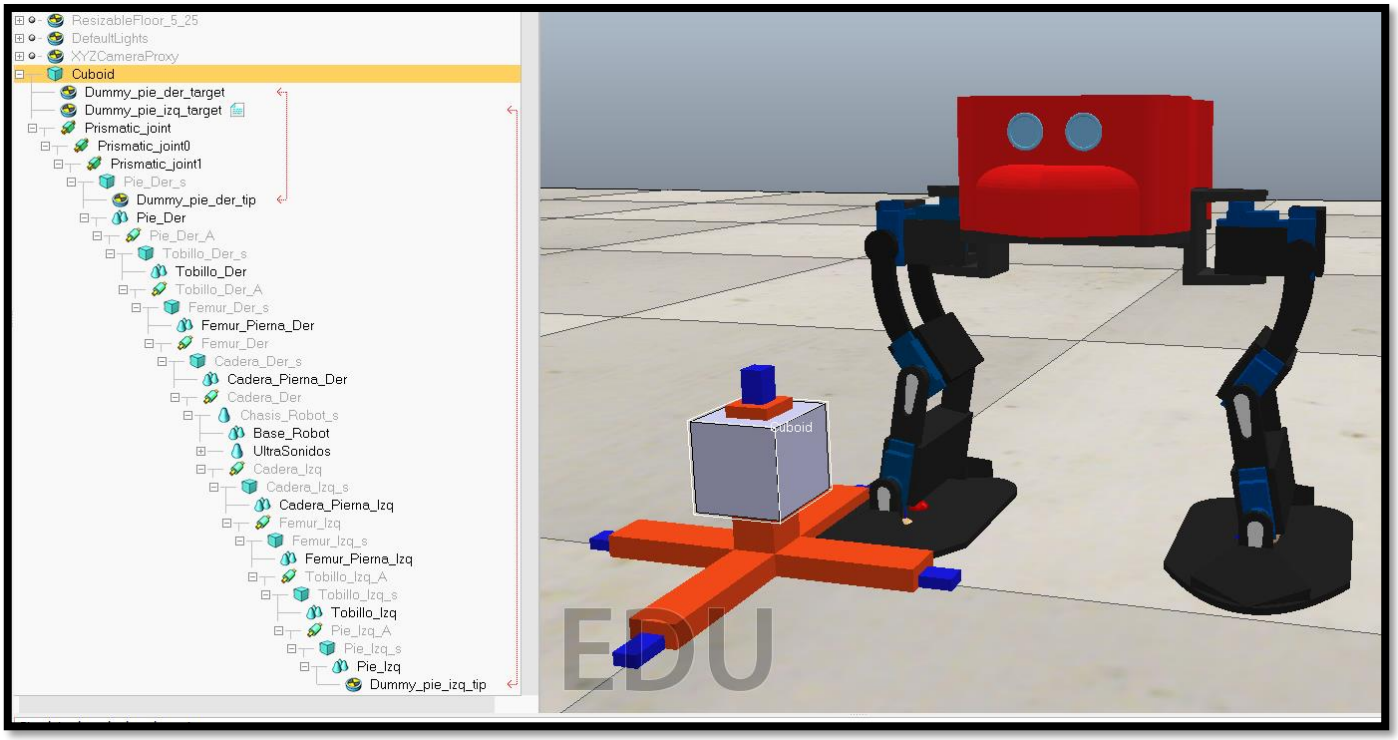


Ilustración 41. Nueva jerarquía para poder mover ambos pies.

Los 2 nuevos dummies introducidos y a los que he llamado *Dummy_pie_der_tip y Dummy_pie_der_target* son la otra pareja necesaria para el movimiento de este pie. Al igual que con los otros *dummies* debemos vincularlos entre ellos y volver a crear un *IK module* para el cálculo de la cinemática inversa. En este caso la base de esta cadena cinemática no será el pie derecho, ni el pie izquierdo por los problemas de jerarquía mencionados anteriormente, esta vez será el cubo creado.

La pareja de *dummies target* los hacemos hijos del cubo, ya que no importa mucho donde estén, pero los *tips* sí que son importantes. El del pie izquierdo lo dejamos hijo de la *shape pie_izq,* y el del pie derecho en la *shape pie_der.*

Con todo esto creado y bien organizado, lo único que nos queda es crear los *Paths* que queramos para que se mueva y dé tantos pasos como deseemos. En mi caso he creado 17 *Paths,* para que el robot de un número determinado de pasos, suba 3 escalones y chute una pelota. El código y las funciones empleadas son las mismas que con el de avanzar una pierna solo. Lo único a cambiar es el definir las variables necesarias para cada camino, e ir saltando de un bucle a otro en función de la altura a la que nos encontremos del *Path,* para así ir moviendo un pie y otro.

En la siguiente imagen (Ilustración 42) se ve el entorno diseñado para la simulación. Se puede diseñar el que quieras siempre y cuando presente movimientos naturales.

La demostración de la simulación se encuentra en un archivo de vídeo entregado junto a esta memoria, además de andar por los *Paths* se ha diseñado un escenario en el que sube unos escalones y golpea una pelota.

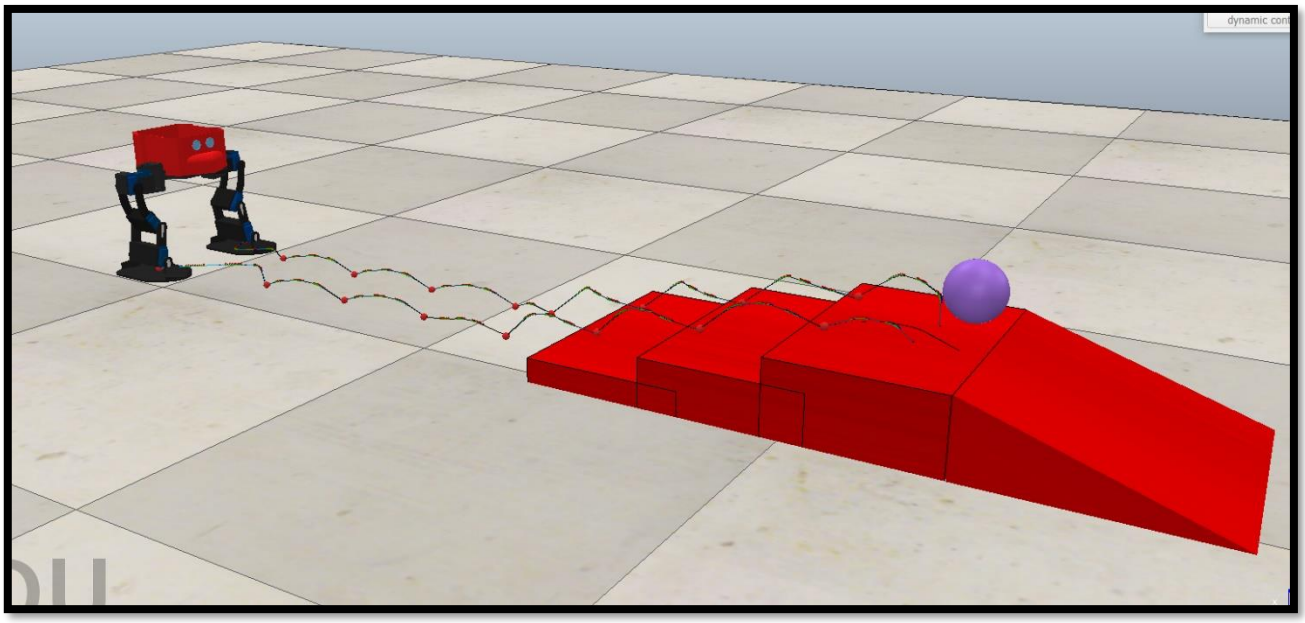


Ilustración 42. Camino diseñado para el robot.

2.6.- Impresión en 3D del prototipo

Para la impresión 3D de este prototipo se ha utilizado la impresora 3D modelo Creality Ender 3 (Ilustración 36). Es una impresora que puedes obtener fácilmente por internet y de montaje sencillo.

Ilustración 43. Impresora Creality Ender 3 utilizada en el proyecto. [14]

El volumen de impresión es de 220 x 220 x 250 mm y emplea extrusión tipo bowden, que empuja el filamento de ABS por el interior de un tubo de teflón hasta donde se funde. La cama de impresión alcanza los 90°C necesarios para imprimir con ABS.

Todas las piezas se han impreso con las mismas características:

- 240°C de temperatura de impresión
- 60 mm/s de velocidad de impresión
- 3 paredes, 3 capas inferiores y 3 capas superiores
- 20% de rellenos de las piezas y en forma de rejilla

Durante la impresión de algunas piezas hubo problemas ya conocidos en el mundo de impresión 3D como es el efecto “cracking”, donde las capas de ABS se separan debido a la diferencia de temperaturas, o también el conocido “warping”, que se produce al separarse la base de la pieza de la cama, por lo que la pieza queda deformada (Ilustración 37).

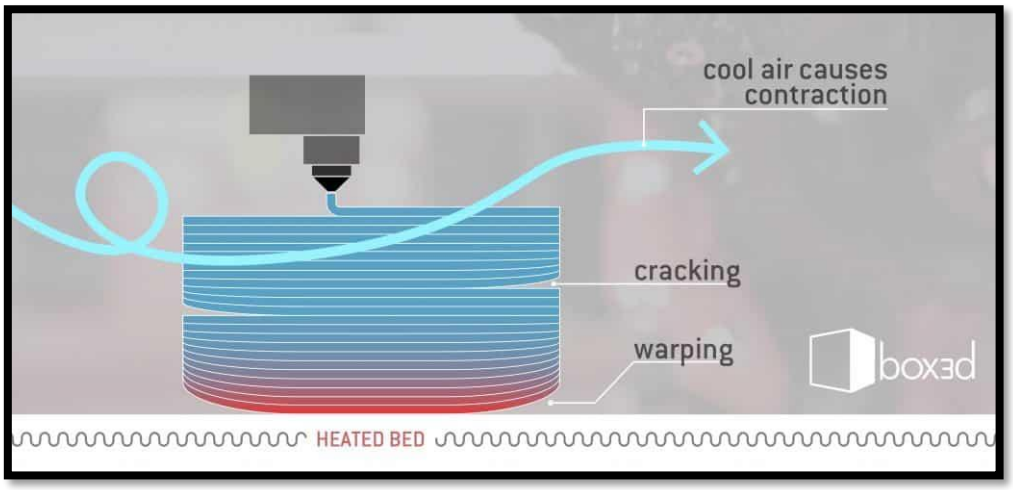


Ilustración 44. Efector "cracking" y "warping" de las impresoras 3D.

La impresión total del robot y el montaje final se puede observar en la siguiente imagen (Ilustración 38), se ha decidido no imprimir la cabeza ya que se trata solo de un prototipo para ver más que nada la estructura y aguante de las piernas.

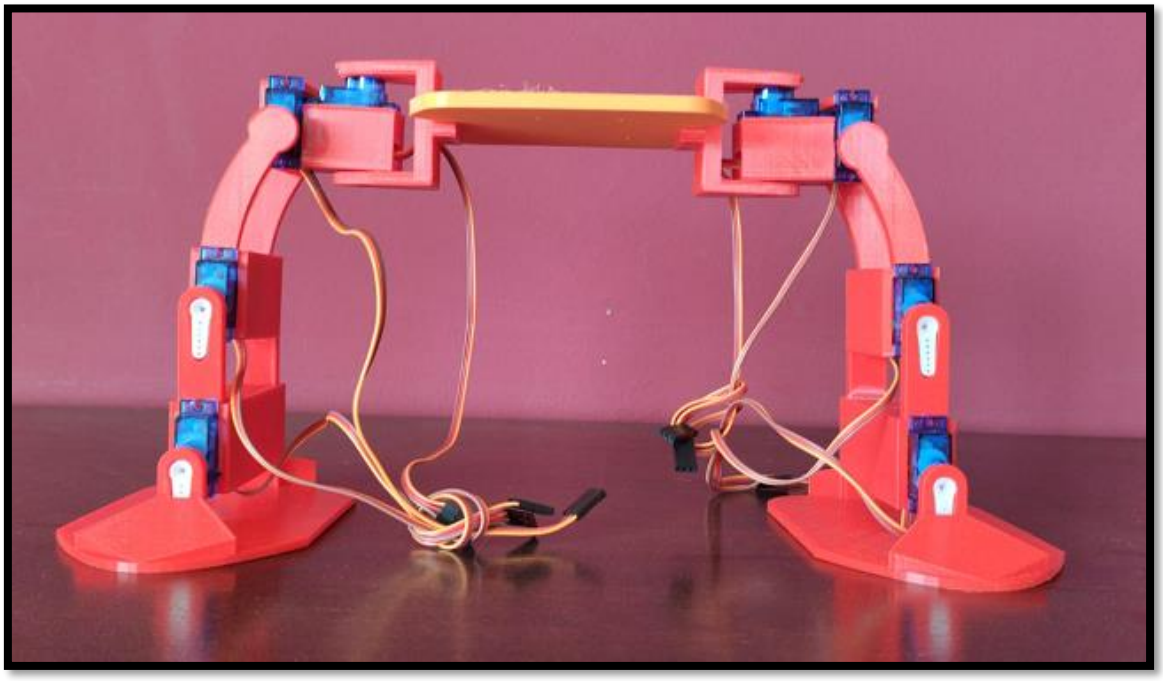


Ilustración 45. Impresión en 3D y montaje de las piernas y la base del robot.

Los servos encajan perfectamente en sus huecos diseñados, además de que la estructura se aguanta de pie.

3.- BIBLIOGRAFÍA

[1] *Sloth-kit-robot-bipedo-4-dof-con-arduino-nano-stem.jpg (458×458)*. (s. f.). Recuperado 26 de agosto de 2020, de https://www.tiendatec.es/4916-large_default/sloth-kit-robot-bipedo-4-dof-con-arduino-nano-stem.jpg

[2] Karel Čapek. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Karel_%C4%8Capek&oldid=123485837

[3] Seis grados de libertad. (2019). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Seis_grados_de_libertad&oldid=118703388

[4] *Cuerpo-de-Robot-b-pedo-de-13-grados-de-libertad-color-plata-y-negro-opcional-para.jpg_q50.jpg (700×1001)*. (s. f.). Recuperado 26 de agosto de 2020, de https://ae01.alicdn.com/kf/HLB1vRbHM4jaK1RjSZKzq6xVwXXaM/Cuerpo-de-Robot-b-pedo-de-13-grados-de-libertad-color-plata-y-negro-opcional-para.jpg_q50.jpg

[5] *Arduino Uno Rev3 | Arduino Official Store*. (s. f.). Recuperado 26 de agosto de 2020, de https://store.arduino.cc/arduino-uno-rev3

[6] *Wemos ESP32 D1 R32 WiFi y Bluetooth – Robótica Fácil*. (s. f.). Recuperado 26 de agosto de 2020, de https://roboticafacil.es/prod/wemos-esp32/

[7] *Servomotor MG995*. (s. f.). Electronicos Caldas. Recuperado 26 de agosto de 2020, de https://www.electronicoscaldas.com/es/motores-y-servos/608-servo-motor-mg995.html

[8] *Towerpro-sg90-mini-servo-electronilab-01.jpg (800×800)*. (s. f.). Recuperado 26 de agosto de 2020, de https://electronilab.co/wp-content/uploads/2014/05/towerpro-sg90-mini-servo-electronilab-01.jpg

[9] Gago. (2016, marzo 18). ¿Que es una batería LiPo? Aprende a leer su nomenclatura. *Mobus drones - Formación y desarrollo de drones*. https://mobus.es/blog/que-es-una-bateria-lipo/

[10] *Soporte de 4 baterías 18650 (Portapilas) – Robótica Fácil*. (s. f.). Recuperado 10 de septiembre de 2020, de https://roboticafacil.es/prod/soporte-de-4-baterias-18650-portapilas/

[11] *2 Baterías Li-ion 18650 3.7V 9800mAh – Robótica Fácil*. (s. f.). Recuperado 28 de agosto de 2020, de https://roboticafacil.es/prod/2-baterias-li-ion-18650-3-7v-9800mah/

[12] *I/O Extension Shield para Arduino Uno – Robótica Fácil*. (s. f.). Recuperado 30 de agosto de 2020, de https://roboticafacil.es/prod/extension-shield-arduino-uno/

[13] *Sensor Ultrasonidos HC-SR04 – Robótica Fácil*. (s. f.). Recuperado 30 de agosto de 2020, de https://roboticafacil.es/prod/hc-sr04/

[14] *Amazon.com: Creality Ender 3 Impresora 3D de código abierto con impresión de reanudación todo el marco de metal FDM Impresoras de bricolaje 8.661 x 8.661 x 9.843 in: Industrial & Scientific*. (s. f.). Recuperado 7 de septiembre de 2020, de https://www.amazon.com/-/es/Impresora-impresi%C3%B3n-reanudaci%C3%B3n-Impresoras-bricolaje/dp/B07D218NX3

[15] Warping and Cracking With Closed Environment FDM 3D Printers, Box3d. (2017, octubre 20). *Box3d*. https://box3d.eu/warping-cracking-closed-environment-3d-printers/

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO, ESTUDIO Y SIMULACIÓN DE UN ROBOT BÍPEDO EN COPPELIASIM (V-REP) Y CONTRUCCIÓN DE UN PROTOTIPO EN IMPRESIÓN 3D

ANEJO 1 :

CÓDIGO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Guillermo López Valero

TUTORIZADO POR

Leopoldo Armesto Ángel

CURSO ACADÉMICO: 2019/2020

```
function sysCall_threadmain()
    -- config pie izquierdo
    objetivo1=sim.getObjectHandle('Dummy_pie_izq_target')
    pie_izq=sim.getObjectHandle('Dummy_pie_izq_tip')
    path1=sim.getObjectHandle('Path_izq')
    pathLength1=sim.getPathLength(path1)
    posOnPath1=0
     l1=0
    -- config pie derecho
    objetivo2=sim.getObjectHandle('Dummy_pie_der_target')
    pie_der=sim.getObjectHandle('Dummy_pie_der_tip')
    path2=sim.getObjectHandle('Path_der')
    pathLength2=sim.getPathLength(path2)
    posOnPath2=0
    l2=0
    -- config pie izquierdo
    path3=sim.getObjectHandle('Path_izq2')
    pathLength3=sim.getPathLength(path3)
    posOnPath3=0
    l3=0
    -- config pie derecho
    path4=sim.getObjectHandle('Path_der2')
    pathLength4=sim.getPathLength(path4)
    posOnPath4=0
    l4=0
    -- config pie izquierdo
    path5=sim.getObjectHandle('Path_izq3')
    pathLength5=sim.getPathLength(path5)
    posOnPath5=0
    l5=0
```

```
-- config pie derecho
  path6=sim.getObjectHandle('Path_der3')
  pathLength6=sim.getPathLength(path6)
  posOnPath6=0
  l6=0
  -- config pie izquierdo
  path7=sim.getObjectHandle('Path_izq4')
  pathLength7=sim.getPathLength(path7)
  posOnPath7=0
  l7=0
  -- config pie derecho
  path8=sim.getObjectHandle('Path_der4')
  pathLength8=sim.getPathLength(path8)
  posOnPath8=0
  l8=0
  -- config pie izquierdo
  path9=sim.getObjectHandle('Path_izq5')
  pathLength9=sim.getPathLength(path9)
  posOnPath9=0
  l9=0
  - config pie izquierdo escalera
  path10=sim.getObjectHandle('Path_izq6')
  pathLength10=sim.getPathLength(path10)
  posOnPath10=0
  l10=0
 -- config pie derecho escalera
  path11=sim.getObjectHandle('Path_der6')
  pathLength11=sim.getPathLength(path11)
  posOnPath11=0
  l11=0
```

```
-- config pie derecho escalera
  path12=sim.getObjectHandle('Path_der7')
  pathLength12=sim.getPathLength(path12)
  posOnPath12=0
  l12=0

  -- config pie izquierdo escalera
  path13=sim.getObjectHandle('Path_izq7')
  pathLength13=sim.getPathLength(path13)
  posOnPath13=0
  l13=0

  -- config pie izquierdo escalera
  path14=sim.getObjectHandle('Path_izq8')
  pathLength14=sim.getPathLength(path14)
  posOnPath14=0
  l14=0

  -- config pie derecho escalera
  path15=sim.getObjectHandle('Path_der8')
  pathLength15=sim.getPathLength(path15)
  posOnPath15=0
  l15=0

  -- config pie izquierdo
  path16=sim.getObjectHandle('Path_izq9')
  pathLength16=sim.getPathLength(path16)
  posOnPath16=0
  l16=0
```

```
   -- config pie derecho
   path17=sim.getObjectHandle('Path_der9')
   pathLength17=sim.getPathLength(path17)
   posOnPath17=0
   l17=0
   -- config velocidades
   v1=0.16
   v2=0.2
   v3=0.34

while true do

   -- mover pie izquierdo1
   if (l1<1.2) then
   posOnPath1=posOnPath1+v1*sim.getSimulationTimeStep()
   l1 = posOnPath1/pathLength1
   PosRelative1=sim.getPositionOnPath(path1,l1)
   OrRelative1=sim.getOrientationOnPath(path1,l1)
 sim.setObjectPosition(objetivo1,-1,PosRelative1)
 sim.setObjectOrientation(objetivo1,-1,OrRelative1)
 sim.switchThread()
   end
-- mover pie derecho1
   if (l1>1.2) and (l2<1.05) then
   posOnPath2=posOnPath2+v2*sim.getSimulationTimeStep()
   l2 = posOnPath2/pathLength2
   PosRelative2=sim.getPositionOnPath(path2,l2)
   OrRelative2=sim.getOrientationOnPath(path2,l2)
```

```
sim.setObjectPosition(objetivo2,-1,PosRelative2)
sim.setObjectOrientation(objetivo2,-1,OrRelative2)
sim.switchThread()
 end
 -- mover pie izquierdo2
 if (l2>1.05) and (l3<1.3) then
 posOnPath3=posOnPath3+v2*sim.getSimulationTimeStep()
 l3 = posOnPath3/pathLength3
 PosRelative3=sim.getPositionOnPath(path3,l3)
 OrRelative3=sim.getOrientationOnPath(path3,l3)
sim.setObjectPosition(objetivo1,-1,PosRelative3)
sim.setObjectOrientation(objetivo1,-1,OrRelative3)
sim.switchThread()
 end
 -- mover pie derecho2
 if (l3>1.3) and (l4<1.3) then
 posOnPath4=posOnPath4+v2*sim.getSimulationTimeStep()
 l4 = posOnPath4/pathLength4
 PosRelative4=sim.getPositionOnPath(path4,l4)
 OrRelative4=sim.getOrientationOnPath(path4,l4)
sim.setObjectPosition(objetivo2,-1,PosRelative4)
sim.setObjectOrientation(objetivo2,-1,OrRelative4)
sim.switchThread()
 end
 -- mover pie izquierdo3
 if (l4>1.3) and (l5<1.3) then
 posOnPath5=posOnPath5+v2*sim.getSimulationTimeStep()
 l5 = posOnPath5/pathLength5
 PosRelative5=sim.getPositionOnPath(path5,l5)
 OrRelative5=sim.getOrientationOnPath(path5,l5)
```

```
sim.setObjectPosition(objetivo1,-1,PosRelative5)
 sim.setObjectOrientation(objetivo1,-1,OrRelative5)
 sim.switchThread()
  end
  -- mover pie derecho3
  if (l5>1.3) and (l6<1.3) then
  posOnPath6=posOnPath6+v2*sim.getSimulationTimeStep()
  l6 = posOnPath6/pathLength6
  PosRelative6=sim.getPositionOnPath(path6,l6)
  OrRelative6=sim.getOrientationOnPath(path6,l6)
 sim.setObjectPosition(objetivo2,-1,PosRelative6)
 sim.setObjectOrientation(objetivo2,-1,OrRelative6)
 sim.switchThread()
  end
  -- mover pie izquierdo4
 if (l6>1.3) and (l7<1.35) then
  posOnPath7=posOnPath7+v2*sim.getSimulationTimeStep()
  l7 = posOnPath7/pathLength7
  PosRelative7=sim.getPositionOnPath(path7,l7)
  OrRelative7=sim.getOrientationOnPath(path7,l7)
 sim.setObjectPosition(objetivo1,-1,PosRelative7)
 sim.setObjectOrientation(objetivo1,-1,OrRelative7)
 sim.switchThread()
end
  -- mover pie derecho4
  if (l7>1.35) and (l8<1.35)  then
  posOnPath8=posOnPath8+v2*sim.getSimulationTimeStep()
  l8 = posOnPath8/pathLength8
  PosRelative8=sim.getPositionOnPath(path8,l8)
  OrRelative8=sim.getOrientationOnPath(path8,l8)
```

```
sim.setObjectPosition(objetivo2,-1,PosRelative8)
 sim.setObjectOrientation(objetivo2,-1,OrRelative8)
 sim.switchThread()
  end
  -- mover pie izquierdo5
  if (l8>1.35) and (l9<1.35)  then
  posOnPath9=posOnPath9+v2*sim.getSimulationTimeStep()
  l9 = posOnPath9/pathLength9
  PosRelative9=sim.getPositionOnPath(path9,l9)
  OrRelative9=sim.getOrientationOnPath(path9,l9)
 sim.setObjectPosition(objetivo1,-1,PosRelative9)
 sim.setObjectOrientation(objetivo1,-1,OrRelative9)
 sim.switchThread()
  end
  -- escalon pie izquierdo6
  if (l9>1.35) and (l10<1.35)  then
  posOnPath10=posOnPath10+v2*sim.getSimulationTimeStep()
  l10 = posOnPath10/pathLength10
  PosRelative10=sim.getPositionOnPath(path10,l10)
  OrRelative10=sim.getOrientationOnPath(path10,l10)
 sim.setObjectPosition(objetivo1,-1,PosRelative10)
 sim.setObjectOrientation(objetivo1,-1,OrRelative10)
 sim.switchThread()
  end
 -- escalon pie derecho6
  if (l10>1.35) and (l11<1.35)  then
  posOnPath11=posOnPath11+v2*sim.getSimulationTimeStep()
  l11 = posOnPath11/pathLength11
PosRelative11=sim.getPositionOnPath(path11,l11)
  OrRelative11=sim.getOrientationOnPath(path11,l11)
 sim.setObjectPosition(objetivo2,-1,PosRelative11)
 sim.setObjectOrientation(objetivo2,-1,OrRelative11)
```

```
sim.switchThread()
 end
 -- escalon pie derecho7
 if (l11>1.35) and (l12<1.35)  then
 posOnPath12=posOnPath12+v2*sim.getSimulationTimeStep()
 l12 = posOnPath12/pathLength12
 PosRelative12=sim.getPositionOnPath(path12,l12)
 OrRelative12=sim.getOrientationOnPath(path12,l12)
sim.setObjectPosition(objetivo2,-1,PosRelative12)
sim.setObjectOrientation(objetivo2,-1,OrRelative12)
sim.switchThread()
 end
 -- escalon pie izquierdo7
 if (l12>1.35) and (l13<1.35)  then
 posOnPath13=posOnPath13+v2*sim.getSimulationTimeStep()
 l13 = posOnPath13/pathLength13
 PosRelative13=sim.getPositionOnPath(path13,l13)
 OrRelative13=sim.getOrientationOnPath(path13,l13)
sim.setObjectPosition(objetivo1,-1,PosRelative13)
sim.setObjectOrientation(objetivo1,-1,OrRelative13)
sim.switchThread()
 end
  -- escalon pie izquierdo8
 if (l13>1.35) and (l14<1.50)  then
 posOnPath14=posOnPath14+v2*sim.getSimulationTimeStep()
 l14 = posOnPath14/pathLength14
 PosRelative14=sim.getPositionOnPath(path14,l14)
 OrRelative14=sim.getOrientationOnPath(path14,l14)
sim.setObjectPosition(objetivo1,-1,PosRelative14)
sim.setObjectOrientation(objetivo1,-1,OrRelative14)
sim.switchThread()
 end
```

```
    -- escalon pie derecho8
   if (l14>1.50) and (l15<1.50)  then
   posOnPath15=posOnPath15+v2*sim.getSimulationTimeStep()
   l15 = posOnPath15/pathLength15
   PosRelative15=sim.getPositionOnPath(path15,l15)
   OrRelative15=sim.getOrientationOnPath(path15,l15)
  sim.setObjectPosition(objetivo2,-1,PosRelative15)
  sim.setObjectOrientation(objetivo2,-1,OrRelative15)
  sim.switchThread()
   end
   -- golpear pie izquierdo9
   if (l15>1.35) and (l16<2)  then
 posOnPath16=posOnPath16+v3*sim.getSimulationTimeStep()
   l16 = posOnPath16/pathLength16
   PosRelative16=sim.getPositionOnPath(path16,l16)
   OrRelative16=sim.getOrientationOnPath(path16,l16)
  sim.setObjectPosition(objetivo1,-1,PosRelative16)
  sim.setObjectOrientation(objetivo1,-1,OrRelative16)
  sim.switchThread()
   end
   -- golpear pie derecho9
   if (l16>2) and (l17<1.50)  then
   posOnPath17=posOnPath17+v2*sim.getSimulationTimeStep()
   l17 = posOnPath17/pathLength17
   PosRelative17=sim.getPositionOnPath(path17,l17)
   OrRelative17=sim.getOrientationOnPath(path17,l17)
  sim.setObjectPosition(objetivo2,-1,PosRelative17)
  sim.setObjectOrientation(objetivo2,-1,OrRelative17)
  sim.switchThread()
   end
end
end
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO, ESTUDIO Y SIMULACIÓN DE UN ROBOT BÍPEDO EN COPPELIASIM (V-REP) Y CONTRUCCIÓN DE UN PROTOTIPO EN IMPRESIÓN 3D

PLANOS

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

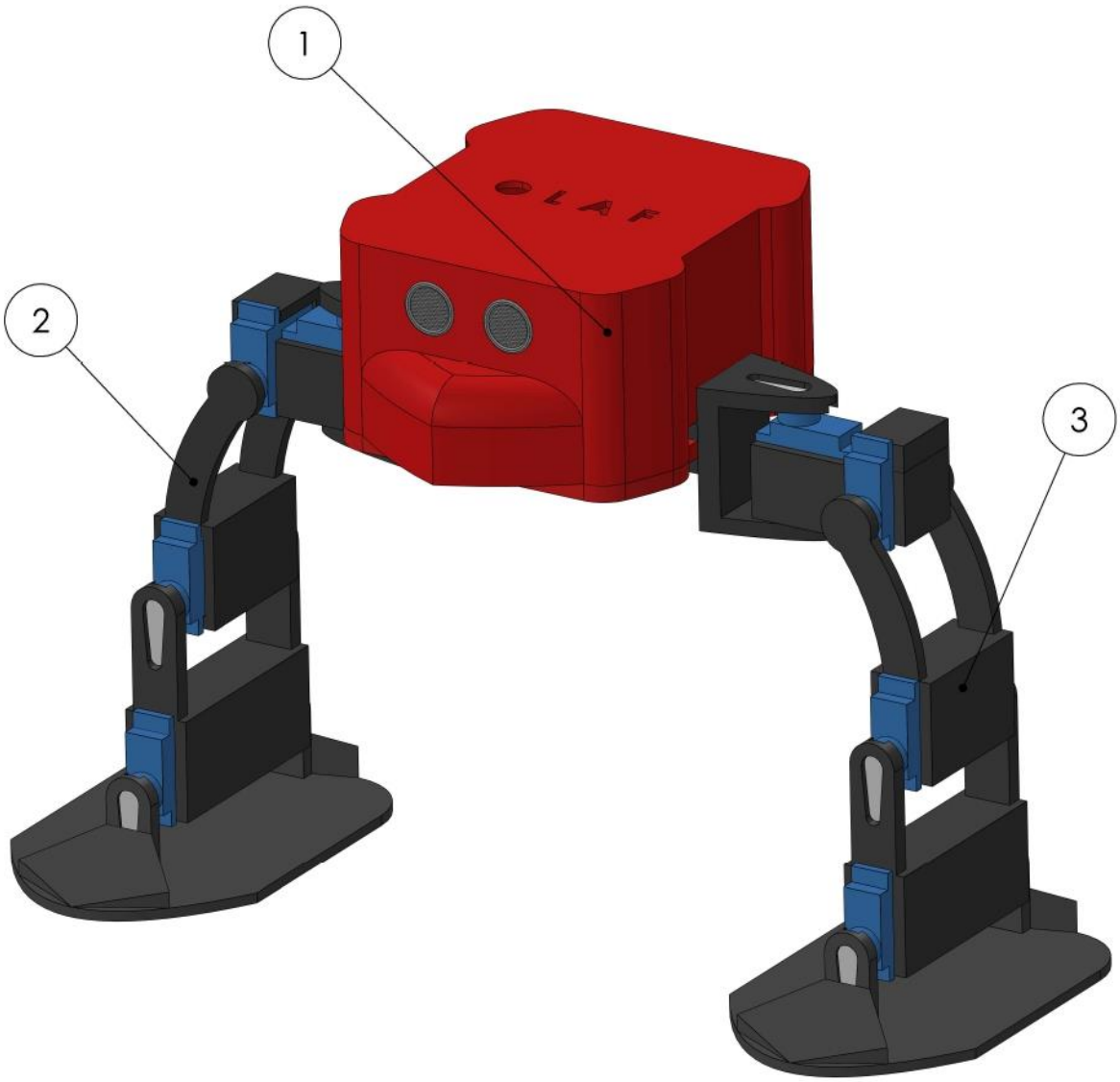
REALIZADO POR

Guillermo López Valero

TUTORIZADO POR

Leopoldo Armesto Ángel

CURSO ACADÉMICO: 2019/2020



Marca	Nº piezas	Designación	Material	Norma
3	1	Pierna izquierda	PLA	UNE 53978:2019
2	1	Pierna derecha	PLA	UNE 53978:2019
1	1	Chasis	PLA	UNE 53978:2019

Firma: Guillermo López Valero

Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D

Material: PLA

Referencia: 15042020-1

Escala: 1:2

Fecha: 15/04/2020

Tipo de documento: Vista general

Empresa: **Bpedtronics**

Título: **ROBOT BÍPEDO**

Plano Nº: 2

Hoja: 1/1

A4

Marca	Nº piezas	Designación	Material	Norma
2	1	Base	PLA	UNE 53978:2019
1	1	Cabeza	PLA	UNE 53978:2019

Firma: Guillermo López Valero

Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D

Material: PLA

Referencia: 15042020-4

Escala: 1:1

Fecha: 15/04/2020

Tipo de documento: Vista general de subconjunto

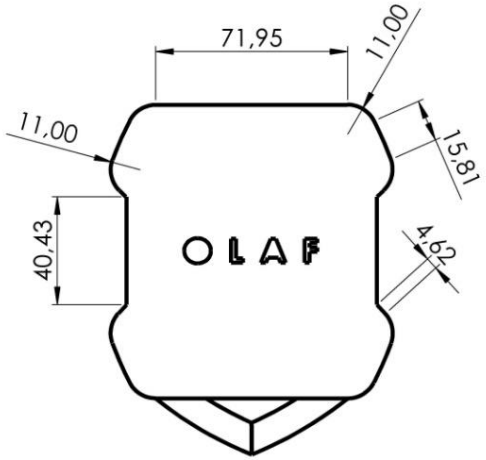
Empresa: **Bpedtronics**

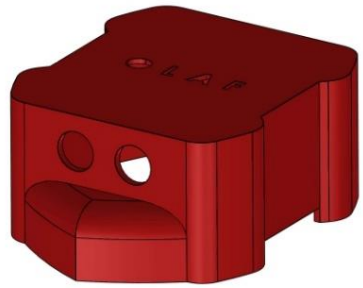
Título: **ENSAMBLAJE CHASIS**

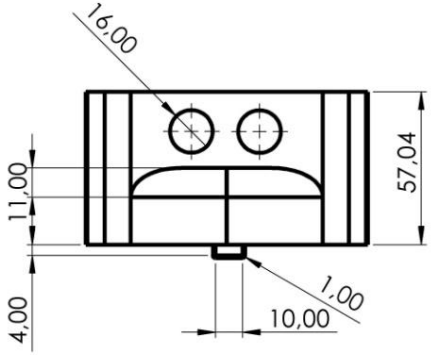
Plano Nº: 2.1

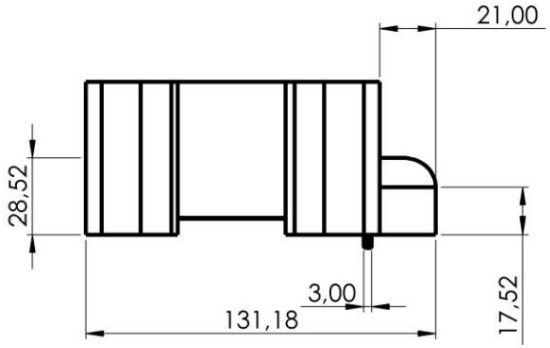
Hoja: 1/1

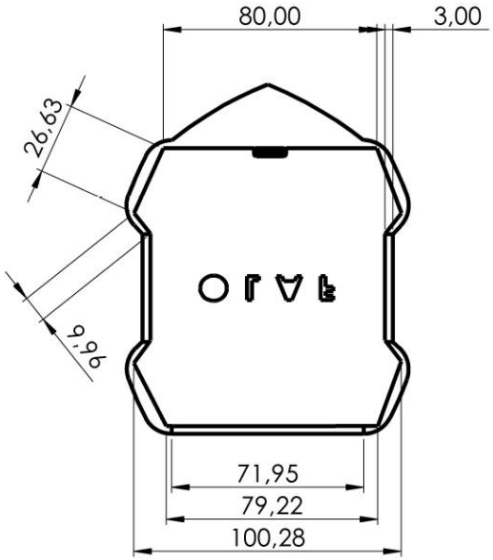
A4



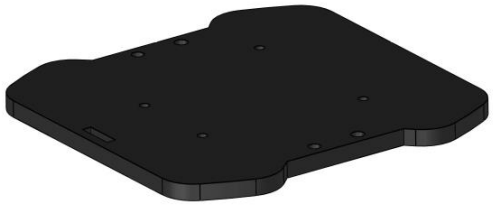


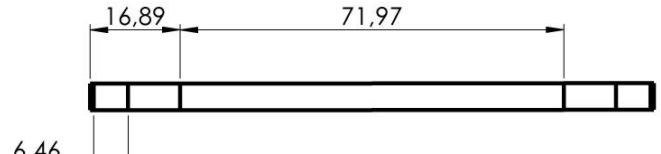


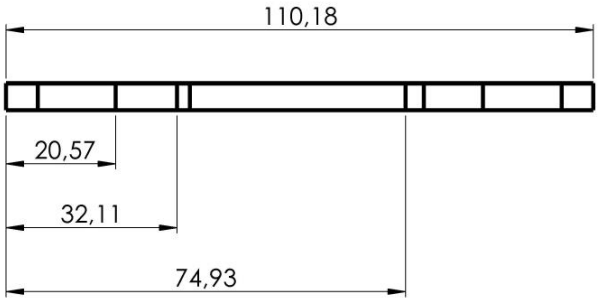


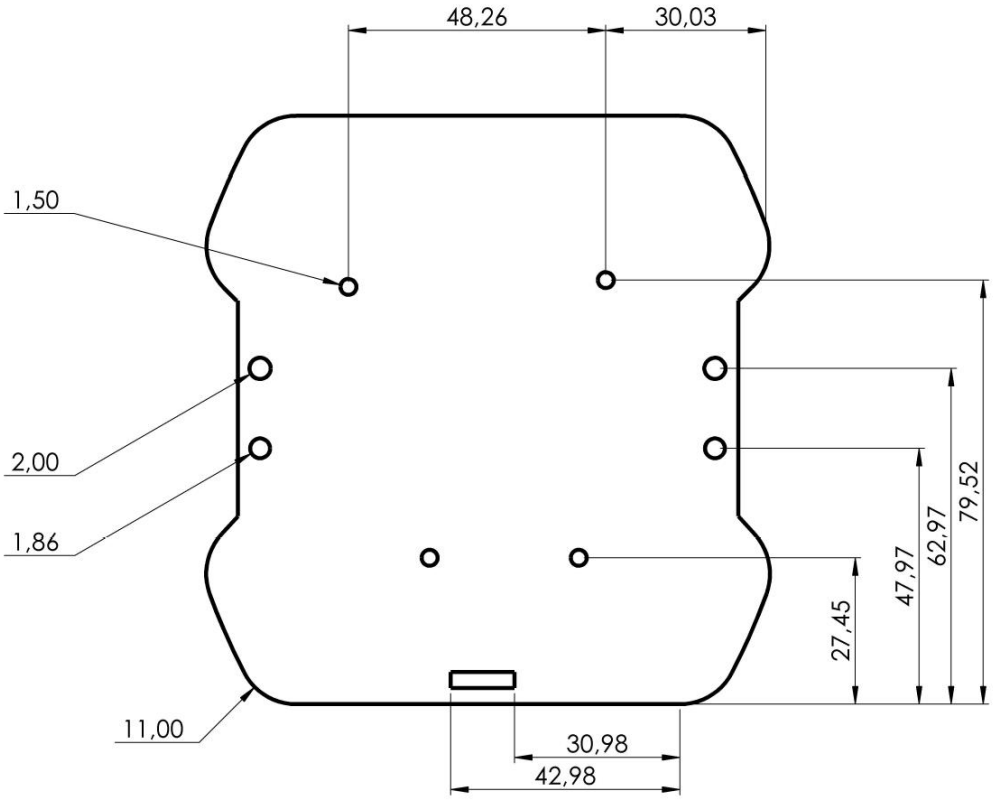


Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 15042020-5
		Escala: 1/2	Fecha: 15/04/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1211
Título: **CABEZA**		Plano Nº: 2.1.1	Hoja: 1/1 A3

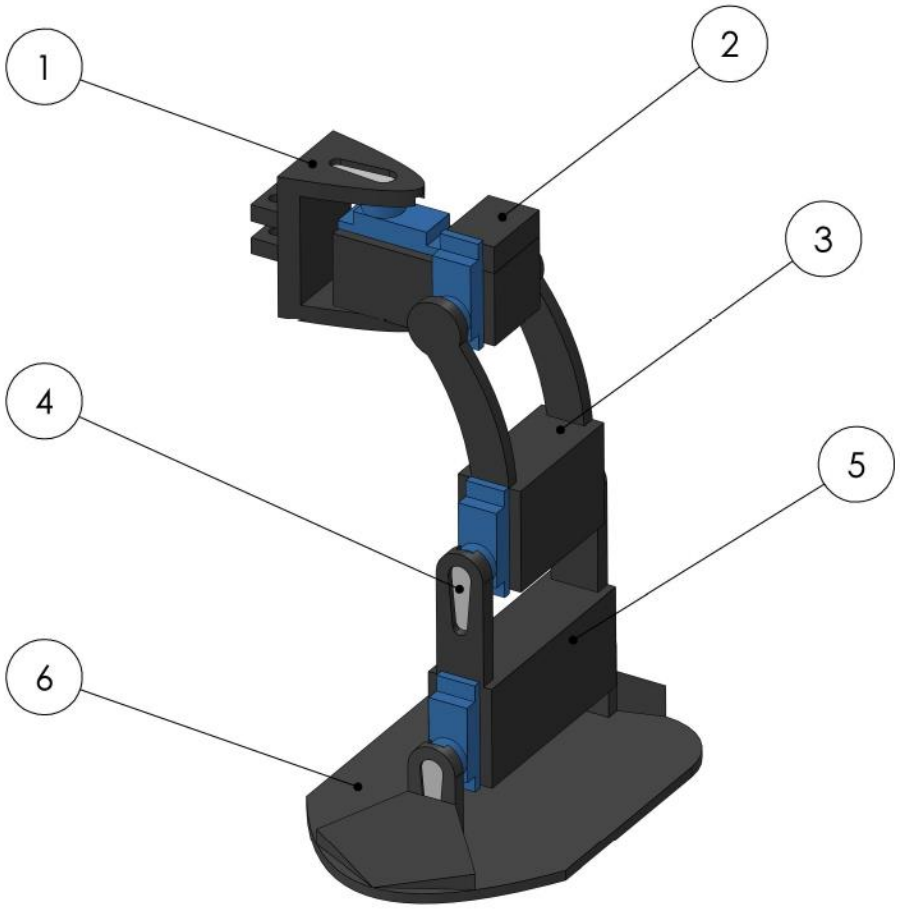








Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 15042020-6
		Escala: 1/1	Fecha: 15/04/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1211
Título: **BASE**		Plano Nº: 2.1.2	Hoja: 1/1 A3



Marca	Nª piezas	Designación	Material	Norma
6	1	Pie izq	PLA	UNE 53978:2019
5	1	Tobillo izq	PLA	UNE 53978:2019
4	4	Plástico Servo	PLA	UNE 53978:2019
3	1	Femur izq	PLA	UNE 53978:2019
2	1	Cadera izq	PLA	UNE 53978:2019
1	1	Sujeción izq	PLA	UNE 53978:2019

Firma:
Guillermo López Valero

Nombre del proyecto:
Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D

Material: PLA

Referencia: 15042020-2

Escala: 1:2

Fecha: 15/04/2020

Tipo de documento:
Vista general de subconjunto

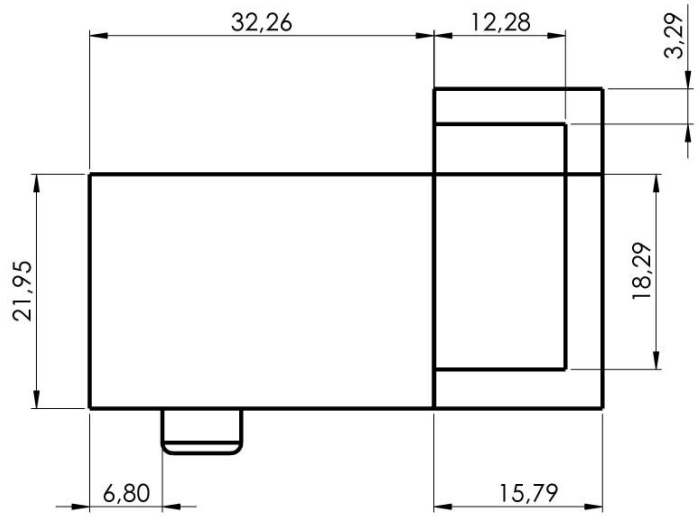
Empresa:
Bpedtronics

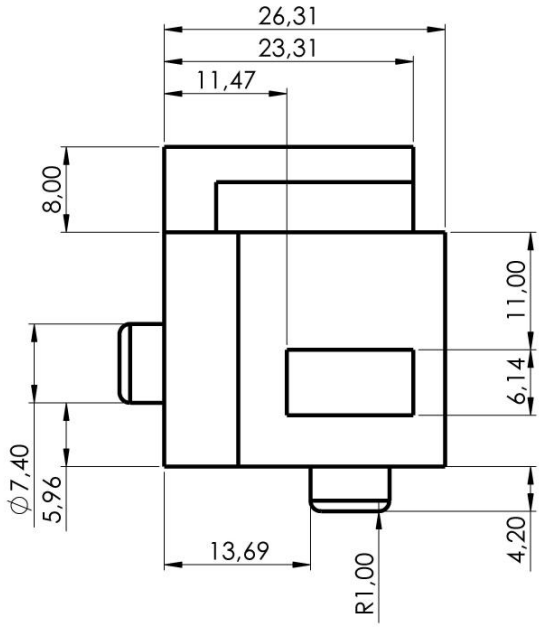
Título:
ENSAMBLAJE PIERNA IZQUIERDA

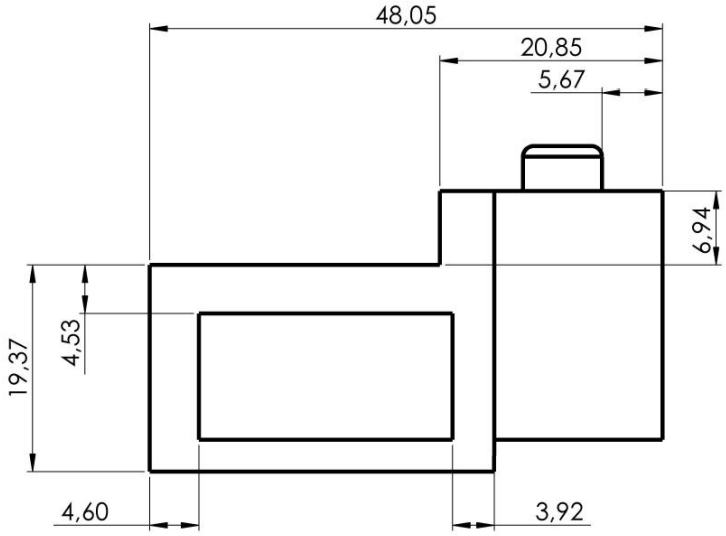
Plano Nº: 1.2.2

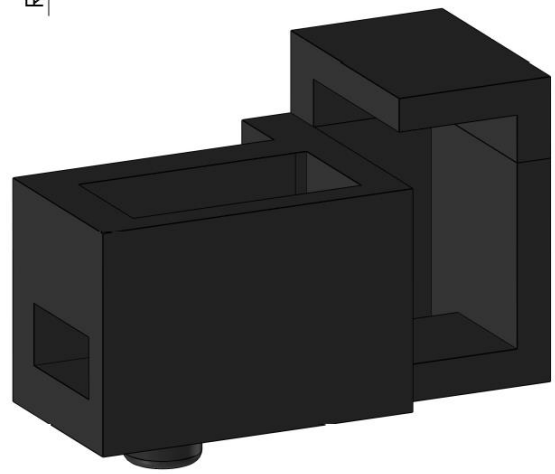
Hoja: 1/1

A4

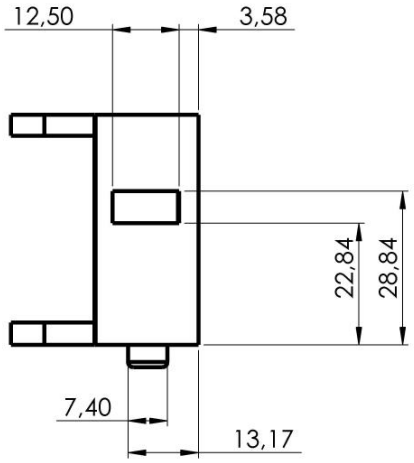


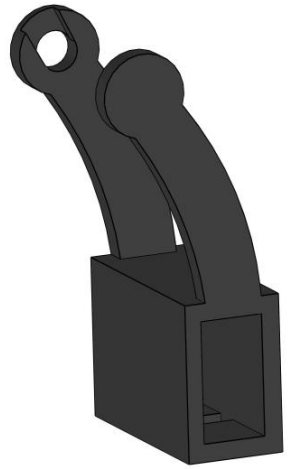


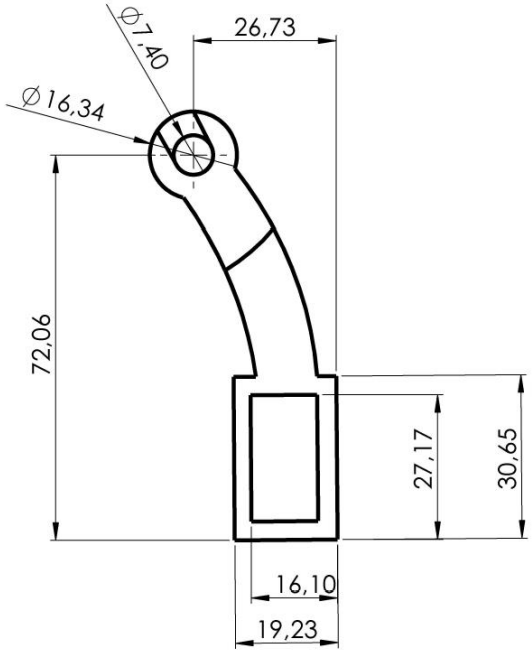


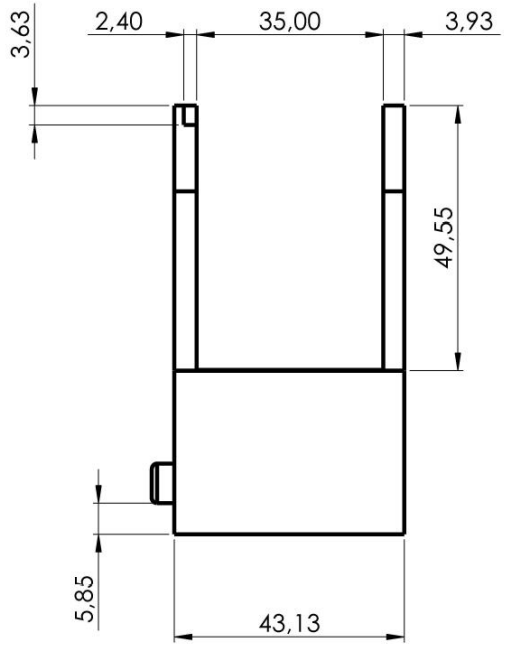


Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 15042020-9
		Escala: 2/1	Fecha: 16/04/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1222
Título: **CADERA IZQUIERDA**		Plano Nº: 1.2.2.1	Hoja: 1/1 A3

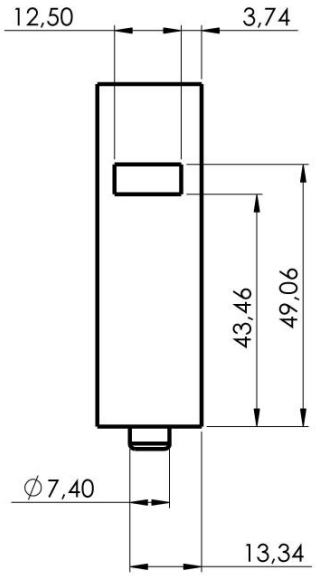


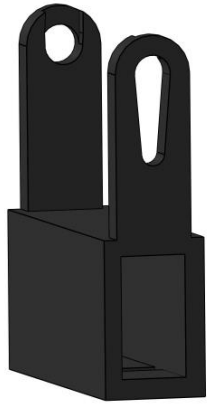


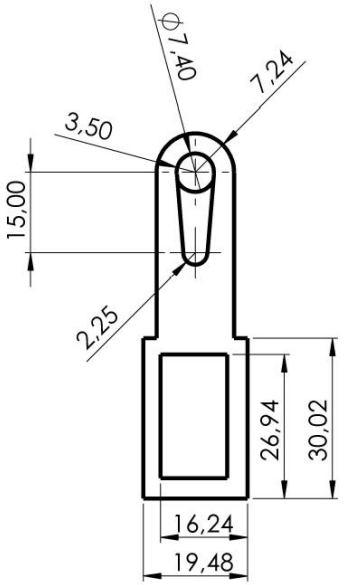


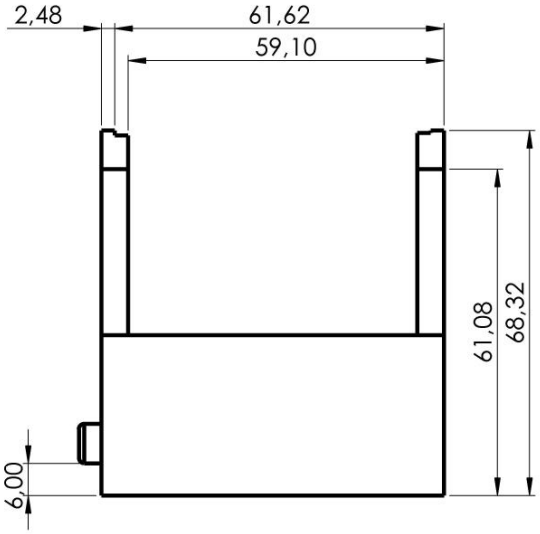


Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 15042020-11
		Escala: 1/1	Fecha: 21/04/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1223
Título: **FÉMUR IZQUIERDO**		Plano Nº: 1.2.2.2	Hoja: 1/1 A3

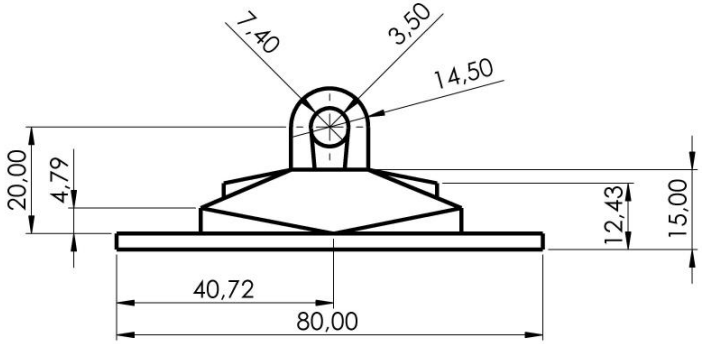


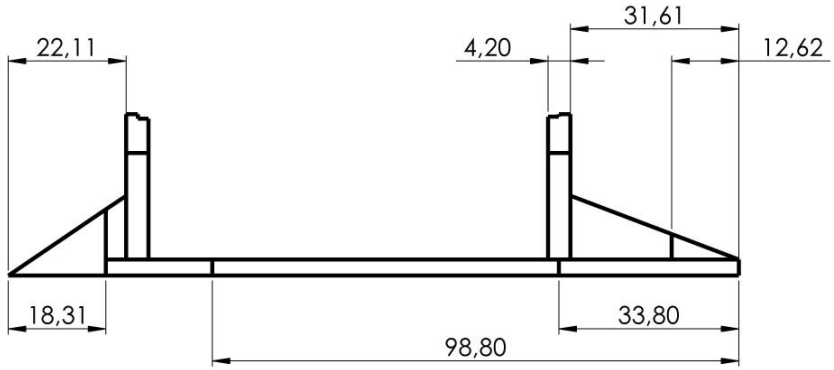


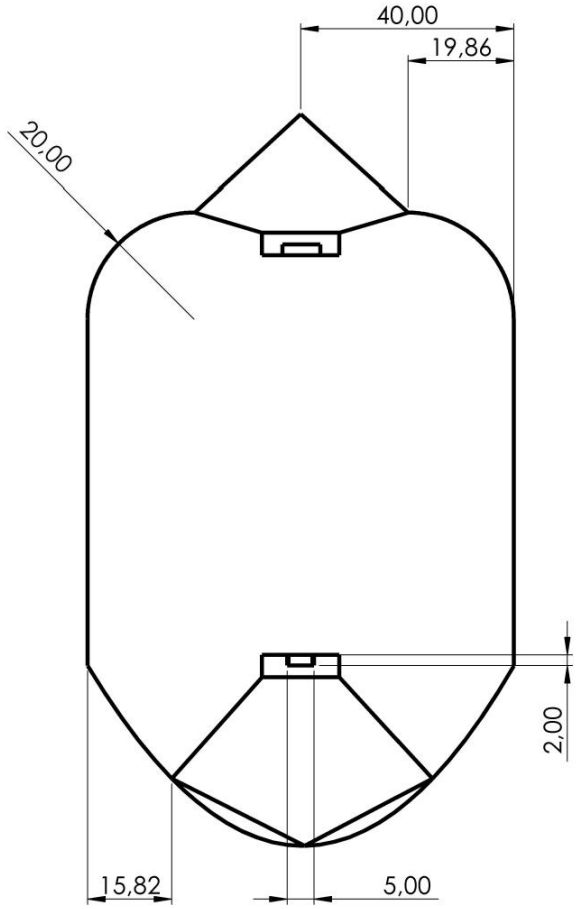


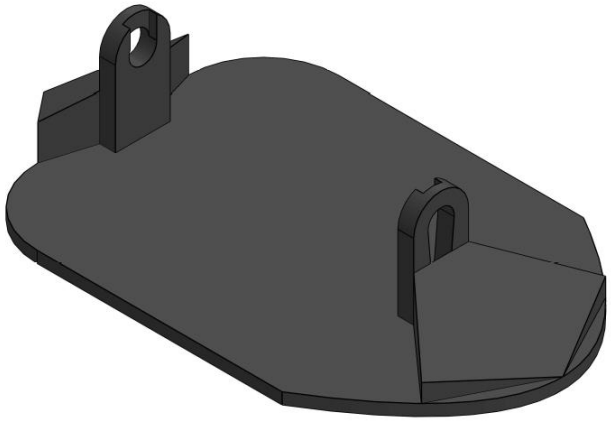


Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 15042020-13
		Escala: 1/1	Fecha: 26/04/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1224
Título: **TOBILLO IZQUIERDO**		Plano Nº: 1.2.2.3	Hoja: 1/1 A3

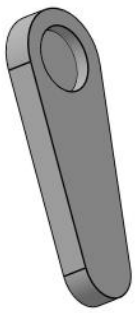


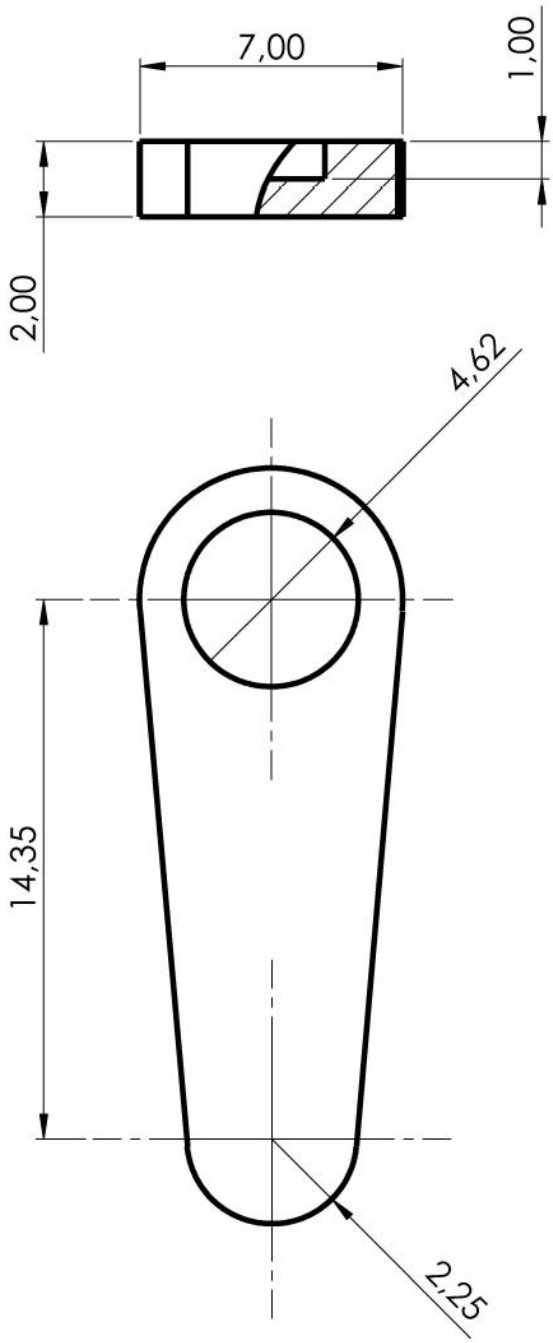




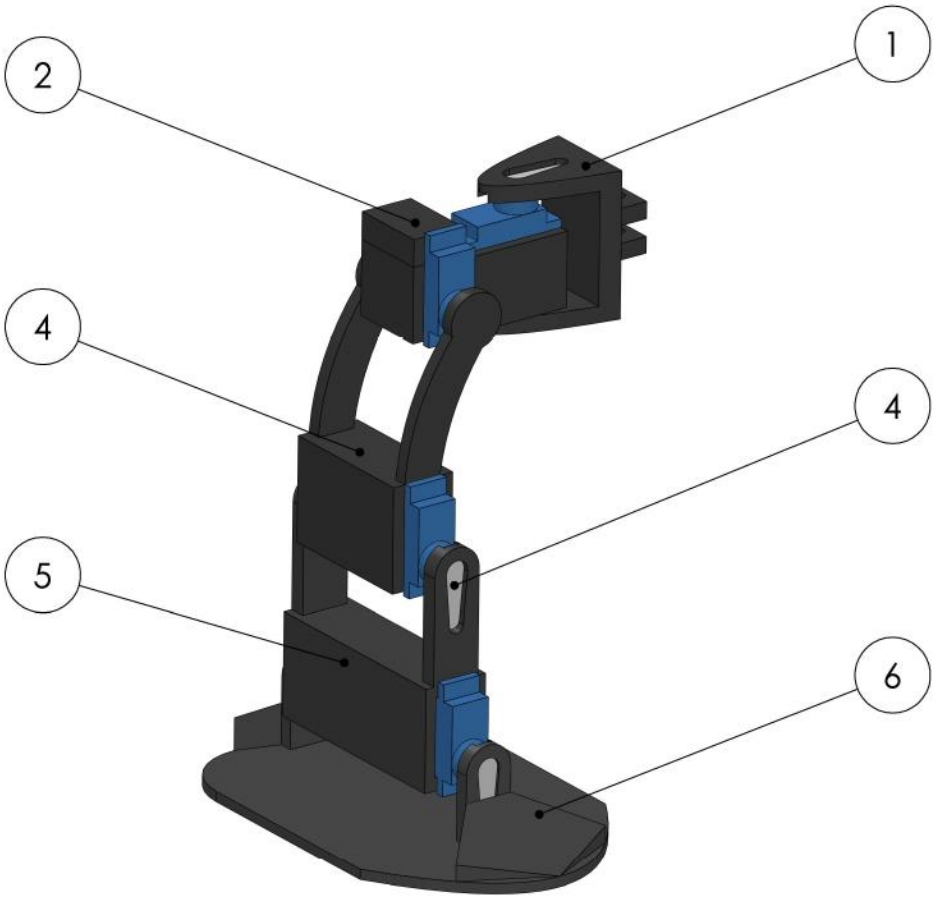


Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 15042020-15
		Escala: 1/1	Fecha: 30/04/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1225
Título: **PIE**		Plano Nº: 1.2.2.4	Hoja: 1/1 A3





Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 15042020-16
		Escala: 5/1	Fecha: 4/05/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1226
Título: **PIEZA SERVO**		Plano Nº: 1.2.2.5	Hoja: 1/1 — A4



Marca	Nª piezas	Designación	Material	Norma
6	1	Pie der	PLA	UNE 53978:2019
5	1	Tobillo der	PLA	UNE 53978:2019
4	4	Plástico Servo	PLA	UNE 53978:2019
3	1	Femur der	PLA	UNE 53978:2019
2	1	Cadera der	PLA	UNE 53978:2019
1	1	Sujeción der	PLA	UNE 53978:2019

Firma: Guillermo López Valero

Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D

Material: PLA

Referencia: 15042020-3

Escala: 1:2

Fecha: 15/04/2020

Tipo de documento: Vista general de subconjunto

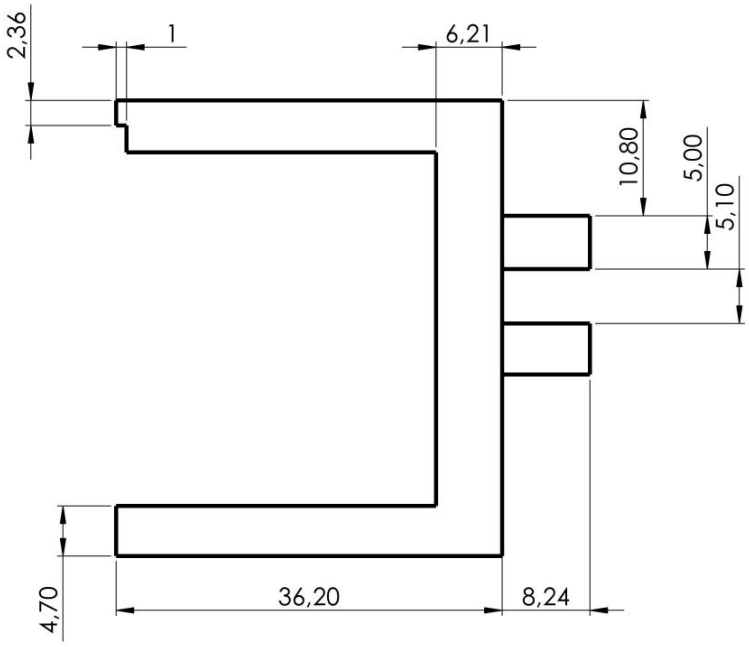
Empresa: **Bpedtronics**

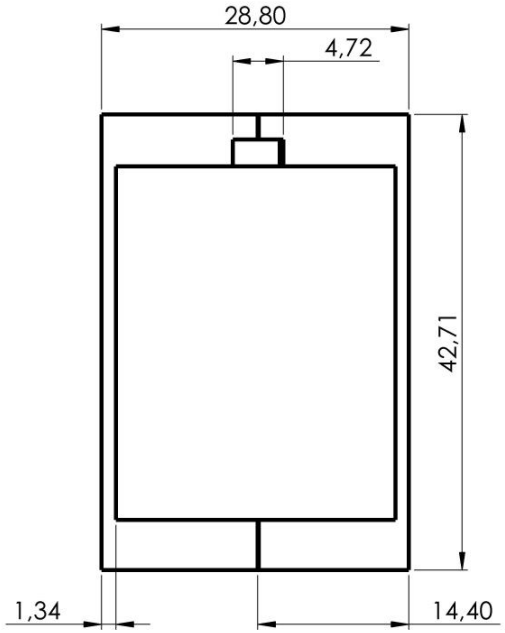
Título: **ENSAMBLAJE PIERNA DERECHA**

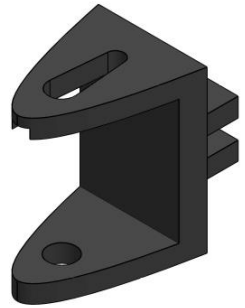
Plano Nº: 2.3

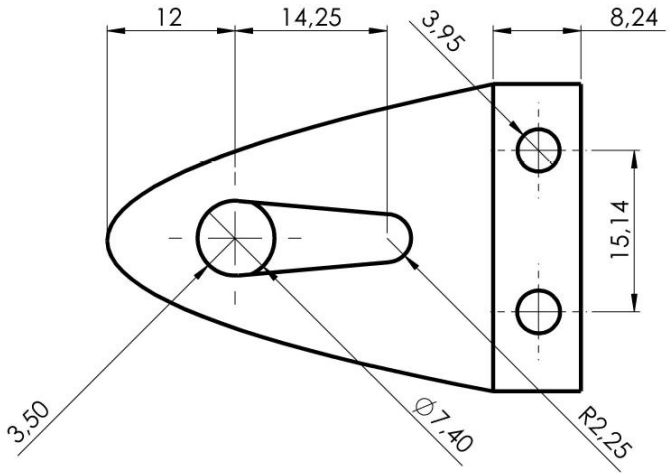
Hoja: 1/1

A4

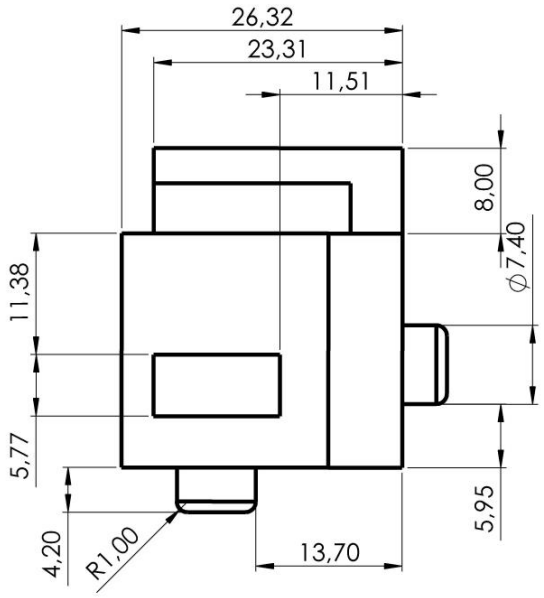


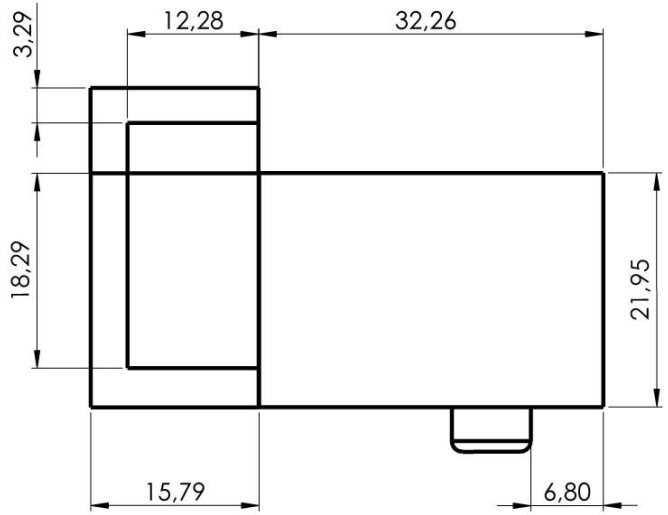


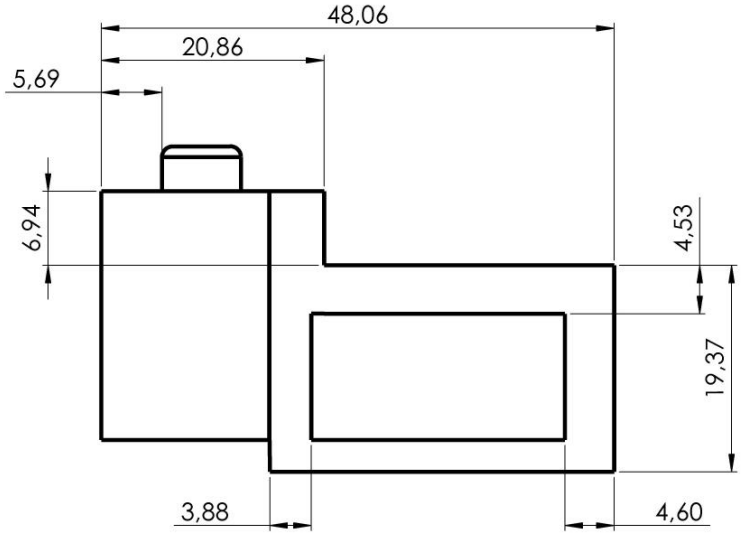


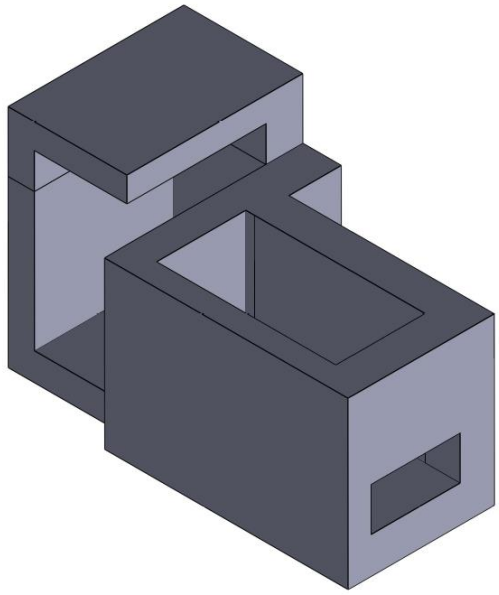


Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 15042020-7
		Escala: 2/1	Fecha: 16/04/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1231
Título: **SUJECIÓN DERECHA**		Plano Nº: 2.3.1	Hoja: 1/1 A3

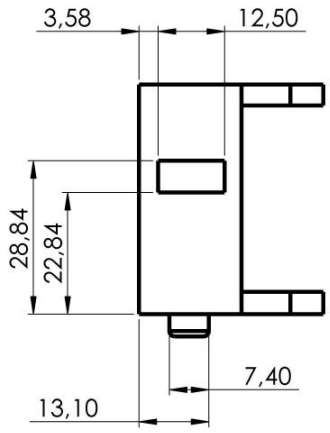


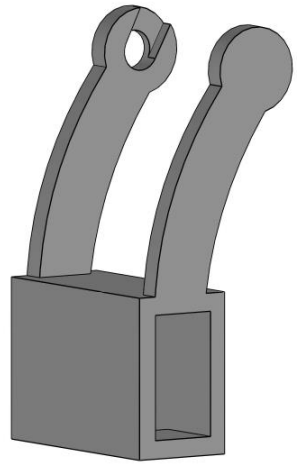


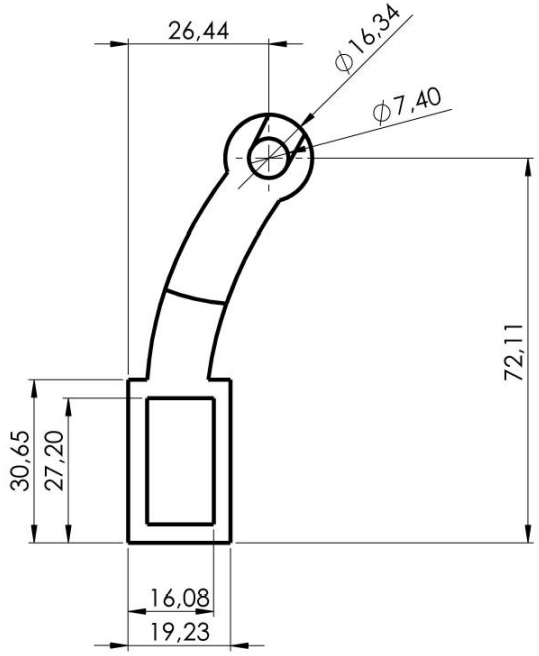


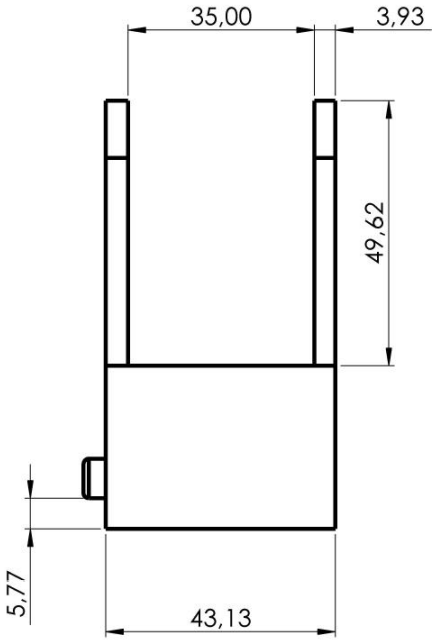


Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 16042020-1
		Escala: 2/1	Fecha: 16/04/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1232
Título: **CADERA DERECHA**		Plano Nº: 2.3.2	Hoja: 1/1 A3









Firma: Guillermo López Valero	Nombre del proyecto: Diseño, estudio y simulación de un robot bípedo en CoppeliaSim (V-REP) y montaje de este mediante impresión 3D	Material: PLA	Referencia: 21042020-2
		Escala: 1/1	Fecha: 21/04/2020
Tipo de documento: Dibujo de subconjunto		Empresa: **Bpedtronics**	Marca: 1233
Título: **FEMUR DERECHO**		Plano Nº: 2.3.3	Hoja: 1/1 A3

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO, ESTUDIO Y SIMULACIÓN DE UN ROBOT BÍPEDO EN COPPELIASIM (V-REP) Y CONTRUCCIÓN DE UN PROTOTIPO EN IMPRESIÓN 3D

PLIEGO DE CONDICIONES

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Guillermo López Valero

TUTORIZADO POR

Leopoldo Armesto Ángel

CURSO ACADÉMICO: 2019/2020

1.- OBJETO

El departamento de ingeniería se Sistemas y Automática de la Universidad Politécnica de Valencia solicita el diseño de un robot bípedo y la simulación del movimiento de este.

El promotor de dicho proyecto es Leopoldo Armesto Ángel, quién sugiere al ingeniero Guillermo López Valero la realización de un proyecto que cumpla con esos requisitos. Las condiciones de entrega serán la demostración mediante una memoria y un vídeo del proceso y trabajo realizado y el funcionamiento del robot.

El diseño del robot deberá ser realizado con una herramienta de dibujo en 3D, para así poder imprimir un prototipo del robot, y en un proyecto futuro poder programarlo. Los grados de libertad del robot pueden ser escogidos por el contratista, pero se sugiere un robot de 8 o 12 grados de libertad como máximo.

La entrega como bien se ha mencionado constará de la documentación necesaria para ver el seguimiento del trabajo realizado en el diseño y estudio del robot, y para la parte de programación, un vídeo donde se simule el movimiento de este.

2.- CONDICIONES DE LOS MATERIALES

Todos los materiales que participen en la fabricación del prototipo estarán conforme a la normativa vigente y exigible del marcado CE.

2.1.- Impresión con ABS

El material con el que se deberán imprimir todas las piezas del robot será ABS (Acrilonitrilo butadieno estireno) y cumplirá con la norma UNE-EN ISO 15015 con una elasticidad de tracción de 1500 MPa y una resistencia al impacto de 43 KJ/m².

Los métodos de ensayo y control calidad, realizado con probetas y demás procedimientos, se pueden encontrar en el documento de la norma UNE-EN ISO 15015

Las piezas del robot no podrán exceder las dimensiones de 220 x 220 x 250 mm, que es el máximo de impresión para la impresora que se tiene a disposición de su uso. Además, el número de paredes de las piezas deberá ser de 3, al igual que el número de capas inferiores y superiores. El relleno de la pieza será de un 20% y la altura de capa de 0.2 mm.

2.2.- Movimiento del robot

Los servomotores que se utilizarán en la impresión y simulación deberán ser capaces de mover las articulaciones y eslabones del robot sin que estas cedan.

Se deberán utilizar los servos SG90 o unos más potentes en caso de que el diseño sea pesado o grande, como es el ejemplo de los servos MG995. Además, cada servo deberá cumplir con su normativa y control de calidad, y contener el marcado CE.

Las piezas y accesorios de los servomotores deberán ser diseñadas e impresas también en 3D, cumpliendo las mismas características que el resto del robot.

3.- EJECUCIÓN Y MONTAJE

El proceso a seguir deberá empezar con el estudio e investigación de los robots caminantes y bípedos, para así realizar un diseño que pueda cumplir con las especificaciones y que sea válido. A partir de ahí es cuando se podrá pasar a la implementación del diseño en un simulador y a explicar el proceso seguido para que camine el robot.

Para la simulación se deberá crear un entorno en el que el robot muestre que puede caminar siguiendo un camino y sobrepasar algún obstáculo.

Por último para el montaje del prototipo debe ser con estructuras desmontables o con tornillos, pero nada de soldaduras o pegamento. Se contará con los planos y con el diseño en 3D del robot conforme a la norma UNE-EN ISO/ASTM 52915. En la que se exige la aportación de los documentos digitales de las piezas complejas. De esta forma se podrá construir el robot de una forma sencilla.

4.- ENSAYOS DE RECEPCIÓN

En caso de realizar la impresión y el montaje del prototipo del robot y posteriormente su programación, deberán tenerse en cuenta ciertos controles de calidad:

COMPONENTE	PRUEBA	CONTROL A REALIZAR	NUMERO DE CONTROLES	CONDCIÓN DE NO ACEPTACIÓN
SERVOMOTOR	Movilidad y posicionamiento de las piernas	Comprobar que puede girar de -180° a 180°	Al menos un control por cada servo existente	Los servos no llegan a 150° en uno de los 2 sentidos
PLACA DE CONTROL	Rececpción de órdenes y funcionamiento de los pines	Se comprobarán cada unos de los pines con un multímetro y también que recibe órdenes bluetooth	Un control antes del montaje y otro después	No recibe comandos por Bluetooth o fallan pines esenciales
BATERÍA	Carga, descarga y duración	Comprobación de las tensión máxima y mínima, además de la duración de la batería al primer uso	Un control durante el primer uso	Los valores de tensión no entran dentro de los de especificación o falla la descarga de la batería

5.- ENTREGA

Las condiciones de entrega entre el promotor Leopoldo Armesto Ángel y el contratista Guillermo López Valero establecen que la fecha de entrega se realizará antes del 16 de Octubre de 2020.

Debido a que el proyecto se trata de un estudia y simulación, el prototipo no habrá que entregarlo en mano, pero sí conservar par futuros proyectos. Por otro lado, sí que se hará entrega de la memoria y documentos necesarios donde se explica el trabajo realizado y los planos de las piezas

creadas. Además de un vídeo en el que se pruebe que la simulación funciona correctamente y cumple con lo establecido.

Por último se entregará el archivo de la simulación para comprobarlo en caso de necesidad o poder realizar más simulaciones.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO, ESTUDIO Y SIMULACIÓN DE UN ROBOT BÍPEDO EN COPPELIASIM (V-REP) Y CONTRUCCIÓN DE UN PROTOTIPO EN IMPRESIÓN 3D

PRESUPUESTO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Guillermo López Valero

TUTORIZADO POR

Leopoldo Armesto Ángel

CURSO ACADÉMICO: 2019/2020

1.- ANÁLISIS DEL PRESUPUESTO

Para el cálculo del presupuesto necesario para el proyecto se ha evaluado el coste total de los gastos desde el punto de vista del coste de recursos humanos, así como del coste de los equipos electrónicos utilizados. También se han tenido en cuenta los programas y herramientas informáticas necesarias en cada una de las partes del proyecto, y por último el coste de impresión de las piezas del robot.

1.1.- Coste de los Recursos Humanos

En el cálculo del coste de personal se ha tenido en cuenta las horas empleadas por el alumno en el desarrollo del TFG, lo que comprende un total de 300 horas, y contando con el coste medio de un ingeniero de 20€/h.

	Personal	Descripción	Cantidad (horas)	Coste unitario (€)	Importe total (€)
RECURSOS HUMANOS	Ingeniero junior	Diseño y desarrollo de las piezas y estructura del robot	150	20	3000
		Programación del movimiento y código del robot	100	20	2000
		Montaje del prototipo del robot (cuidado de las piezas, ensamblaje y revisión)	50	20	1000
	Total		300	60	**6000**

1.2.- Coste del Hardware y Software

Aquí se calculan los costes de las herramientas informáticas utilizadas, así como los equipos electrónicos empleados. Se tendrá en cuenta una amortización del 5% del coste original.

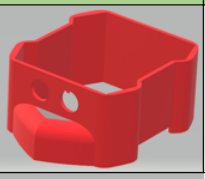
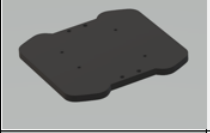
	Nombre	Descripción	Amortización (%)	Importe total (€)
HARDWARE Y SOFTWARE	Impresora 3D	Impresora 3D modelo Creality Ender 3 utilizada en la impresión de las piezas	5	15
	Ordenador	Programación del movimiento y código del robot	5	75
	CoppeliaSim	Software de simulación utilizado en la programación del robot	5	10
	SolidWorks	Software de diseño en 3D utilizado en la creación de las piezas	5	15
	Total			**115**

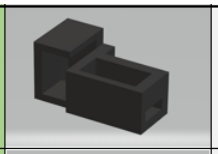
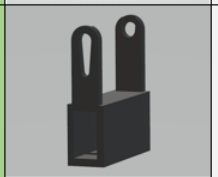
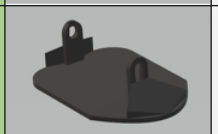
1.3.- Coste de la electrónica

Cálculo del coste de la electrónica utilizada para el montaje del prototipo, es decir, la placa, los servos, etc.

	Elemento	Imagen	Descripción	Cantidad (unidades)	Coste unitario (€)	Importe total (€)
ELEMENTOS ELECTRÓNICOS	Wemos ESP32 D1 R32		Placa de desarrollo de procesador ESP32 con WiFi y Bluetooth (BLE) compatible con shields de Arduino Uno y es mucho más rápido que el procesador ESP8266 (NodeMCU y similares). Se puede programar con Arduino IDE y Facilino.	1	11,60	11,60
	Porta pilas 18650		Este soporte de 4 baterías 186500 dispone de la electrónica para poder recargar baterías de Litio 18650. Este cargador es idóneo para proyectos que requieran un alto consumo de corriente, como por ejemplo robots con más de 4 servos	1	8,00	8,00
	Placa de extensión I/O		Shield para Arduino Uno y tarjetas compatibles con Arduino Uno (también para Wemos D1 R2 y Wemos D1 R32). Facilita la conexión de todas sus señales, proporcionando 3 pines (Señal, 5V y GND) por cada pin de entrada/salida.	1	4,50	4,50
	Batería 18650		Baterías de Li-Ion (modelo 18650) de 3.7V con capacidad 9800mAh (no es 100% real).	4	3,00	12,00
	Servo SG90		El servo SG90 Tower Pro un servo miniatura de diminutas dimensiones y muy económico. Los cables en el conector están distribuidos de la siguiente forma: Rojo = Alimentación (+), Marrrón = GND (-), Orange = Señal PPM.	8	2,00	16,00
	Sensor UltraSonidos HC-SR04		El HC-SR04 es un sensor de distancias por ultrasonidos capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 450 cm. El sensor funciona por ultrasonidos y contiene toda la electrónica encargada de hacer la medición.	1	2,00	2,00
	Total					54,10

1.4.- Coste de las piezas plásticas

	Elemento	Imagen	Descripción	Cantidad (unidades)	Coste unitario (€)	Importe total (€)
PIEZAS PLÁSTICAS	Cabeza		Pieza plástica del robot. Impresa con un 20% de relleno y una altura de capa de 0.2 mm.	1	0,930	0,930
	Base		Pieza plástica del robot. Impresa con un 20% de relleno y una altura de capa de 0.2 mm.	1	0,300	0,300
	Sujeción		Pieza plástica del robot. Impresa con un 20% de relleno y una altura de capa de 0.2 mm.	2	0,140	0,280

PIEZAS PLÁSTICAS	Cadera		Pieza plástica del robot. Impresa con un 20% de relleno y una altura de capa de 0.2 mm.	2	0,180	0,360
	Fémur		Pieza plástica del robot. Impresa con un 20% de relleno y una altura de capa de 0.2 mm.	2	0,290	0,580
	Tobillo		Pieza plástica del robot. Impresa con un 20% de relleno y una altura de capa de 0.2 mm.	2	0,270	0,540
	Pie		Pieza plástica del robot. Impresa con un 20% de relleno y una altura de capa de 0.2 mm.	2	0,310	0,620
	Total					2,990

1.5.- Presupuesto total

PRESUPUESTO TOTAL	COSTES	IMPORTE TOTAL (€)
	RECURSOS HUMANOS	6000
	HARDWARE Y SOFTWARE	115
	ELEMENTOS ELECTRÓNICOS	54,1
	PIEZAS PLÁSTICAS	2,99
	TOTAL	**6172,09**