# UNIVERSITAT POLITÉCNICA DE VALÈNCIA

## Escuela Técnica Superior de ingeniería del Diseño

# Robot boxeador multifunción

**TRABAJO FINAL DEL**

**Grado en Ingeniería Electrónica Industrial y Automática**

**REALIZADO POR**

**Adrià Bosch Serra**

**TUTORIZADO POR**

**Leopoldo Armesto Ángel**

**CURSO ACADÉMICO: 2020/2021**

# Resumen

La realización de este TFG consistirá en el diseño, montaje y programación de un robot boxeador. Por una parte, se diseñarán e imprimirán en 3D todas las piezas además de la selección de los componentes electrónicos. Por otra parte, se programará el robot, mediante el lenguaje Arduino, diversas funciones, una autónoma y otra de control mediante una aplicación. Esta aplicación también será desarrollada en el TFG utilizando el programa ThunkableX. En esta aplicación se podrá controlar mediante bluetooth los diversos movimientos del robot.

# Índice

# Índice de figuras

# DOCUMENTO 1: MEMORIA

# 1. Introducción

En este proyecto se desarrolla el diseño e implementación de un robot boxeador con dos funciones automáticas y una opción de control remoto. Para ello en este documento se especificarán los aspectos técnicos del proyecto además del contexto y objetivo de este.

Desde la invención de la robótica se ha buscado la creación de un robot lo más humanizado posible con tal de que realicen tareas similares a las de los humanos o poder asistirles en las mismas. Además, en los últimos años, la robótica se ha aproximado a la sociedad y se han creado otras funcionalidades más allá de la industrial como el entretenimiento o tareas del hogar. Por otra parte, gracias al avance de las tecnologías de impresión 3D o los microcontroladores Arduino, la creación de robots está al alcance de una gran parte de la población.

Con todo ello en este proyecto se diseña e implementa un robot boxeador utilizando tecnología de impresión 3D. Se trata de un robot bípedo capaz de imitar los movimientos de un boxeador humano. Este robot podrá ser controlado por el usuario mediante una aplicación también diseñada en este proyecto. Por otro lado, se dispondrá de dos programas los cuales establezcan un control automático del robot.

# 2. Motivación

La motivación del proyecto propuesto en este trabajo de final de grado proviene de diversas fuentes. En primer lugar, se persigue el aprendizaje en diversas áreas del conocimiento como la impresión y el diseño 3D, electrónica básica y robótica. En específico se busca un conocimiento práctico en diseño e impresión 3D dado que son conocimientos que no se imparten en el grado cursado. Por otra parte, se busca la experimentación con robots bípedos puesto que es una tipología de robot móvil que no ha sido explorada en el grado cursado. Por último, se pretende aplicar conocimientos sobre electrónica de control adquiridos.

# 3. Objetivos

El objetivo del presente proyecto es el diseño y ensamblaje de un robot boxeador impreso en 3D capaz de imitar los movimientos de un boxeador humano. A continuación, se enumeran los diversos subobjetivos del proyecto:

- Creación de un robot bípedo capaz de imitar movimiento de un boxeador humano
- Diseño e impresión 3D de todas las piezas del robot
- Diseño e implementación de toda la electrónica del robot
- Programación de dos aplicaciones autónomas. Una primera con un comportamiento agresivo y otra aplicación con un comportamiento más defensivo
- Programación de una aplicación de control remoto

# 4. Estado del arte

En este apartado se explica el contexto en el que se desarrolla el proyecto. El contexto de este proyecto está constituido por los conceptos de impresión 3D la robótica y el movimiento "maker". En particular se desarrolla la historia y conceptos básicos tanto de la robótica como de la impresión 3D, sin embargo, en lo referente al movimiento "maker" al ser un concepto de mayor ambigüedad se tratará de definir este concepto además de aportar datos históricos sobre este.

## 4.1. Impresión 3D

### 4.1.1. Historia de la impresión 3D

La impresión 3D tiene origen en la década de los 70 donde, en sus comienzos, no tenía una utilidad a nivel industrial, sin embargo, se presentaron múltiples patentes relacionadas con la fabricación por métodos aditivos asistida por ordenador (Savini & Savini, 2015). Es en 1984 cuando Charles Hull invento la estereolitografía (SLT) un proceso por el cual polímeros líquidos se solidificaban capa por capa bajo luz ultravioleta. Con este proceso fue capaz de imprimir el primer objeto impreso en 3D el cual consistía en una copa de 5 cm de alto cuyo proceso de fabricación duró dos meses. Dos años más tarde fundó la empresa 3D System dedicada a la fabricación de objetos por este método, sin embargo, seguía sin utilizase el termino de impresora 3D (Savini & Savini, 2015)(Zavaleta, 2019).

Por otra parte, al final de la década de los 80 se inventó el método de fabricación laminada de objetos (LOM) cuya aplicación no fue muy exitosa y cuyos productores fueron Helisys(USA), Solido3D(Israel) y Kira (japón) entre otros (Savini & Savini, 2015).

Poco después, en el estado de Tejas, Carl Deckard inventó el método sintetizado selectivo por láser (SLS) que fue patentado en 1989. Este método consistía en fundir partículas de polvo con un láser imprimiendo así capa por capa (Savini & Savini, 2015)(Zavaleta, 2019).

Finalmente, el método que dio nombre a las impresoras 3D fue inventado por Scott Crump en la década de los 80. Este método consistía en depositar capas de material termoplástico fundido mediante un robot de tres ejes formando así un objeto 3D. En 1992 patentó este método llamado modelado por deposición fundida (FDM) y fundó la empresa Dtratasys Inc dedicada a producir estas impresoras 3D (Savini & Savini, 2015)(Zavaleta, 2019).

Pese a todos estos avances en el año 2000 las impresoras 3D era máquinas de gran coste que no estaban al alcance del consumidor medio y solo se utilizaban en empresas para prototipado. Debido a esto surgieron diversas iniciativas que pretendían facilitar el acceso a la impresión 3D. Una de estas iniciativas fue el "Rep Rap" ideado por Adriann Bowyer en 2005 en la universidad de Bath. Esta iniciativa consistía en una impresora capaz de imprimir sus propias piezas dando como resultado la impresora Darwin (Figura 1) y la impresora Mendel (figura 2). Tanto el software como el hardware eran de código abierto, es decir, accesible a todo el mundo, y estaban basados en la plataforma Arduino facilitando así el acceso (Savini & Savini, 2015)(Zavaleta, 2019).

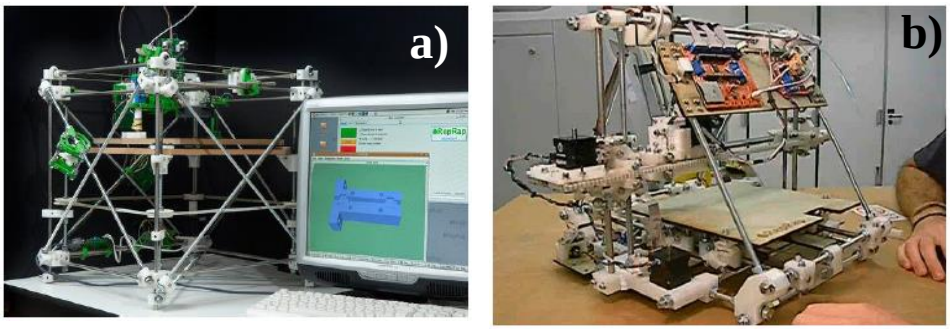



***Figura 1:*** *Primeras impresoras 3D: a) Impresora 3D Darwin (Zavaleta, 2019), b) Impresora 3D Mendel (Zavaleta,2019)*

Surgieron otras iniciativas similares como por ejemplo la surgida en 2006 en la universidad de Conell. Esta iniciativa llamada Fab@Home consistía en una impresora de tres ejes con diversos extrusores para diversos materiales cuyo hardware y software eran también abiertos (Savini & Savini, 2015)(Zavaleta, 2019).

Gracias a estos movimientos surgió una comunidad "Do it yourself" o "makers" movimiento muy favorecido por las iniciativas de software y hardware abierto. Sin embargo, la verdadera revolución sucedió con la expansión del consumo de las impresoras 3D. Esta revolución estaba encabezada por el profesor del MIT N.Gershenfield el cual había estado dando clases llamadas "How to make (almost) anything". Con este movimiento se crearon muchas comunidades de makers originando así la primera feria de makers en 2006(Savini & Savini, 2015).

### 4.1.2. Métodos y procedimiento de impresión 3D

Principalmente se utilizan tres métodos de impresión 3D dependiendo de la tecnología y material que se utilice. Según datos del 2017 y como se observa en la figura 2 el método más utilizado es el método de extrusión FDM seguido de cerca por el SLS. También se utiliza en una menor medida el SLA (Villagomez et al., 2017)

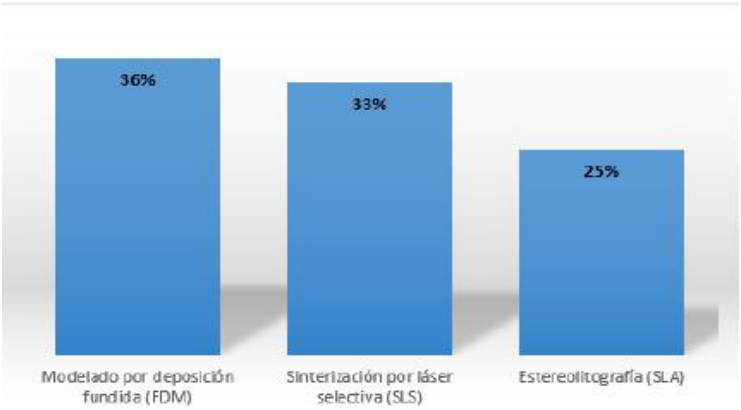



**Figura 2:** Top tecnologías de impresión 3D (Villagómez et al., 2017)

El método FDM, ejemplificado en la figura 3, es un método de extrusión que consiste en calentar a temperatura de fusión un material, generalmente PLA (Ácido poliláctico) o ABS (acrilonitrilo butadieno estireno), convirtiéndolo en fluido para depositarlo capa a capa. Seguidamente el material se solidifica formado así una pieza 3D (Zavaleta, 2019)(Villagomez et al., 2017).

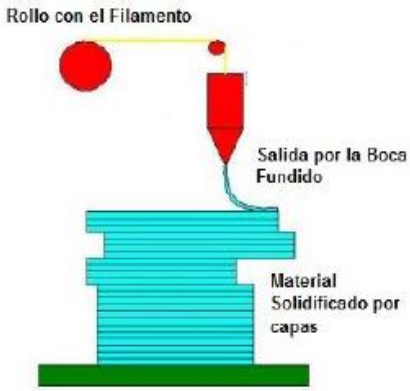



**Figura 3:** Método de Impresión 3D FDM (Villagómez et al., 2017)

Por otra parte, el método SLS, ejemplificado en la figura 4, es un método granular que consiste en la solidificación de material en polvo, depositado en una cama, con un láser. Después se vuelve a alimentar la cama con el material en polvo formando así capa tras capa (Zavaleta, 2019)(Villagomez et al., 2017).

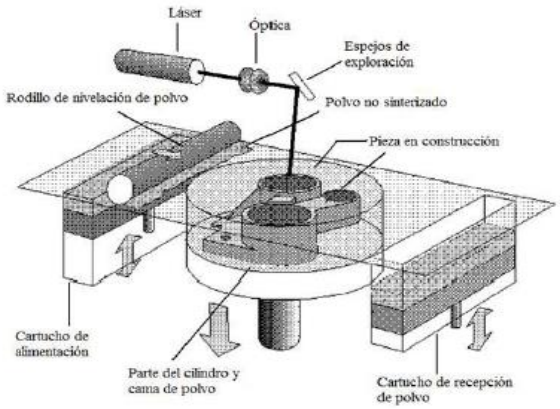



**Figura 4:** Método de impresión SLS (Villagómez et al., 2017)

Por último, el método de SLA, ejemplificado en la figura 5, consiste en una base sumergida en resina liquida, esta resina es solidificada por un láser para seguidamente sumergir la base lo necesario para solidificar otra capa de resina creando capa por capa la pieza 3D (Zavaleta, 2019)(Villagomez et al., 2017).

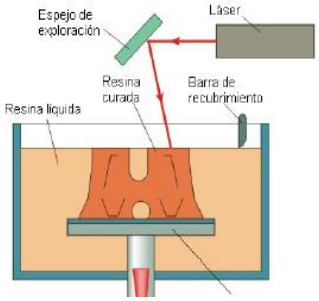



**Figura 5:** Método de impresión SLA (Villagómez et al., 2017)

Una vez escogido el método de impresión el proceso de impresión no varía. En primer lugar, se idea la pieza que se desea imprimir para seguidamente diseñarla en un programa de diseño 3D. A la hora de diseñar una pieza se ha de tener en cuenta las características necesarias para que la pieza cumpla su función, sin embargo, ha de tenerse en cuenta también que la pieza pueda ser impresa. Seguidamente se realiza el "slicing" que consiste en importar la pieza 3D a un programa específico capaz de transformar la pieza al código utilizado en la impresora. Además se han de configurar diversos parámetros como altura de las capas, forma del relleno, porcentaje de relleno etc.. Después se procede a la impresión y acabado de la pieza eliminando los soportes en caso de que hayan sido necesarios y mejorando el tacto de la pieza como por ejemplo lijándola para dar mayor suavidad.

## 4.2. El movimiento maker

El término maker es un término algo ambiguo y cuya definición no está establecida, pero tiene origen en el movimiento del "do it yourself" (DIY). El crecimiento de este movimiento en la última década es evidente y se puede observar en el aumento de búsquedas de conceptos como "maker" o "movimiento maker" (figura 6). Además, este movimiento ha sido definido por algunos como la nueva "revolución industrial" o "la democratización de la innovación" (Dougherty, 2012)(Rosa et al., 2017). Pero ¿Que es un "maker"? ¿En qué consiste este movimiento?

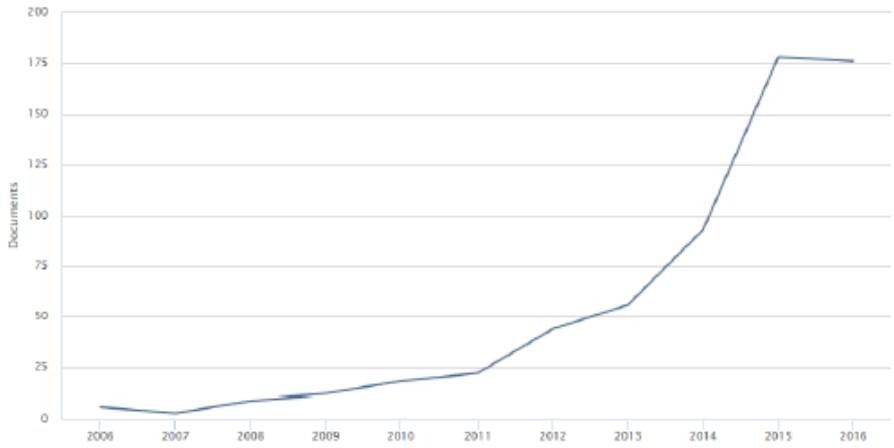



**Figura 6:** Búsquedas del término maker año 2017 (Rosa et al., 2017)

Para comprender este término se puede acudir al significado de la expresión "do it yourself", hazlo tú mismo, esta expresión quiere referirse a personas que fabrican, reparan o modifican por si mismos lo que necesitan. Ejemplo de esto sería una persona con conocimientos básicos de mecánica que en vez de llevar su coche a un taller es capaz de repararlo por sí mismo. Sin embargo, esta definición es ambigua puesto que una persona que se cocina su propia comida también entraría dentro de esta definición. (Dougherty, 2012)

Tampoco se podría utilizar el término "inventor" para ello puesto que la gran mayoría de "makers" no están inventando un nuevo producto, sino que simplemente fabrican, modifican o arreglan productos ya existentes (Dougherty, 2012). Sin embargo, personas como Steve Wozniak o Stewart Brand, creadores de Apple y Microsoft, también podrían considerarse de alguna forma "makers" puesto que estaban trabajando en proyectos DIY (Rosa et al., 2017).

Volviendo a la expresión de "la democratización de la innovación" para definir el movimiento "maker", este se basa en el libre acceso al conocimiento y materiales técnicos de ahí la palabra

democratización. Pese a que lo que encajaría con el término "maker" existe desde el origen del ser humano, lo que se entiende actualmente como "maker" surgió en la época de la información debido a esta "democratización" de lo técnico. Ejemplo de ello podrían ser las impresoras 3D antes mencionadas, o dispositivos de hardware y software abierto como las placas Arduino. Productos como estos y la divulgación del conocimiento en plataformas gratuitas de internet como Youtube han permitido que las personas tengan la informática, robótica u otros ámbitos como "Hobby". Todo ello provoco lo que actualmente se entiende como el movimiento "maker".

Con la expansión del movimiento "maker" se formaron diversas comunidades en iniciativas como FabLabs o Hackerspaces. Estas iniciativas tienen como objetivo que la gente de la comunidad DIY se conozca, intercambie ideas y colabore (Rosa et al., 2017). Un ejemplo de estas comunidades sería el grupo Makers de la Universitat Politècnica de València de donde han surgido proyectos como Fórmula Student o Azalea UPV.

## 4.3. Robótica

### 4.3.1. Historia de la robótica

El origen de la palabra robot es bien conocido, y fue en 1921 de la mano del escritor checo Karen Capek en el estreno de su obra *Rossum's Universal Robot*. El término robot proviene de la palabra eslava *Robota* que significa servidumbre o trabajo forzado y en la obra de Capek representaban máquinas androides que servían a sus jefes humanos en trabajos físicos. Tiempo después Isaac Asimov impulsó el uso de esta palabra en novelas como *Yo, robot*, donde además formuló las populares tres leyes de la robótica (Sotelo et al., 2007)(Ruiz-de-garibay & Estado, n.d.).

Sin embargo, los mecanismos automáticos han acompañado a la humanidad desde la antigüedad con mecanismos como los pájaros de Heron, un conjunto de aves que vuelan, gorjean y beben que se puede observar en la figura 8. Pero es en los siglos VIII a XV en la cultura árabe cuando se introducen estos seres mecánicos en la vida cotidiana con sistemas dispersores de agua o el gallo de Estrasburgo (1352) (figura 7) que formaba parte del reloj de la catedral de Estrasburgo (Sotelo et al., 2007)(Ruiz-de-garibay & Estado, n.d.).

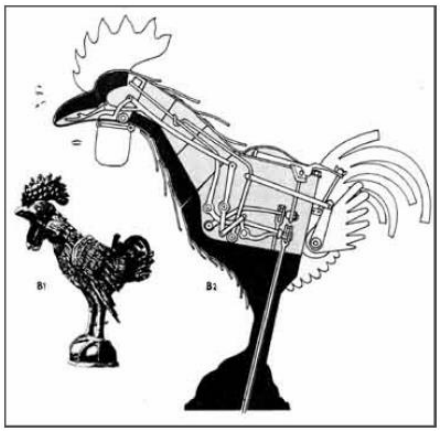


**Figura 7:** Gallo de Estrasburgo (Ruiz-de-garibay & Estado, n.d.)

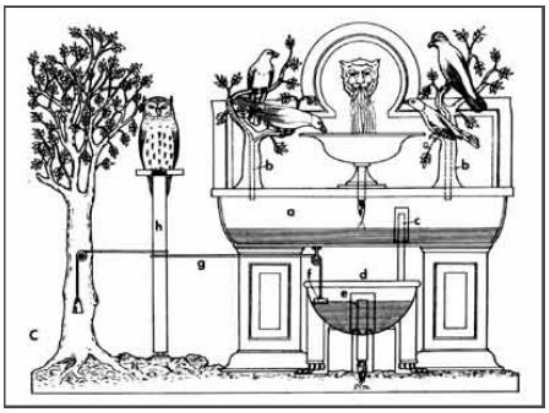


**Figura 8:** Pájaros de Heron (Ruiz-de-garibay & Estado, n.d.).

El origen del término robótica como ciencia nace en el siglo XX, pero su origen es en siglo XVIII donde se fabricaron autómatas como el pato de Vaucason (figura 9) o los muñecos de la Familia Droz. Además, en el siglo XIX con la industria textil surgieron diversos automatismos para mejorar la producción en esta industria. (Sotelo et al., 2007).

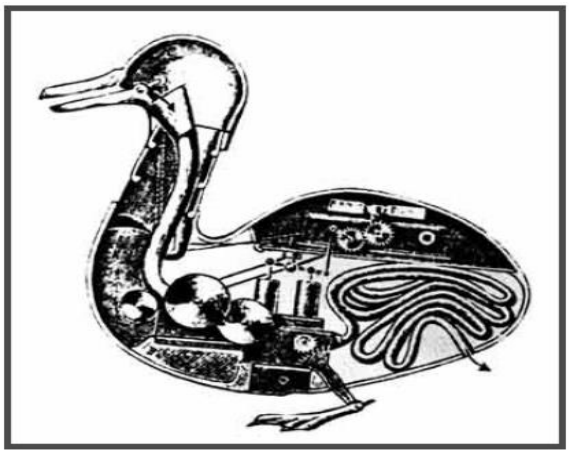


**Figura 9:** Pato de Vaucason (Ruiz-de-garibay & Estado, n.d.)

Es a partir de la década de los 60 cuando el crecimiento de la robótica toma un gran impulso comenzando con robots como los robots de Unimation (figura 10) los primeros robots industriales. Al tiempo se formaron las primeras asociaciones de robótica como el JIRA (Japón), el RIA (USA) o ASEA (Europa) (Ruiz-de-garibay & Estado, n.d.). Desde entonces el avance de la robótica ha sido evidente con logros como el control de un robot mediante un ordenador (1970) o la aplicación de inteligencia artificial a la robótica como por ejemplo el robot Shakey.

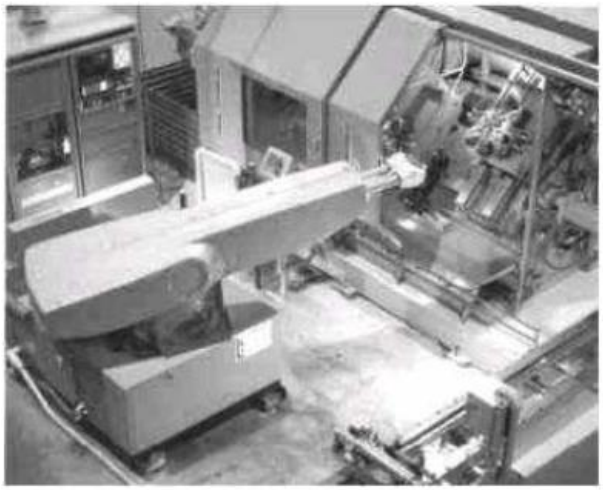


**Figura 10**: Primer robot industrial (1960) (Ruiz-de-garibay & Estado, n.d.)

### 4.3.2. Robots móviles

Con la evolución de la robótica surgieron diversos tipos de robots desde manipuladores o industriales a zoomórficos (Ruiz-de-garibay & Estado, n.d.). Sin embargo, en este trabajo se tratan los robots móviles puesto que es el tipo de robot al que pertenece el robot desarrollado. Un robot móvil se define como un sistema electromecánico capaz de desplazarse de manera autónoma sin

estar sujeto a un solo punto. En este género se incluyen los desplazados con ruedas patas orugas etc…(figura 11).

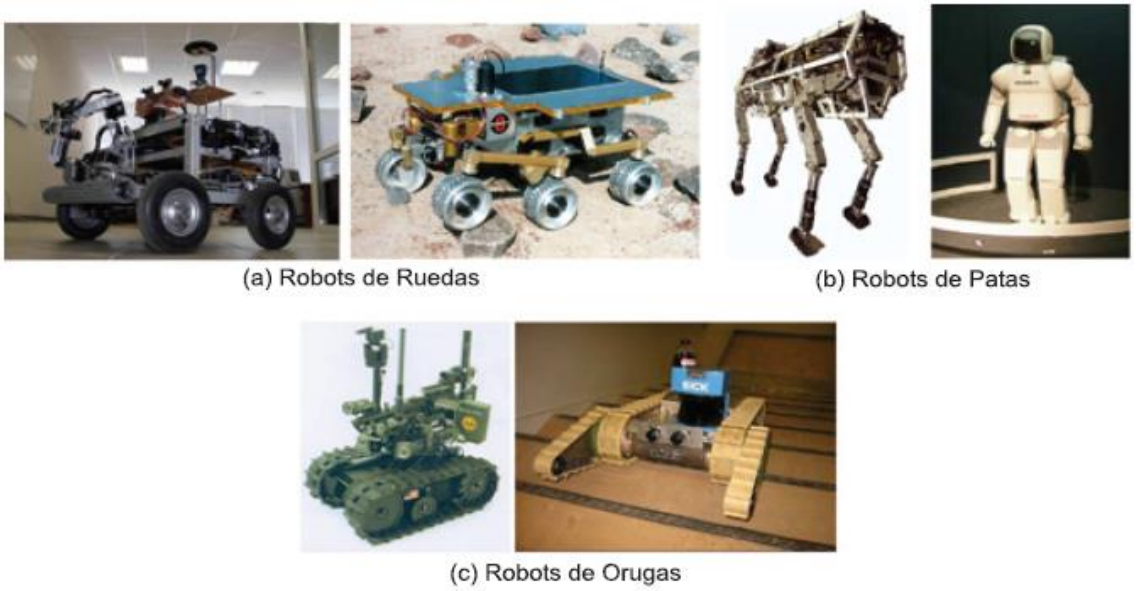



**Figura 11:** Tipos de robots móviles (Sotelo et al., 2007)

Actualmente los robots móviles forman parte de la sociedad en diversas formas desde tareas domésticas a exploración espacial. Cabe destacar el avance que han supuesto algunos robots móviles como por ejemplo el Rover de marte de la NASA (figura 12 a)) clave en la exploración espacial. Otro ejemplo de avance serían los robots de Boston Dynamic que han supuesto una revolución en el campo de la robótica móvil con robots cuadrúpedos, humanoides (figura12 b)) u oruga que poseen una movilidad extraordinaria (Guizzo, 2019).

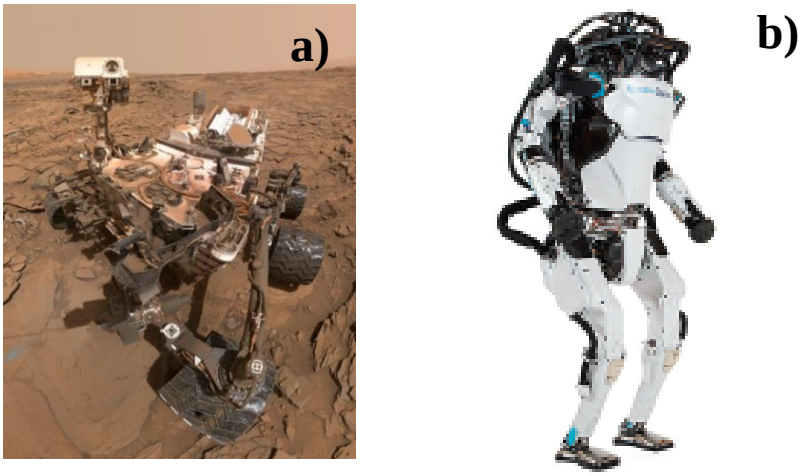



***Figura 12**:Robots móviles: a) Rover de Marte Curiosity Fuente: https://www.nasa.gov/, b) Robot Atlas Boston Dynamic (Guizzo, 2019)*

Otros robots móviles se han incorporado a la vida cotidiana como por ejemplo el juguete zowi (figura 13) o el robot domestico Roomba (figura 14) ejemplos claros de la integración de los robots en la sociedad.

**Figura 13:** Robot educativo Zowi. Fuente: https://www.rtve.es/infantil/series/zowi-robot-clan/


**Figura 14**: Robot doméstico Roomba. Fuente: https://www.amazon.es/

# 5. Antecedentes

A continuación, se detallarán antecedentes de productos o proyectos similares al realizado en este proyecto.

En primer lugar, destaca el popular juego de mesa de boxeadores representado en la figura 15. Este juego consiste en dos figuras representado robots boxeadores, estas figuras se manejan a través de una barra y al detectar cierto golpe el muñeco se bloquea. Existen versiones más modernas, sin embargo, el concepto es el mismo. Este producto no es un robot, pero imita el concepto de robot boxeador en el cual está inspirado el presente proyecto.

**Figura 15:** Juego de mesa de robots boxeadores. Fuente: https://www.amazon.es/

Otro producto similar es el Jackyee Robot boxing (figura 16) cuyo concepto es muy similar al desarrollado en este proyecto. Este concepto es un robot boxeador móvil, sin embargo, existen múltiples diferencias. La primera de ellas es la tipología de robot móvil, dado que el robot desarrollado en este TFG es un robot bípedo al contrario que el Jackyee es un robot de dos ruedas fijas. Por otra parte, el Jackyee no imita ningún movimiento de un boxeador real y solo puede manejarse desde un mando y no existe ningún modo automático.

**Figura 16:** Robot boxeador Jackyee. Fuente: https://www.amazon.es/

Continuando con conceptos similares al robot desarrollado cabe destacar el robot educativo zowi(figura 15) cuya forma de movimiento es en la que se basa este proyecto. En concreto se trata de un robot bípedo multifunción pensado para introducir a niños en la robótica. Puede manejarse desde una aplicación o programar de una manera muy básica programas simples. Sin embargo, el cuerpo del robot zowi, a diferencia del robot desarrollado en este proyecto, no tiene brazos, pero si cuenta con un pequeño panel LED para mejorar la expresividad del robot.

# 6. Materiales y Métodos

## 6.1. Materiales

### 6.1.1. Impresión 3D

A continuación, se detallan los materiales utilizados en la impresión de las piezas en 3D. La impresora es una impresora de tipo FDM modelo Ender 3 Pro (Figura 17). Se ha escogido este modelo en específico dado que pertenece a la nueva generación de impresoras 3D cuyo precio es relativamente bajo. Tiene una precisión similar al resto de impresoras de su generación y la anterior, sin embargo, su estructura es metálica lo cual provoca que sea mucho más sólida y no requiera de reajustes a lo largo del tiempo. Por otra parte, la cama de impresión es de cristal lo cual favorece la impresión.

**Figura 17:** Impresora 3D Ender 3 Pro.
Fuente: https://www.creality3dshop.eu/

En cuanto al material utilizado para la impresión se ha utilizado PLA. Este tipo de impresoras solo imprimen dos tipos de material ABS y PLA. Se ha escogido el material PLA debido a que su utilización es más simple y es menos exigente para la impresora dado que la temperatura de uso es menor.

### 6.1.2. Electrónica

La electrónica destinada al control está compuesta de un microcontrolador, un shield para el control de los servos, y un módulo para la comunicación bluetooth. En lo que se refiere al microcontrolador este es un modelo ELEGOO R3 (Figura 18 a)) basado en arquitectura Arduino y que utiliza el mismo lenguaje. Por otra parte, el shield de control de servos es el modelo PCA9685 (Figura 18 b)) con 16 pines. Se ha escogido este modelo debido a que es muy usual y apropiado para este tipo de proyectos por su compatibilidad con la arquitectura y lenguaje Arduino. Además, Se ha utilizado en módulo de bluetooth HM-10 debido a su compatibilidad con la arquitectura Arduino y que dispone de Bluetooth Low Energy (BLE) (Figura 18 c)).

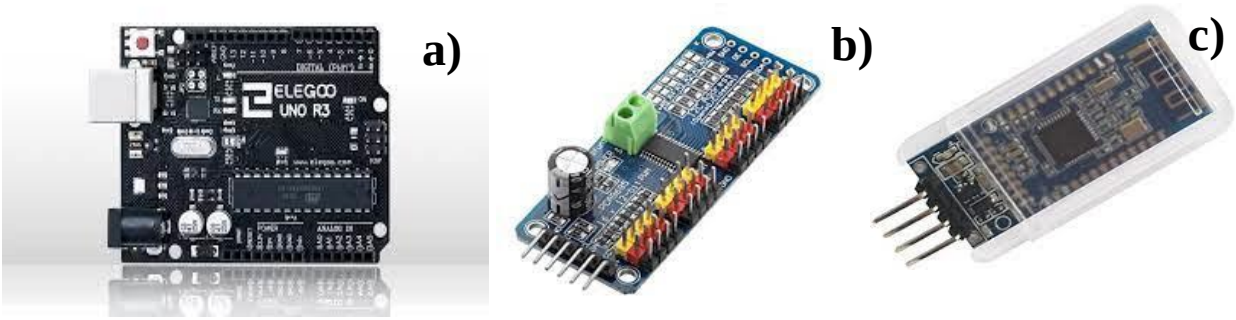



***Figura 18:****Componentes, a) Microcontrolador ELEGOO R3, b) PCA9685.c) Módulo bluetooth HM-10. Fuente: https://www.amazon.es/*

Por otro lado, la alimentación está compuesta por una pila Lipo (figura 19 a)), un convertidor DC-DC (figura 19 b)) y cableado diverso. La pila Lipo se trata de un modelo recargable de 9V 600mAh, escogido debido a su reducido tamaño y su voltaje idóneo para la electrónica escogida. Además, se necesitó un convertidor DC-DC con tal de alimentar a un voltaje apropiado la electrónica además de entregar suficiente potencia a los servomotores, aproximadamente 7V y 4A. En lo referente al cableado se han utilizado cables macho-macho y macho-hembra además de una adaptador Jack el cual adapta la entrada de la batería a cableado que se conecta directamente al convertidor DC-DC.

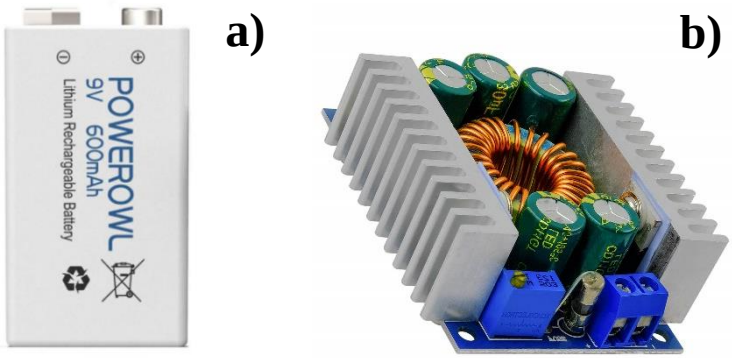



***Figura 19:*** *Componentes: a) Pila Lipo 9V, b) Convertidor DC-DC. Fuente: https://www.amazon.es/*

En cuanto a los sensores y actuadores se ha utilizado un sensor de ultrasonidos y ocho servomotores. El sensor de ultrasonidos se trata de un modelo HC-SR04 (figura 20) escogido debido a su compatibilidad con la arquitectura Arduino y su precio. Sin embargo, los servomotores se han escogido por criterios de rendimiento, en específico su fuerza y su estructura interna. Se trata del modelo MG996R (figura 21) con 13 Kg-cm de fuerza de parada y engranaje metálico para asegurar la movilidad y estabilidad del robot.


**Figura 20:** Sensor ultrasonidos HC-SR04. Fuente: https://www.amazon.es/

**Figura 21:** Servomotor MG996R. Fuente: https://www.amazon.es/

## 6.2. Método

### 6.2.1. Estructura

A continuación, se procede a detallar el método por el cual se ha diseñado la estructura. En primer lugar, se definirá el proceso por el cual se diseñan e imprimen todas las piezas, para seguidamente explicar el diseño de cada una de las piezas. La estructura se divide en tres secciones: tren inferior tren superior y articulaciones. En la figura 22 se puede observar la estructura general del robot.

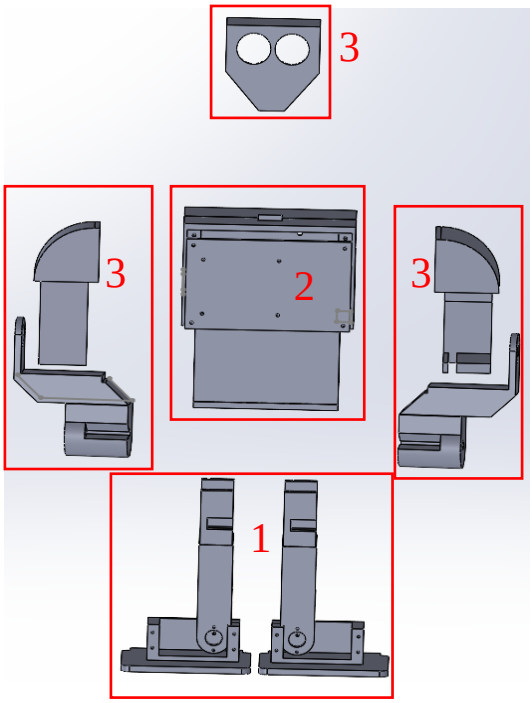



**Figura 22:** Estructura del robot. Tren inferior (1). Tren superior (2). Articulaciones

#### 6.2.1.1. Proceso de impresión

Todas las piezas utilizadas para este proyecto han sido impresas en 3D mediante el proceso FDM para lo cual se ha seguido el siguiente método.

En primer lugar, se realiza un croquis a escala 1:1 de la pieza con tal de poder visualizar parámetros de la pieza como el tamaño o compatibilidad con otras piezas y componentes como los servomotores o electrónica

En segundo lugar, se procede a diseñar la pieza en un programa de diseño 3D. En este caso se ha utilizado Solid Works dada su compatibilidad con los softwares de "slicing" que se utilizarán más adelante. Durante este proceso se ha de tener en cuenta si la pieza es imprimible o no y optimizar la impresión. Para ello se evita diseñar elementos que no estén apoyados en la base de impresión o elementos demasiado detallados o pequeños.

A continuación, se realiza el "Slicing" un proceso en el cual se utiliza un software específico para configurar la impresión. En primer lugar, se establece la orientación más optima de la pieza con tal de obtener la mejor calidad de esta en el menor tiempo posible, para ello se trata de no orientar la pieza de forma que no necesite soportes. Después se configuran parámetros entre los que destacan la altura de las capas y el tipo de base de la impresión. En lo referente a la altura de la capa esta determina la calidad de la pieza en altura a menor altura mayor calidad y detalle, sin embargo, la impresión tiene mayor duración, en el modelo de impresora utilizado puede variar de 0.1 mm a 0.4mm. En cuanto a la base de la impresión se trata de una superficie optativa de impresión que se imprime previamente a la pieza para asegurar un buen agarre a la cama de impresión, es recomendable para impresiones de gran tamaño.

Por último, se imprime la pieza y se le da el acabado final. Por una parte, puede haber soportes que haya que retirar y lijar con tal de obtener un mejor acabado. Por otra parte, a causa de imprecisiones puede haber alguna capa que lijar para igualar la pieza y mejorar su calidad.

#### 6.2.1.2. Tren inferior

La primera pieza del tren inferior son los pies del robot (figura 23 y 24). En el diseño de esta pieza se tuvieron en cuenta dos factores, la superficie de base y el tamaño de la pieza. La superficie tiene gran importancia debido a que si no es suficientemente grande el robot no se podrá mantener de pie con una sola pierna, esencial para su movimiento, por lo que a mayor superficie más podrá inclinarse el robot a una pierna. Sin embargo, si la pieza es demasiado grande entorpecerá el movimiento del robot. Por otra parte, se insertan surcos circulares con la función de mejorar el agarre utilizando silicona en ellos. Además, se diseñó la pieza para insertar un servomotor en un pequeño espacio de 47x28x20 mm para el que se debió tener en cuenta un pequeño espacio para los cables además de la forma de inserción del servomotor. Este se encajará atornillándose a la parte frontal del soporte y gracias a que el soporte tiene el tamaño ajustado para el servomotor. Por último, se ha diseñado un pequeño espacio en la parte trasera donde encajar la siguiente pieza.

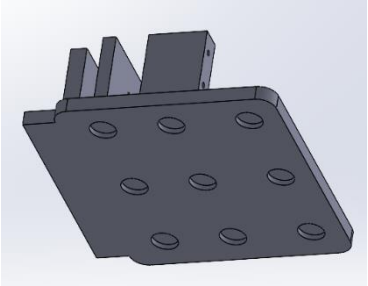


**Figura 23:** Pieza base (Pie parte superior)

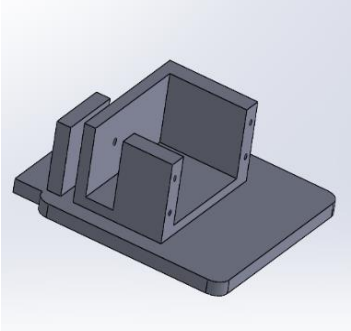


**Figura 24:** Pieza Base (Pie parte superior)

La segunda pieza se trata de la pieza que cumple a la función de pierna (figura 25). El objetivo de esta pieza es darle altura al robot y junto a la pieza de la base realizar un movimiento de tobillo. Esta pieza está diseñada para atornillarse al servomotor mediante la pieza representada en la figura 26. Además, para evitar que la pieza se doble por el peso estará atornillada a una sección del soporte para el servo motor de la pieza base como se observa en la figura 24. Para el diseño de esta pieza además se ha tenido en cuenta la altura de esta, dado que a mayor altura menor estabilidad. Por ello se ha diseñado la pieza con una altura de 70mm. Por otra parte, se ha añadido un travesaño para asegurar la estabilidad de la pieza y por tanto del robot. Por último, se le han añadido dos orificios en la parte superior para la unión con la siguiente pieza.

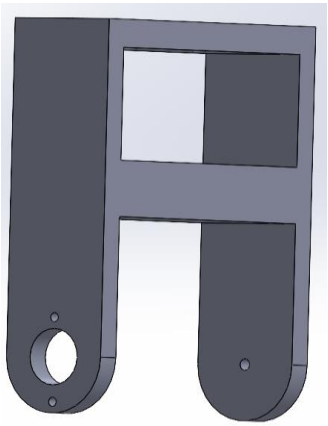


**Figura 25:** Pieza con función de pierna

**Figura 26:** Pieza de unión

La tercera pieza (figura 27) tiene la función de sujeción del servomotor con función del movimiento de cadera. Esta pieza tiene un tamaño de 30x41 para poder insertar verticalmente entre sus columnas un servomotor. Por otra parte, esta pieza tendrá dos orificios por los cuales se atornillará a la pieza anterior. También se ha tenido en cuenta el cableado del servomotor por lo que se ha diseñado un espacio suficiente para el paso del cableado. Finalmente, el servomotor se atornillará a las columnas por cuatro orificios en la parte superior de las mismas.

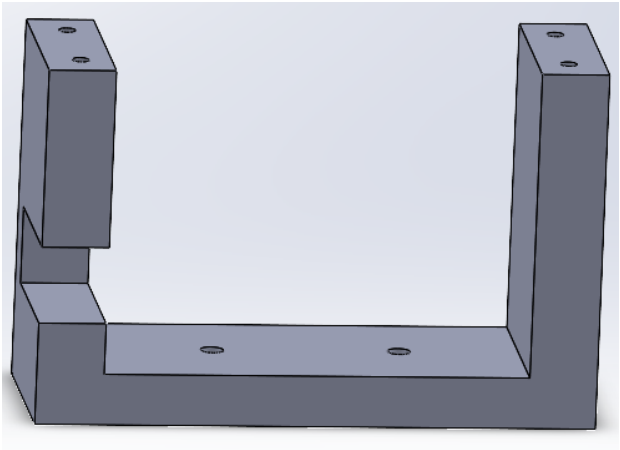


**Figura 27:** Pieza de soporte

#### 6.2.1.3. Tren superior

El tren superior contiene tres piezas y la primera de ellas (figura 28) ejerce la función de unión entre el tren superior y el tren inferior además de las dos piernas. Esta pieza se trata de una placa de 4 mm con los orificios necesarios para atornillar tanto los servos de las piernas como el tren superior como la placa PCA9685. Finalmente, esta pieza se ha diseñado para albergar la batería por lo que se ha dejado un espacio en la parte delantera para este propósito.

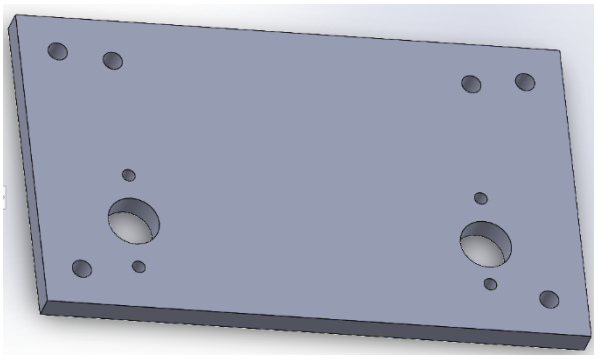


**Figura 28:** Base cadera

La siguiente pieza (figura 29) se ha diseñado para albergar el microcontrolador además de otras funciones estructurales. Para ello se ha diseñado en dos partes, una primera parte menos ancha que unirá con la pieza anterior y albergará la batería y una segunda parte más amplia que contendrá en microcontrolador y parte de dos servomotores. En lo que a la parte inferior se refiere se han diseñado dos pestañas con orificios en el interior para poder atornillar la pieza a la cadera. Por otra parte, se han incorporado dos orificios, uno en la parte trasera y otro en el lateral para permitir el paso del cableado de alimentación y de la PCA9685. Por otro lado, la parte superior tiene mayor anchura y profundidad con el objetivo de albergar el microcontrolador y parte de dos servomotores. Por otra parte, se han diseñado una serie de orificios en la parte trasera y lateral para atornillar el microcontrolador y los dos servos. Además, se han diseñado diversas aperturas, dos de ellas en el lateral para poder acceder a los puertos del microcontrolador y otra en la parte superior para encajar un servomotor de tamaño reducido.

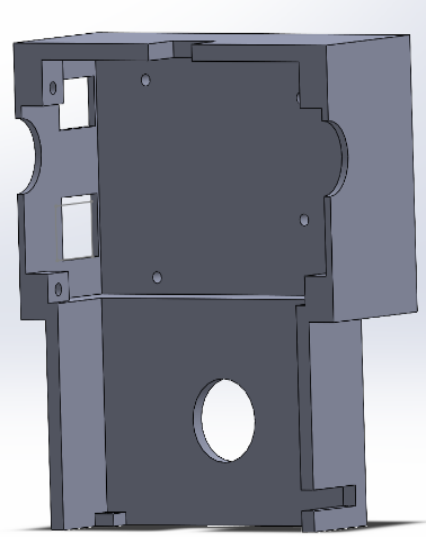


**Figura 29:** Pieza espalda

La siguiente pieza (figura 30) es complementaria a la pieza anterior, es decir, encajan perfectamente una con otra y comparten los orificios tanto de los servos laterales como del superior. Esta pieza está unida a la base inferior de la misma manera que la anterior y se une a la espalda atornillando los servos como se especifica más adelante. Por otra parte, en los laterales de la parte inferior se han diseñado unas aperturas por la cual se introduce la batería. Finalmente, Esta pieza tiene una apertura frontal para poder acceder a la electrónica fácilmente.

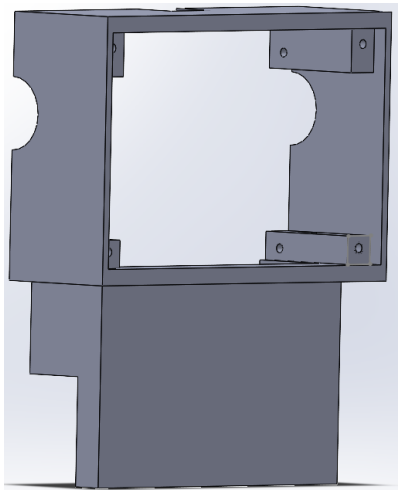


**Figura 30:** Pieza de torso

#### 6.2.1.4. Articulaciones

En lo referente a los brazos están compuestos de dos piezas, una primera pieza (figura 31 a)) con función de hombro y albergar el siguiente servomotor y una segunda pieza (figura 31 b)) imitando un puño. La primera pieza está unida al servomotor mediante la pieza mostrada en la figura 31 la cual esta atornillada al hombro. Por otra parte, se ha diseñado un espacio en el que insertar y atornillar el servomotor. En lo que al puño se refiere se tuvo en cuenta la longitud de este y un desplazamiento al centro con la intención de que los golpes sean más precisos.

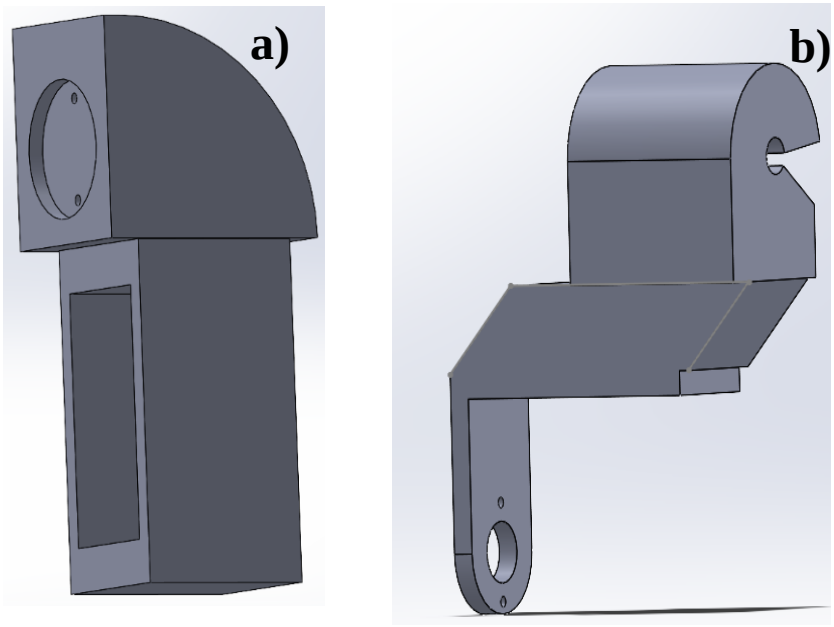



***Figura 31:*** *Piezas de las extremidades: a) Brazo, b) Puño*

Por otra parte, se ha diseñado una tapa para de la apertura frontal (figura 32 a)). Esta tapa se atornillará por las cuatro esquinas a la pieza frontal con tal de poder acceder a la electrónica fácilmente. Por último, se ha diseñado una pieza que simula la cabeza (figura 32 b)) del robot además de soporte del sensor de ultrasonidos. Para ello se ha diseñado la pieza con dos orificios en la parte frontal por los cuales se encajan los componentes del sensor y un orificio para atornillar la pieza al servomotor del cuello para unirse al cuerpo.

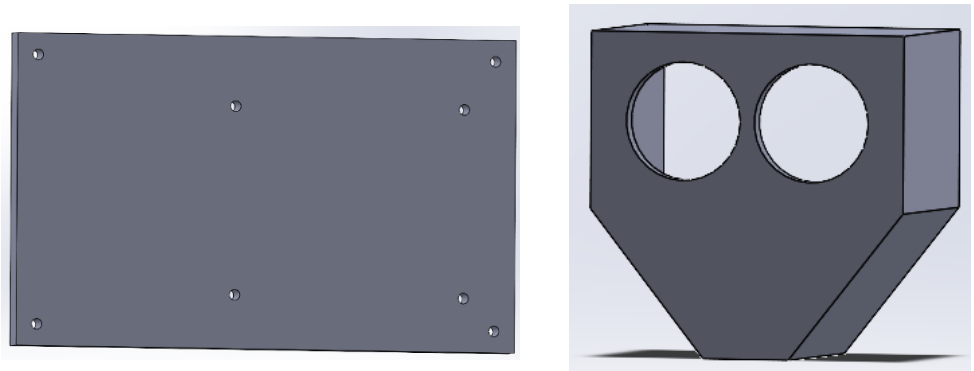


***Figura 32:*** *Piezas finales: a) tapa apertura frontal, b) cabeza*

#### 6.2.1.5. Montaje

En este apartado se procede a detallar el proceso de montaje del robot. Durante este proceso se han utilizado tornillería M6 de diversa longitud exceptuando en las uniones de los servomotores (figura 31) y al atornillar los servomotores que se utiliza tornillería propia del servomotor.

##### 6.2.1.5.1. Montaje tren inferior

El montaje comienza por el tren inferior, en concreto el primer paso consiste en insertar un servomotor en cada pie. Para ello se encaja en el espacio indicado en la figura 37 tratando que los cables se introduzcan en el espacio designado para después atornillar el servo y la pieza por los agujeros indicados en la figura 36 y 37.

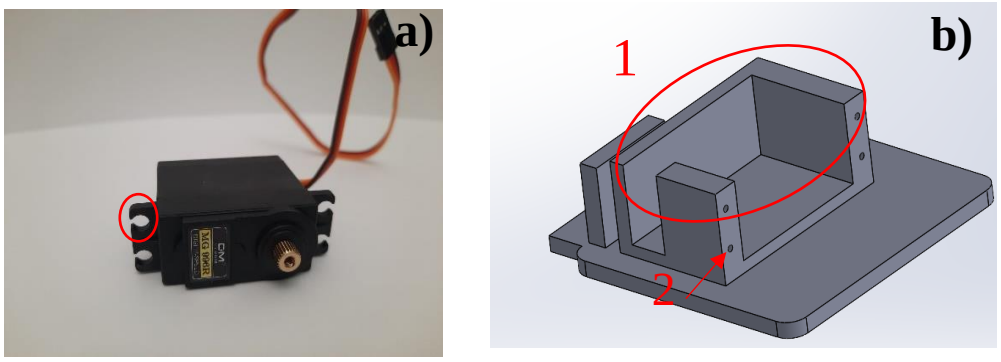



***Figura 33:*** *Encaje del servomotor: a) Orificio para atornillar servomotor, b) Espacio de encaje del servomotor (1) orificio para atornillar (2)*

A continuación, se procede a unir la pierna (figura 25) al servo y la pieza anterior, para ello se utiliza la unión del servomotor (figura 26) la cual se atornilla a los orificios indicados en la figura 34. Esta pieza se puede atornillar al eje del servomotor lo cual permite el movimiento de la pierna. Por último, la pierna se encaja y atornilla en el espacio señalado en la figura 35.

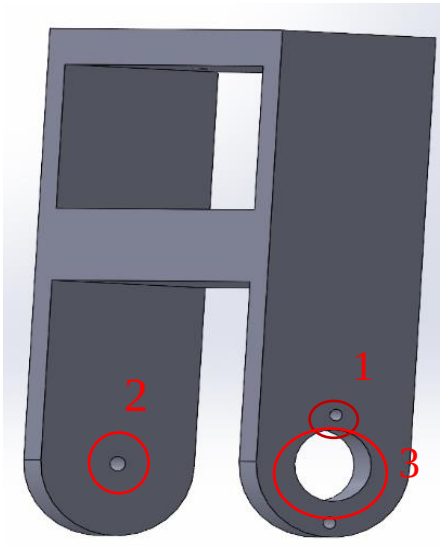



**Figura 34:** Orificio para atornillar la pieza del eje (1) orificio para atornillar al pie (2) orificio del eje del servomotor

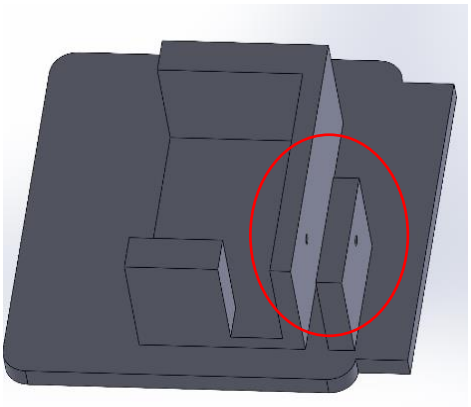


**Figura 35:** Zona de encaje de la pierna

El resultado final se observa en las figuras 36 y 37:

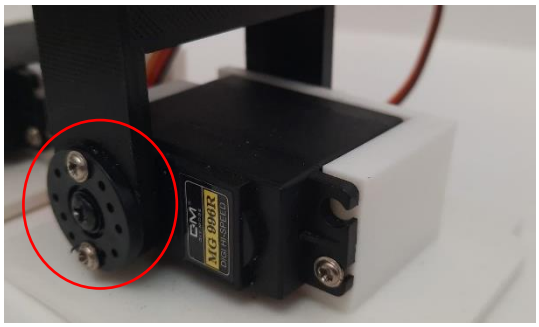


**Figura 36:** Resultado final (parte delantera)

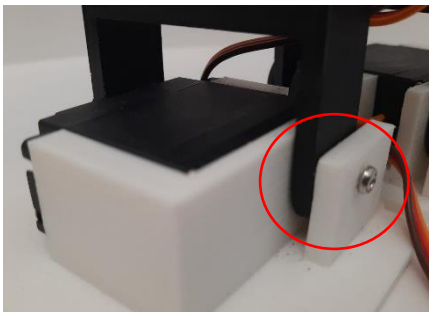


**Figura 37:** Resultado final (parte trasera)

La siguiente pieza a montar es la unión del servomotor (figura 26) con la pierna atornillándose por los orificios señalados en las figuras 38 y 39. Por otra parte se atornilla un servomotor con los orificios del servomotor señalados en la figura 36 a los orificios de la pieza soporte de la figura 43.

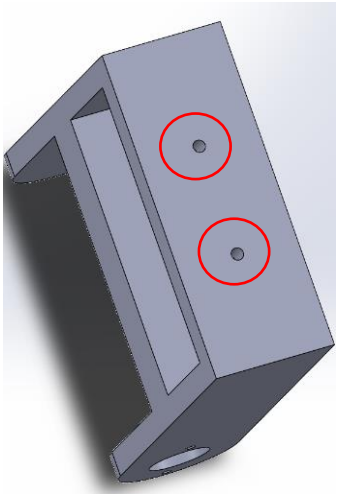


**Figura 38:** Orificios para atornillar la siguiente pieza

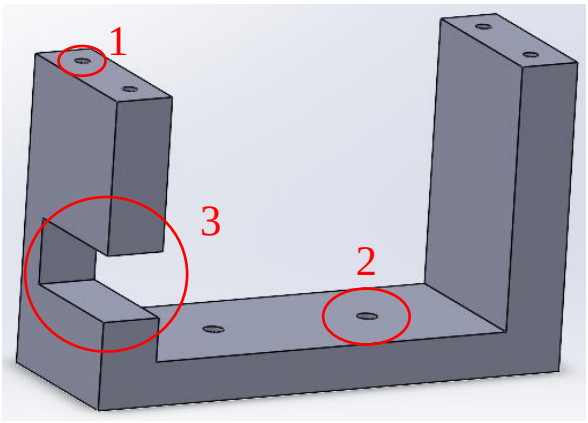



**Figura 39:** Orificios para atornillar el servomotor (1), orificios para atornillar la anterior pieza (2), espacio para cableado (3)

Realizando este montaje dos veces se finaliza el montaje del tren superior dando el resultado que se presenta en la figura 40:

**Figura 40:** Resultado montaje tren inferior

#### 6.2.1.5.2. Montaje tren superior

A continuación se procede al montaje del tren superior y la unión con la parte inferior .En primer lugar se inserta la union del servomotor (figura 26) en el orificio indicado en la figura 41 y se atornilla a los orificios indicados para segidamente insertar la pieza en el eje del servomotor. Por último se atornilla la placa PCA9685 por los orificios de la cardera indicados en la figura 41.

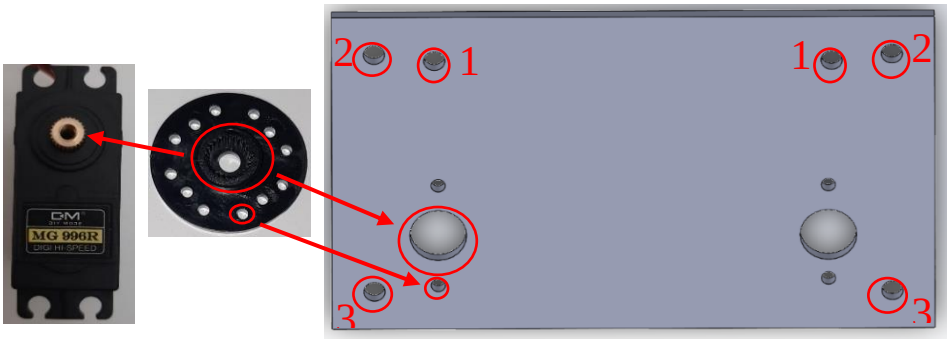



**Figura 41:** Proceso de unión del tronco superior al inferior. Orificios para atornillar la PCA9685(1). Orificios para atornillar la espalda (2). Orificios para atornillar el torso (3).

La siguiente pieza a montar es la espalda (figura 29) la cual se atornilla a la cadera (figura 28) utilizando los orificios indicados en las figuras 42 y 43, cabe destacar que se ha de atornillar desde la parte inferior de la cadera. Por otro lado, se atornilla el microcontrolador utilizando los orificios correspondientes (figura 42). Además, se ha de atornillar los dos servomotores de manera vertical a los laterales de la pieza (utilizando tuercas M6 en esta ocasión únicamente), utilizando los orificios indicados en la figura 42, de tal manera que la mitad del servo sobresalga de la pieza, para ser atornillada en otra pieza.

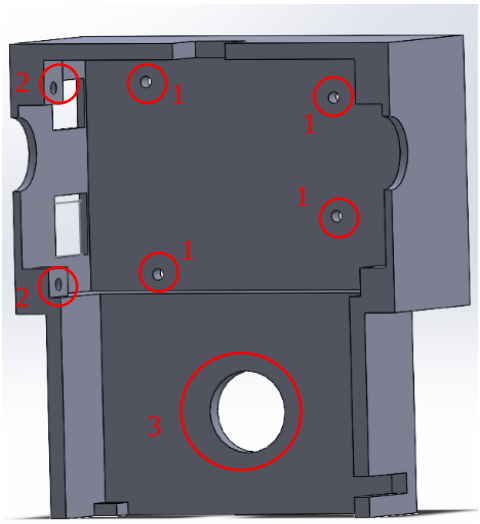



**Figura 42:** Orificio de unión del microcontrolador (1). Orificios de unión de los servomotores (2). Orificio de cableado (3)

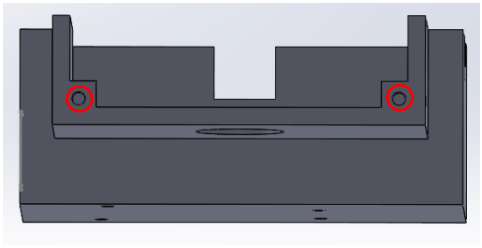


**Figura 43:** Orificios de unión con la cadera

A continuación, se ha de montar la unidad de potencia. Todos los componentes de la unidad de potencia se han de pegar a la parte trasera utilizando silicona. En primer lugar, se pega el adaptador Jack y se conecta, utilizando cableado macho-macho, a la entrada del convertidor DC-DC. Una vez realizada la conexión se conecta la batería para ajustar el voltaje de salida del convertidor a 5 V. El cableado de la salida del convertidos ha de pasarse por el orificio mostrado en la figura 42. A continuación se suelda el cableado a una PCB como se representa en la figura 44, para seguidamente soldar cuatro cables macho-hembra y un cable macho-macho a la línea positiva y tres cables macho-hembra y uno macho-macho en la línea negativa. Una vez soldado todo el cableado se realizan todas las conexiones de alimentación que necesiten los componentes como el microcontrolador, el módulo bluetooth, la placa PCA9685 y el sensor de ultrasonidos. Además, se realizan todas las conexiones entre los componentes electrónicos.

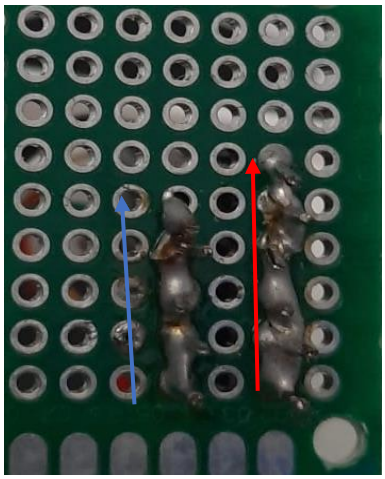


**Figura 44:** Conexión unidad de alimentación

Para finalizar el montaje del tren superior se ha de atornillar el torso (figura 30) a la cadera de la misma manera que se atornilló la espalda. Para encajar completamente el torso se han de atornillar los orificios del servomotor a los orificios laterales de la pieza (utilizando tornillería M6), el resultado final se observa en la figura 45.

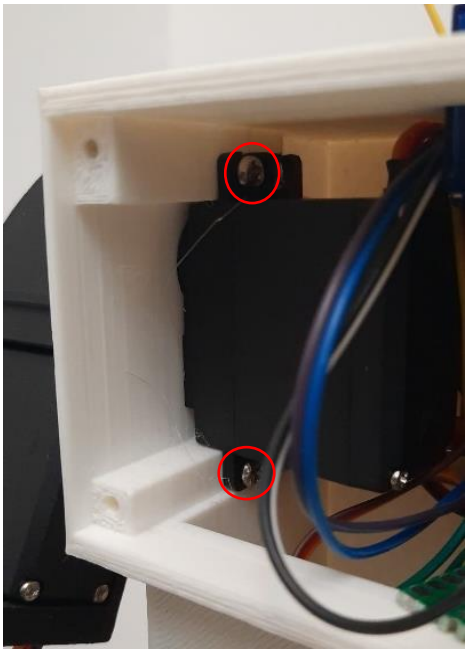


**Figura 45:** Resultado final

#### 6.2.1.5.3. Montaje de las extremidades

A continuación, se han de montar las extremidades. Se comienza con los brazos (figura 31 a)) a los cuales se les encaja y atornilla un servomotor utilizando los orificios señalados en la figura 46 b). Seguidamente se atornilla la unión del servomotor (figura 26) a la pieza del brazo por los orificios marcados en la figura 46 a). Finalmente se encaja la unión del servomotor al eje de este.

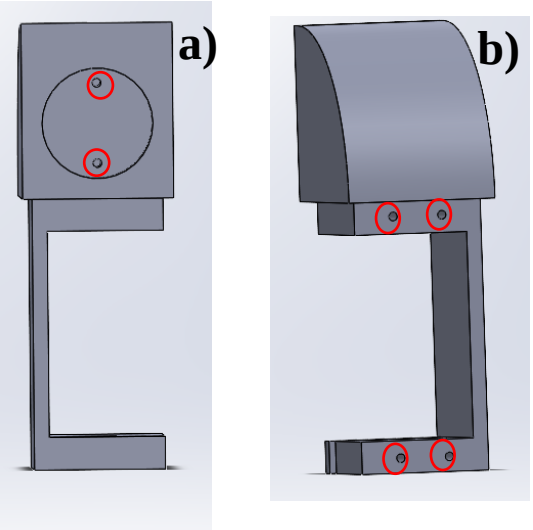



***Figura 46:*** *a) Orificios de encaje de la unión del servomotor, b) orificios de encaje del servomotor*

Para finalizar el montaje de los brazos se ha de atornillar la unión del servomotor (figura 26) a los puños (figura 31 b)) por la parte exterior de estos, utilizando los orificios indicados en la figura 47. Una vez atornillados la unión y la pieza, esta se encaja y atornilla al eje del servomotor. El resultado se observa en la figura 48.

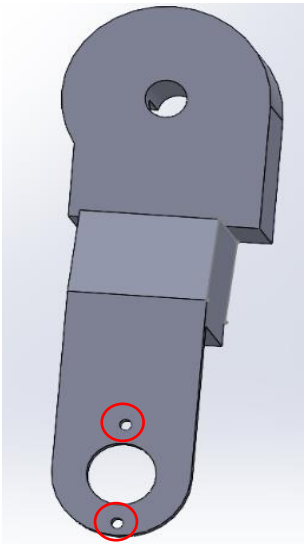


**Figura 47:** Orificios para atornillar la unión del servomotor

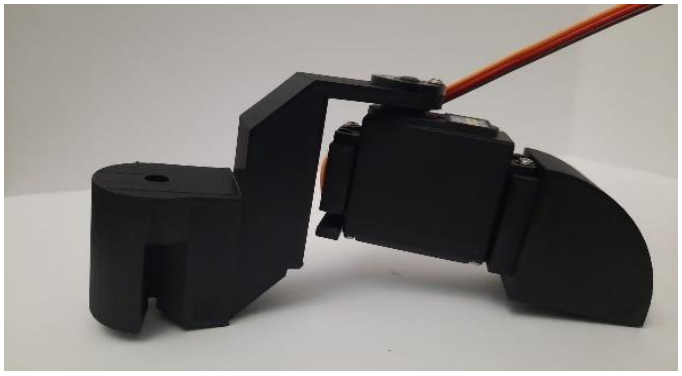


**Figura 48**: Resultado de la unión

La siguiente articulación corresponde al cuello del robot el cual sostiene la cabeza de este (figura 32 b)). En primer lugar, se inserta el servomotor modelo SG90 en la apertura del tren superior y se adhiere con silicona. Para fijar la cabeza al servomotor se utiliza la pieza mostrada en la figura 49 y se atornilla a los orificios indicados en la figura 49. Finalmente se adhiere el sensor de ultrasonidos con silicona.

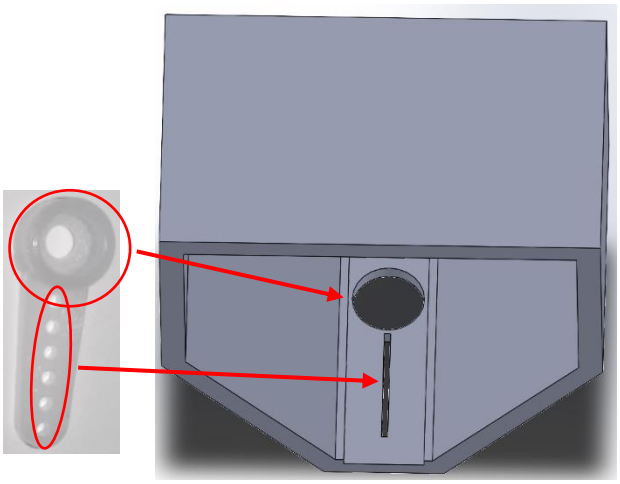


**Figura 49:** Proceso de encaje de la cabeza

Para finalizar el montaje se encajan las articulaciones superiores y se atornilla la tapa del torso (figura 32 a)) dando como resultado lo que se observa en la figura 50.

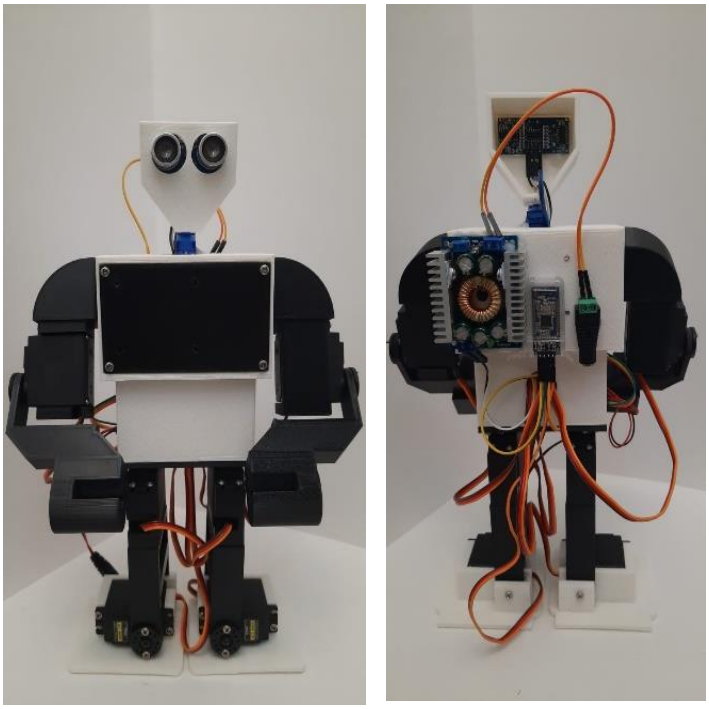


**Figura 50:** Resultado del montaje final

### 6.2.2. Software

Durante el desarrollo de este proyecto se han realizado dos programaciones distintas, por una parte, se desarrolla la programación del robot y por otra la programación de la aplicación. En lo referente a la programación del robot se han realizado tres programas distintos, una de control remoto y dos aplicaciones autónomas. En lo referente a la aplicación se ha programado una única aplicación para el control remoto utilizando Thunkable X.

#### 6.2.2.1. Programación del robot

En lo referente a la programación del robot, se han creado diversas funciones para el movimiento del robot. En primer lugar, se ha creado una función que permite mover cualquier servomotor a un ángulo determinado denominada “WriteServo”. A continuación, se ha creado una función que coordina el movimiento de diversos servomotores al mismo tiempo en la que se permite controlar el tiempo que se tarda en realizar el movimiento con tal de realizar movimientos más suaves.

Por otra parte, se han creado diversas variables o tipos de variables. En primer lugar, se ha definido en estructura denominada “RobotServo” que contiene datos de interés de los servos tales como ángulos máximos y mínimos el pin al cual está conectado y el offset de error. Por otra parte, se han creado dos matrices que contienen los ángulos de cada uno de los servomotores correspondiente a una configuración. Una de las matrices será la configuración anterior (q0) y la otra la configuración deseada a continuación (qn).

Utilizando estas funciones y variables se han programado diversos movimientos utilizando el siguiente método. En primer lugar, se establece la configuración deseada en la matriz correspondiente utilizando la función “Change_vector” A continuación, se ejecuta utilizando la función “coordinateAbsJ” introduciendo en la configuración anterior, la siguiente configuración y el tiempo de ejecución estimado. Por último, se actualiza la matriz de variables anteriores con la configuración recién alcanzada.

##### 6.2.2.1.1. Función “WriteServo”

A continuación, se detallarán cada una de las funciones. En primer lugar, la función “WirteServo” (figura 51) tiene como entrada el servomotor que se desea mover y el ángulo al que se desea mover. Con los datos del servomotor y el ángulo deseado se calcula el ancho de pulso necesario para que el servo se establezca en el ángulo deseado. Seguidamente se establece la señal PWM en el pin correspondiente.

```
void writeServo(RobotServo_t &servo, int angle)
{
  /*
   * Entrada: Servo a mover,ángulo deseado
   * Salida: ninguna
   *
   * Mueve el servo al ángulo deseado
   */
  int pulse_width;
  angle=constrain(angle,servo.min_pos,servo.max_pos);
  pulse_width = map(angle+servo.offset, 0, 180, MIN_PWM, MAX_PWM);
  servos.setPWM(servo.pin,0,pulse_width);
}
```

**Figura 51:** Función "writeServo"

#### 6.2.2.1.2. Función “CoordianteAbsJ”

Utilizando la función “WriteServo” se ha definido la función “coordinateAbsJ” (figura 52) la cual tiene como entrada todos los servos la configuración actual, la configuración deseada y el tiempo en el que se realiza el cambio de configuración. Con estos datos se calcula el ángulo de cada uno de los servos en intervalos de 1 ms. Para calcular este ángulo se necesita saber los parámetros a, b, c y d. Finalmente se calcula el ángulo necesario en esa iteración para el servo correspondiente utilizando los parámetros calculados anteriormente.

```
void coordinateAbsJ(RobotServo_t robotServos[JOINTS], const double q0[JOINTS], const double qT[JOINTS], const double T)
{
  /*Entrada: servos, posición de origen, posición final, tiempo de ejecución
   *Salida: Ninguna
   *
   *Con los datos proporcionados se calcula una trayectoria y se ejecuta
   */
  double a[JOINTS],b[JOINTS],c[JOINTS],d[JOINTS],q,t,t0;
  t0=millis()/1000.0;
  t=0.0;
  while(t<T)
  {
    for (int i=0;i<JOINTS;i++)
    {
      //TODO: cálculo de la trayectoria.
      a[i] = -(2*(qT[i]-q0[i]))/pow(T,3);
      b[i] = (3*(qT[i]-q0[i]))/pow(T,2);
      c[i] = 0;
      d[i] = q0[i];
      q = a[i]*pow(t,3) + b[i]*pow(t,2) + c[i]*t + d[i];
      //Ejecución
      writeServo(robotServos[i],(int)q);
    }
    delay(20);
    t=millis()/1000.0-t0;
  }
}
```

**Figura 52:** Función "coordinateAbsJ"

#### 6.2.2.1.3. Función “ChangeVector”

Por otra parte, se ha creado una función la cual sustituye los valores de la matriz para la siguiente configuración (figura 53). Para ello requiere de entrada la matriz qn y cada uno de los ángulos de la configuración para seguidamente actualizar uno a uno los valores de la matriz qn.

```
void Change_vector(double qn[],double j1,double j2,double j3,double j4,double j5,double j6,double j7,double j8)
{
 /*
 * Entrada: Un vector vacio a rellenar, los ángulos deseados de cada servomotor
 * Salida: Ninguna
 *
 * Renueva el vector a la siguiente configuración deseada
 */
  qn[0] = j1;
  qn[1] = j2;
  qn[2] = j3;
  qn[3] = j4;
  qn[4] = j5;
  qn[5] = j6;
  qn[6] = j7;
  qn[7] = j8;
}
```

**Figura 53:** Función para el cambio de parámetros del vector

#### 6.2.2.1.4. Función “distancia”

Además, se ha creado una función cuya funcionalidad es obtener la distancia del sensor de ultrasonidos. Esta función representada en la figura 54 mide el intervalo de tiempo entre el envío y la recepción de un impulso de ultrasonidos para averiguar la distancia recorrida por el ultrasonido.

```
int distancia()
{
  /*
   * Entrada: Ninguna
   * Salida: distancia medida
   *
   * Mide la distancia detectada por el sensor de ultrasonidos
   */
  int distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  return distance;
}
```

**Figura 54:** Función de medición de la distancia

#### 6.2.2.1.5. Funciones de movimiento

Con estas funciones como base se han desarrollado quince movimientos distintos incluyendo la imitación de los golpes (figura 55), los bailes (figura 56) o las diversas direcciones en la que puede andar el robot.

```
void Gancho_Derecha()
{
 /*
 * Entrada:Ninguna
 * Salida Ninguna
 *
 * Ejecuta una imitación de un gancho de derechas
 */
  Change_vector(qn,90,90,90,90,80,85,40,50);
  coordinateAbsJ(full_robot,q0,qn,0.2);
  for (int i = 0;i<JOINTS;i++)
  {
    q0[i]=qn[i];
  }
  delay(300);

  Change_vector(qn,90,90,105,105,60,180,40,160);
  coordinateAbsJ(full_robot,q0,qn,0.2);
  for (int i = 0;i<JOINTS;i++)
  {
    q0[i]=qn[i];
  }
  delay(300);
  guard();
}
```

**Figura 55:** Función de imitación de un gancho de derechas

```
void Lose_Dance()
{
 /*
 * Entrada:Ninguna
 * Salida Ninguna
 *
 * Ejecuta un baile de derrota
 */
  for (int i = 0;i<3;i++)
  {
    Change_vector(qn,110,110,90,90,120,65,90,90);
    coordinateAbsJ(full_robot,q0,qn,1);
    for (int i = 0;i<JOINTS;i++)
    {
      q0[i]=qn[i];
    }
    delay(500);
    Change_vector(qn,80,80,90,90,120,65,40,150);
    coordinateAbsJ(full_robot,q0,qn,1);
    for (int i = 0;i<JOINTS;i++)
    {
      q0[i]=qn[i];
    }
  }
}
```

**Figura 56:** Función que ejecuta un baile de derrota

A continuación, se enumeran todos los movimientos que puede realizar el robot:

- Paso hacia adelante
- Paso en diagonal dirección adelante-derecha
- Paso en diagonal dirección adelante-izquierda
- Paso hacia detrás
- Paso en diagonal dirección atrás-derecha
- Paso en diagonal dirección atrás-izquierda
- Giro dirección derecha
- Giro dirección izquierda
- Jap derecha
- Jap izquierda
- Gancho de derechas
- Gancho de izquierdas
- Baile de derrota
- Baile de victoria
- Posición de guardia

#### 6.2.2.1.6. Aplicación de control remoto

Utilizando todas las funciones y movimientos que se han descrito anteriormente se han desarrollado tres aplicaciones distintas. La primera de ellas consiste en una aplicación de control remoto utilizando cualquier dispositivo móvil (figura 57). La programación de la aplicación de móvil se detallará de manera más detallada más adelante, sin embargo, cabe destacar, el funcionamiento de esta. La aplicación envía mediante bluetooth un carácter que es recibido por el módulo HM-10 y este carácter se gestiona mediante el software explicado a continuación.

En primer lugar, se incluyen las librerías necesarias para esta aplicación. En este caso se trata de las funciones “Wire” y” Adafruit_PWMServoDriver”, para el control de los servos y “SoftwareSerial” para el uso del puerto serie y bluetooth. A continuación, se configuran los puertos que se utilizaran para controlar los servos y el módulo bluetooth definiendo estos puertos mediante las funciones “Adafruit_PWMServoDriver”, para los servos, y “mySerial”, para el bluetooth.

Seguidamente se definen diversas variables que se van a utilizar en el programa. En primer lugar, se definen constantes como el número de articulaciones o las frecuencias PWM mínimas y máximas. A continuación, se define un tipo de estructura que se utiliza para definir parámetros necesarios en el programa. Por último, se definen las dos matrices “qn” y “q0” explicado anteriormente y se definen todos los parámetros de los servos.

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h> 1
#include <SoftwareSerial.h>

// Configuración de servos y Bluetooth
Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x40); 2
SoftwareSerial mySerial(7, 6); //RX, TX

// Definición de constantes
#define MIN_PWM 130
#define MAX_PWM 570  3
#define JOINTS 8

//Definición de variables de los servos
typedef struct
{
  uint8_t pin;
  int offset;
  int min_pos;
  int max_pos;
} RobotServo_t;

//Definición de posiciones
double q0[JOINTS] = {90, 90, 90, 90, 80, 120, 40, 150};
double qn[JOINTS];                                          4
//Definicion de los servos
RobotServo_t full_robot[JOINTS] = {{10, 10, 0, 140}, {11, 3, 40, 180},
```

**Figura 57:** Inclusión de librerías (1). Configuración de puertos (2). Definición de constantes (3). Definición de variables (4)

A continuación se procede a realizar al configuración inicial del robot en la función "setup()" (figura 58) de Arduino. En esta función se realizan diversos procedimientos, el primero de ellos es iniciar los servomotores utilizando la función "servos.begin()" para a continuación configurar la frecuencia de funcionamiento de los servomotores a 60 Hz utilizando la función "servos.setPWMFreq()". En segundo lugar, se realiza la configuración del puerto serie para poder visualizar datos desde la computadora, para ello se utilizan las funciones "Serial.begin()" y "mySerial.begin()". Por último, se coloca el robot en la posición inicial utilizando una de las funciones c

```
void setup() {
  //Configuración de la frecuencia de los servos
  servos.begin();          1
  servos.setPWMFreq(60);
  //Configuración del puerto serie para visualizar los datos
  Serial.begin(9600);
  mySerial.begin(9600);    2
  Serial.println("Empiezo a recibir caracteres!");
  //Posición inicial
  guard();
  delay(3000);             3
}
```

**Figura 58:** Configuración de la frecuencia de los servos (1). Configuración del puerto serie (2). Posición de inicio (3)

Por último se crea la aplicación en la función "loop()" de Arduino. Para ello en primer lugar se comprueba si el puerto serie está o no disponible utilizando la sentencia "if". En caso de estar disponible se recibe y almacena el carácter enviado en una variable llamada "c". A continuación, se realiza una sentencia "switch" en la que en función del carácter se ejecuta un movimientos o secuencia de estos. La correspondencia de carácter y movimientos es al siguiente:

- "f": Movimiento hacia adelante
- "l": Giro hacia la derecha

- “c”: Posición de guardia
- “r”: Giro hacia la izquierda
- “b”: Paso hacia atrás
- “o”: Gancho de derechas
- “z”: Jap derecha
- “k”: Gancho de izquierdas
- “j”: Jap izquierda
- “y”: Baile de derrota
- “t”: Baile de victoria
- “x”: Combo de golpes nº1
- “p”: Combo de golpes nº2
- “v”: Combo de golpes nº3
- “q”: Combo de golpes nº4
- “g”: Paso en diagonal dirección adelante-izquierda
- “h”: Paso en diagonal dirección adelante-derecha
- “n”: Paso en diagonal dirección atrás-izquierda
- “m”: Paso en diagonal dirección atrás-derecha

#### 6.2.2.1.7. Aplicación autónoma 1

Esta aplicación autónoma tiene como objetivo simular un comportamiento de un boxeador agresivo. Para ello el robot avanzará hasta detectar un objeto y le asesta una serie de golpes. Una vez realizados los golpes si sigue detectando un objeto ejecutara un baile de derrota. Si por lo contrario deja de detectar el objeto ejecutará un baile de victoria.

El comienzo del programa es el mismo que para la anterior aplicación a excepción de la configuración de los pines del sensor de ultrasonidos y tres variables necesarias para la utilización del sensor de ultrasonidos una para almacenar la duración del impulso de ultrasonidos, y otras dos para almacenar la distancia calculada.

La siguiente parte del programa es la configuración inicial en la función “setup” de Arduino. Esta función es muy similar a la función “setup” de la anterior aplicación a excepción de la configuración del sensor de ultrasonidos como se muestra en la figura 59.

```
void setup() {
  servos.begin();
  servos.setPWMFreq(60);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
  guard();
  e = 0;
  delay(3000);
}
```

**Figura 59:**Función setup de la primera aplicación autónoma

La principal diferencia en el programa radica en la función “loop”, que ya no depende del carácter enviado por la aplicación, sino que es completamente autónoma. Para ello se han realizado una serie de sentencias condicionales las cuales determinan el comportamiento del robot:

- En el caso que la distancia sea menor a 12 cm y no se haya detectado objeto anteriormente se realizan dos golpes. Para determinar si se ha detectado o no un objeto anteriormente se crea una variable “e” la cual se le suma 1 cada vez que se detecte un objeto. Además, la variable “e” ascenderá de valor.
- En el caso que la distancia sea menor que 12 cm y si se haya detectado objeto anteriormente, es decir; “e” tiene valor de 1, el robot realiza otra serie de golpes. Además, la variable “e” ascenderá de valor.
- En el caso de que la distancia sea menor que 12 cm y el valor de “e” sea 2 realiza una serie de golpes finales. Además, la variable “e” ascenderá de valor.
- En el caso de que la distancia sea menor que 12 cm y el valor de “e” sea 3 el robot realiza un baile de derrota. Además, la variable “e” ascenderá de valor.
- En el caso de que la distancia sea mayor de 12 cm y el valor de “e” sea mayor que 0, es decir, ha detectado objeto anteriormente, realiza un baile de victoria. Además, la variable “e” ascenderá de valor.
- Finalmente, si no se cumple ninguna de las condiciones el robot continuará hacia adelante y el valor de “e” será 0.

#### 6.2.2.2.8. Aplicación autónoma 2

A diferencia de la anterior aplicación esta intenta imitar un comportamiento defensivo. Para ello el robot permanecerá en guardia hasta detectar un objeto, una vez detecta el objeto el robot retrocede dando golpes. En el caso de seguir detectando objeto el robot realiza un baile de derrota, en el caso contrario realizará un baile de victoria.

Para esta aplicación se utiliza el código de la aplicación autónoma 1 exceptuando el bucle principal puesto que las sentencias condicionales son las siguientes:

- En el caso de que la distancia detectada por el sensor de ultrasonidos sea menor a 12 cm y que la variable “e” sea mayor a 0 y menor a 3, el robot retrocederá asestará dos golpes y volverá a retroceder. Además, la variable “e” ascenderá de valor.
- En el caso de que la distancia sea menor a 12 cm y la variable “e” sea igual a 3 el robot ejecutará una serie de golpes sin retroceder. Además, la variable “e” ascenderá de valor.
- En el caso en el que la distancia sea menor a 12 cm y la variable “e” sea igual a 4 el robot ejecutará un baile de derrota. Además, la variable “e” ascenderá de valor.
- En el caso en el que la distancia sea menor a 12 cm y la variable “e” sea mayor a 0 el robot ejecutará un baile de victoria. Además, la variable “e” ascenderá de valor.
- En el caso de que no se cumpla ninguna de las condiciones el robot se mantendrá en guardia y “e” tendrá valor 0

#### 6.2.2.3. Programación de la aplicación

A continuación, se detalla cómo se ha realizado la aplicación desde la cual se controla de manera remota el robot. Para la programación de esta aplicación se ha utilizado el programa Thunkable X. Este programa se encuentra en el navegador web y permite crear aplicaciones para móvil que pueden utilizarse en cualquier dispositivo asociando una cuenta de Google.

Esta aplicación permite crear una interfaz con diversas opciones como botones cuadros de texto etc… Por otra parte, Thunkable X permite una programación por bloques en la cual se le da funcionalidad a los elementos de la aplicación anteriormente diseñados.

Utilizando Thunkable X se ha diseñado una aplicación de control con una triple pantalla. La primera pantalla, presentada en la figura 60 a), consiste en un buscador de dispositivos Bluetooth con el objetivo de enlazar el dispositivo móvil al robot. A continuación, se encuentra la pantalla de inicio (figura 60 b)) cuya función únicamente es enlazar con la pantalla de control del robot. Por último, se encuentra la aplicación de control representada en la figura 60 c).

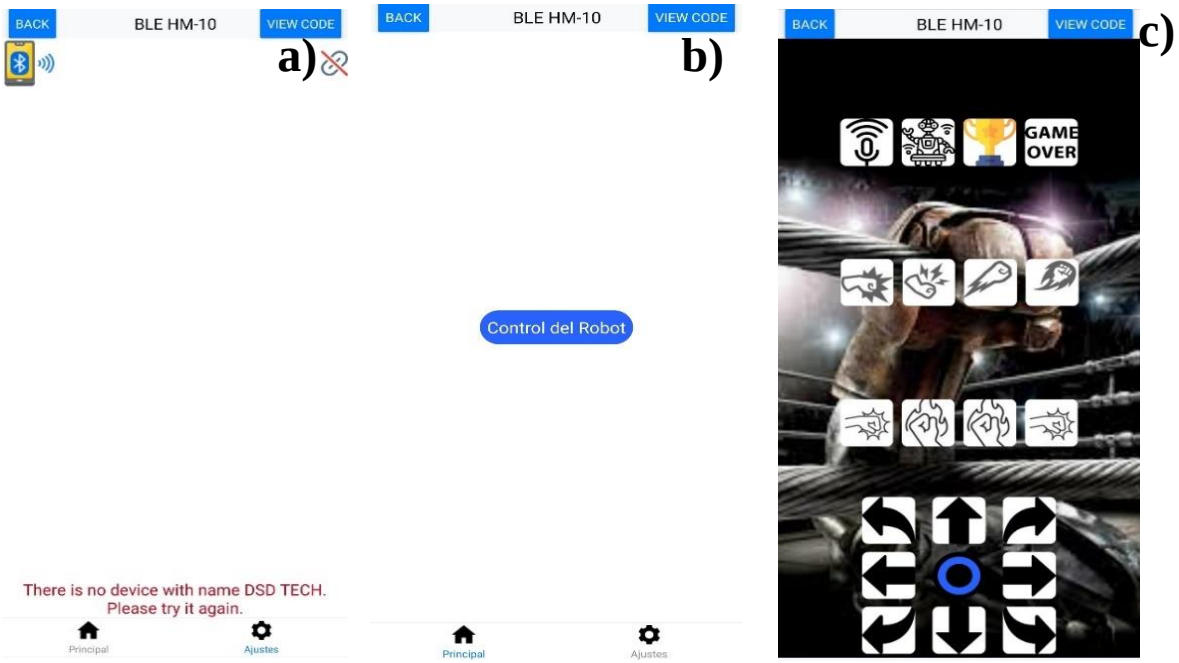



***Figura 60:*** *Aplicación de control remoto: a) Pantalla de configuración, b) pantalla de inicio, c) pantalla de control*

A continuación, se explicará la programación por bloques realizada en ente proyecto. Por una parte, se ha realizado una función la cual envía un carácter determinado a una dirección de bluetooth determinada (figura 61). Por otra parte, se han creado interacciones con cada uno de los botones las cuales utilizan la función anterior para enviar un carácter determinado, se puede observar un ejemplo en la (figura 62)

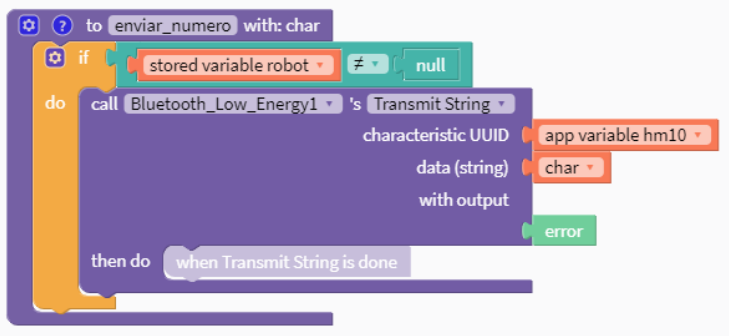



**Figura 61:** Función de envío por bluetooth

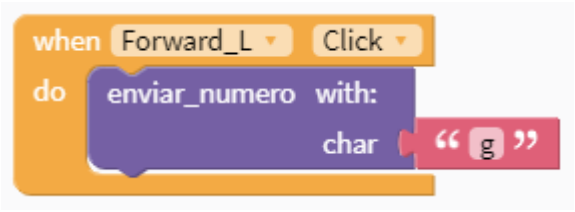



**Figura 62:** Ejemplo de envío de carácter

# 7. Cálculos

Los cálculos necesarios para la realización del proyecto son referidos al cálculo de las configuraciones del robot. El funcionamiento del robot está basado en matrices de configuración, es decir, una matriz que contiene los ángulos de cada uno de los servos que definen la posición del robot. Un movimiento está compuesto por diversas configuraciones que se ejecutan en un tiempo determinado.

Existen cinco tipos de movimientos que implican cada uno una serie de configuraciones que han sido calculadas individualmente. En concreto se han creado tres tipos de movimientos, movimientos rectos hacia adelante o hacia atrás, movimientos en diagonal en todas direcciones y por último giros a derecha o izquierda. Por otra parte, se han creado dos tipos de golpes y dos tipos de baile.

## 7.2. Movimientos en línea recta

En lo referente a los movimientos en línea recta, tanto adelante como hacia detrás, están compuestos por seis configuraciones. En ambos movimientos se sigue el mismo proceso de cálculo y de movimiento, sin embargo, la secuencia esta invertida.

En primer lugar, y tal como se muestra en la figura 63, se eleva el pie derecho poniendo a 80º ambos servos el servo izquierdo será sobre el que se incline el robot y el derecho empujará para favorecer esa inclinación, además se inclinan ambos servos de la cadera alrededor de 75º para poder enlazar correctamente los pasos. A continuación, el pie derecho vuelve a su posición original, por motivos estéticos, y se adelanta la cadera derecha poniendo ambos servos en una posición cercana a los 100º. Pese a que teóricamente se debería posicionar ambos servos al mismo ángulo, esto no se ha realizado por errores de offset. Seguidamente el pie izquierdo vuelve a su posición original para eliminar la inclinación del robot. A continuación, se inclina el pie derecho utilizando el izquierdo para empuje posicionando los servos alrededor de 110º. La quinta configuración regresa el pie izquierdo a su ángulo original y avanza la cadera con una posición de los servos de alrededor de 75º. Por último, el pie derecho regresa a su posición de origen para eliminar la inclinación del robot.

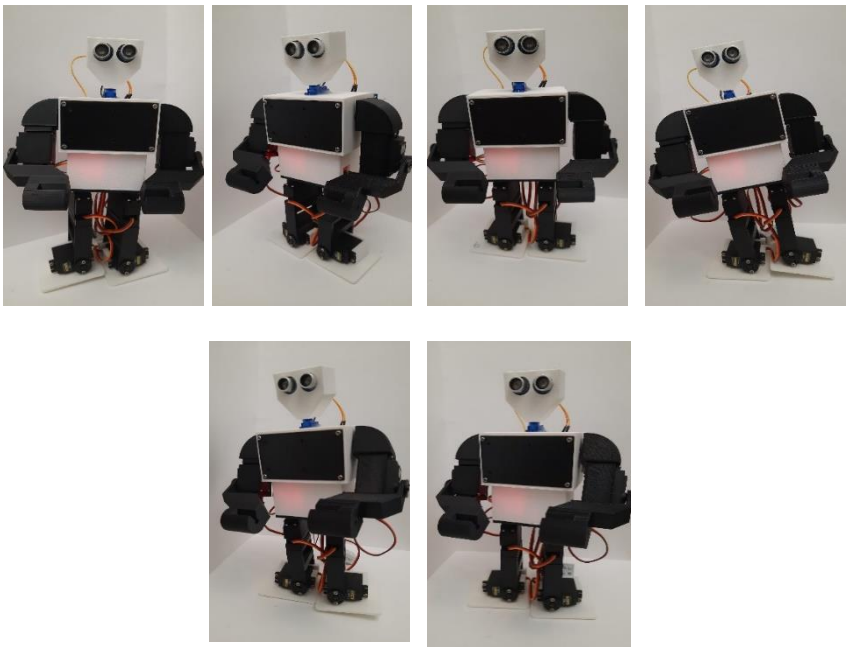


**Figura 63:** Secuencia paso hacia adelante

## 7.3. Movimientos en diagonal

En lo referente al movimiento en diagonal (figura 64) existen cuatro direcciones adelante-derecha, adelante-izquierda, atrás-derecha y atrás-izquierda y todas están compuestas por seis configuraciones. Estos movimientos se caracterizan por que el ángulo de la pierna de apoyo maría más o menos dependiendo de la dirección:

- Adelante-derecha: Mayor giro cuando apoya la pierna derecha
- Adelante-izquierda: Mayor giro cuando apoya la pierna izquierda
- Atrás-derecha: Mayor giro cuando apoya la pierna izquierda
- Atrás-izquierda: Mayor giro cuando apoya la pierna derecha

En lo referente al movimiento de todos los movimientos diagonales en primer lugar se levanta la pierna derecha poniendo ambos servos a 80º sirviendo la izquierda como apoyo y la derecha para inclinar el robot. Además, se inclina la cadera para enlazar los movimientos, para ello se aplican los siguientes ángulos:

- Adelante-derecha: pierna izquierda a 65º pierna derecha a 75º
- Adelante-izquierda: pierna izquierda a 75º pierna derecha a 65º
- Atrás-derecha: pierna izquierda a 100º pierna derecha a 95º
- Atrás-izquierda: pierna izquierda a 95º pierna derecha a 100º

A continuación, el pie que haya servido para favorecer la inclinación regresa a un ángulo de 90º. Además, se inclina la cadera en la dirección requerida con los siguientes ángulos:

- Adelante-derecha: pierna izquierda a 100º pierna derecha a 95º
- Adelante-izquierda: pierna izquierda a 95º pierna derecha a 100º
- Atrás-derecha: pierna izquierda a 65º pierna derecha a 75º
- Atrás-izquierda: pierna izquierda a 75º pierna derecha a 65º

La tercera configuración elimina la inclinación del robot posicionando el servo del pie de apoyo a 90º. Seguidamente en la cuarta configuración se levanta la pierna contraria a la anterior utilizando el mismo método, pero con ángulos alrededor de 110º grados.

En la quinta configuración se realizan los giros de cadera que se detallan a continuación:

- Adelante-derecha: pierna izquierda a 65º pierna derecha a 75º
- Adelante-izquierda: pierna izquierda a 75º pierna derecha a 65º
- Atrás-derecha: pierna izquierda a 65º pierna derecha a 75º
- Atrás-izquierda: pierna izquierda a 75º pierna derecha a 65º

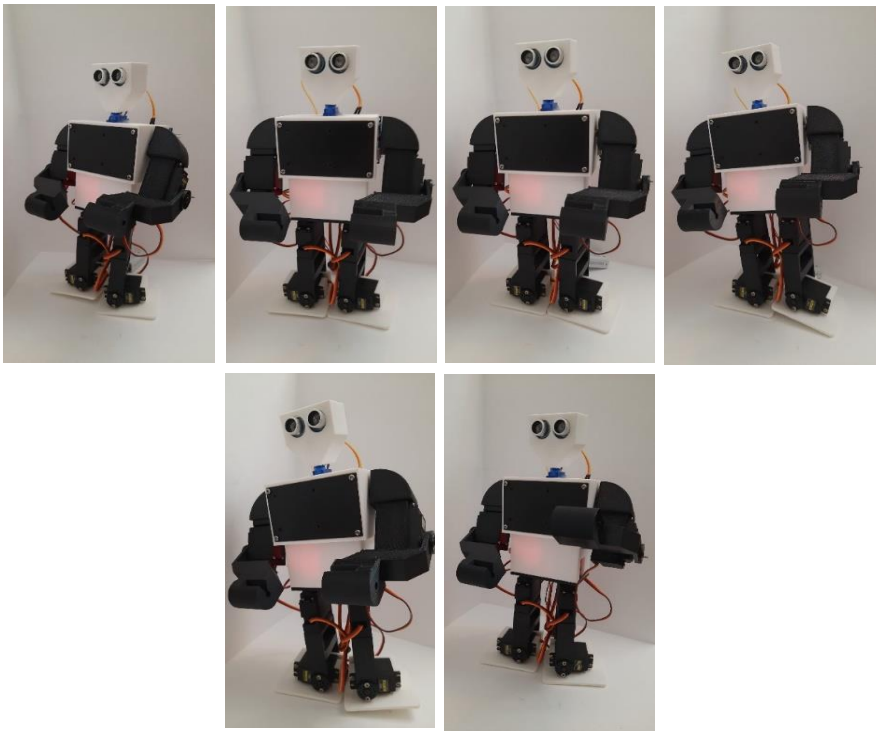


**Figura 64:** Secuencia de paso en diagonal hacia adelante izquierda

## 7.4. Giros

En el caso de los giros están compuestos de seis configuraciones y su comportamiento es distinto a los movimientos anteriores. En este caso la pierna de la dirección de giro retrocede al contrario de la pierna contraria la cual retrocede.

La primera configuración levanta el pie derecho utilizando el derecho como apoyo, en el caso de giro a la derecha, situando los servos alrededor de 110º y situando el servo de la pierna derecha a 80º dejando la izquierda estática para enlazar movimientos. En el caso del giro a la izquierda se sitúan los servomotores de los pies a 80º para levantar el pie derecho y se sitúa el servo de la cadera a 100º para enlazar movimientos.

La segunda configuración sitúa los pies elevados a 90º por motivos estéticos y sitúa los servos de la pierna de apoyo a 100º y 80º para el giro a la derecha e izquierda, respectivamente. Esto se debe a que de esta manera se orienta en la dirección requerida.

La tercera configuración elimina la inclinación del robot posicionando todos los servos de los pies a 90º. A continuación, en la cuarta configuración se levanta la pierna izquierda en el caso del giro

a la derecha con un grado de 80º en ambos servos de los pies, en el caso del giro a la izquierda se levanta la pierna derecha situando ambos servos a 110º.

La quinta configuración tiene como objetivo retrasar el pie de la dirección de giro con tal de lograr una mejor orientación, además el pie que se ha elevado vuelve a 90º por motivos estéticos. Para ello en el caso del giro a la derecha se inclina el servo de la pierna derecha a un ángulo de 80º dejando el izquierdo estático. Sin embargo, en al caso del giro a la izquierda se inclina el servo de la pierna izquierda 100º dejando la izquierda estática. Por último, se elimina la inclinación del robot situando los servos de los pies a 90º. A continuación, se presenta una secuencia del giro a la izquierda.

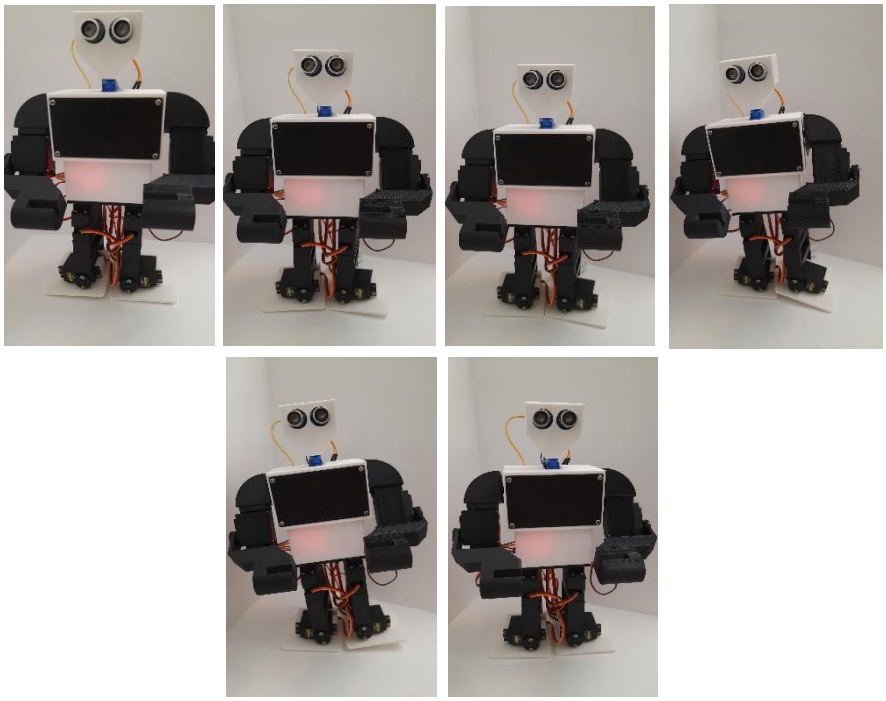


***Figura 65:*** *Secuencia de giro a la izquierda*

## 7.5. Imitación de golpes

En el caso de la imitación de golpes existen dos tipos de golpes que se pueden ejecutar en ambos brazos. El primer tipo de golpe pretende imitar un Jap que es un golpe frontal directo, en cambio el segundo golpe pretende imitar un gancho. El método de cálculo para ambos golpes se basa en un método experimental al igual que los movimientos.

### 7.5.2. Imitación de jap

En este golpe se pretende dar un golpe frontal directo desde cualquier posición (figura 66). Para ello lo que se persigue es colocar el brazo de manera perpendicular además de girar la cadera para favorecer el golpe al robot, por lo cual solo es necesario utilizar una sola configuración. En el caso del brazo derecho se han utilizado los siguientes ángulos:

- Pies: 90º
- Piernas: 105º
- Brazo derecho: 180º
- Brazo izquierdo: 60º

- Puños: 40º

Por contrario en la configuración del brazo izquierdo es:

- Pies: 90º
- Piernas: 75º
- Brazo derecho: 150º
- Brazo izquierdo: 30º
- Puños. 150º

A continuación, se presenta una secuencia explicativa del golpe:

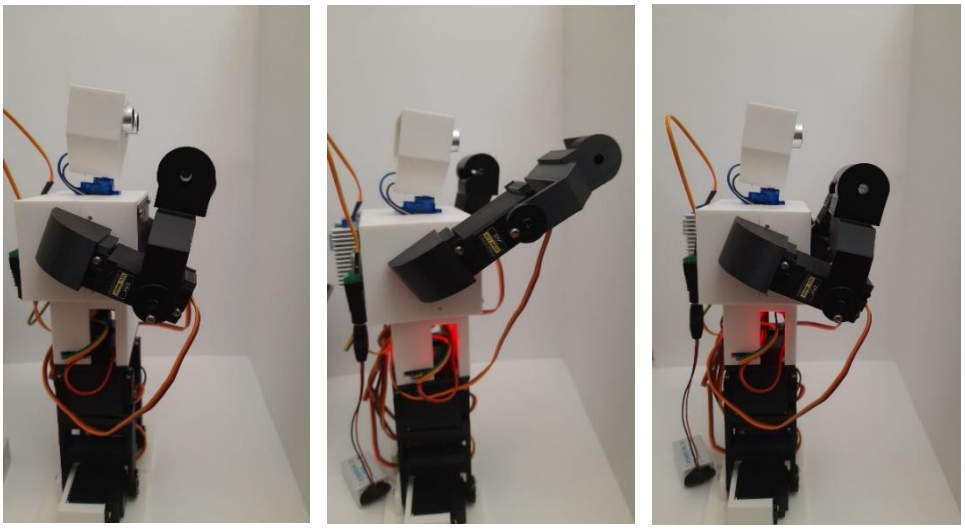


**Figura 66:** Imitación golpe jap

### 7.5.3. Imitación gancho

La imitación de este golpe (figura 67) es más compleja puesto que implica dos movimientos, uno hacia abajo y otro hacia arriba. Para ello se han calculado dos configuraciones teniendo en cuenta que no colisione con el propio robot y seguidamente el punto de golpeo.

La primera configuración baja el brazo a la posición más próxima al cuerpo para ello se posiciona el servo del brazo a 120º y el puño a 130º en el caso del gancho de izquierda y el brazo a 85º y el puño a 50º en el caso del gancho derecha.

La segunda configuración busca asestar el golpe con lo que el brazo subirá a su posición máxima y el puño se recoge para dar el golpe. Además, se mueve la cadera de la misma manera que se mueve en el jap con tal de poder ejecutar un golpe con más fuerza. Para ello se configuran lo siguientes ángulos en los servomotores para el gancho de derechas:

- Pies: 90º
- Piernas: 105º
- Brazo derecho: 40º
- Brazo izquierdo: 60º
- Puño derecho:  160º
- Puño izquierdo: 180

En el caso del gancho de izquierdas:

- Pies: 90º
- Piernas: 75º
- Brazo derecho: 10º
- Brazo izquierdo: 30º
- Puños: 150º

A continuación, se muestra un ejemplo de funcionamiento:

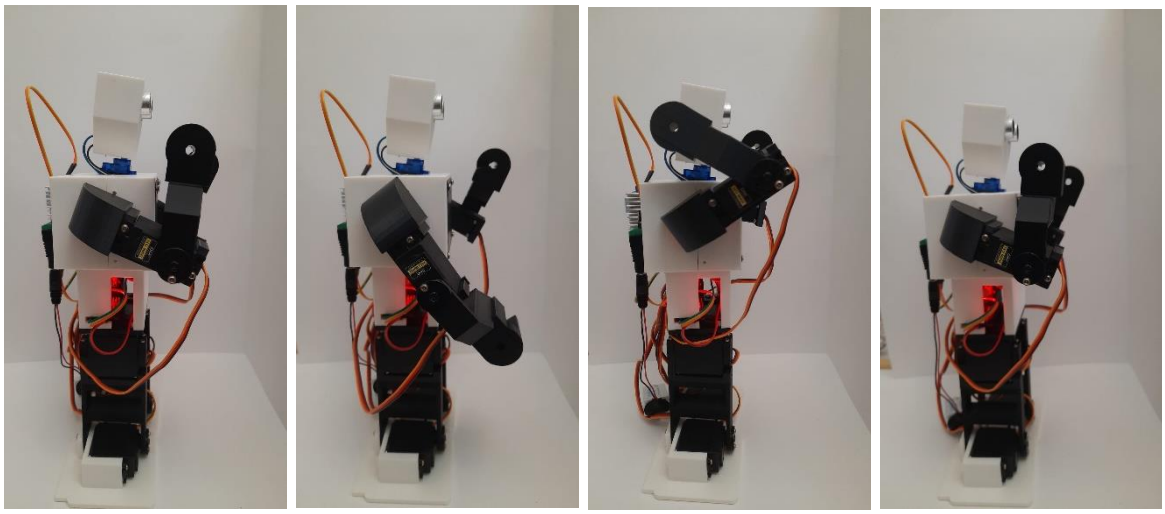


**Figura 67:** Imitación de gancho de derechas

## 7.6. Bailes

En el cálculo de las configuraciones de los bailes son meramente estéticas y el único criterio aplicado ha sido evitar colisiones entre componentes y mantener la estabilidad. Se han creado dos bailes uno de derrota y otro de victoria

### 7.6.2. Baile de victoria

Este baile pretende imitar un baile de victoria (figura 68) levantando los brazos y a si mismo con los pies. Para ello se utiliza una sola configuración nueva además de la configuración base. La nueva configuración consiste en los siguientes ángulos:

- Pie izquierdo: 130º
- Pie derecho: 50º
- Piernas: 90º
- Brazo izquierdo: 0º
- Brazo derecho: 180º
- Puño izquierdo: 150º
- Puño derecho: 40º

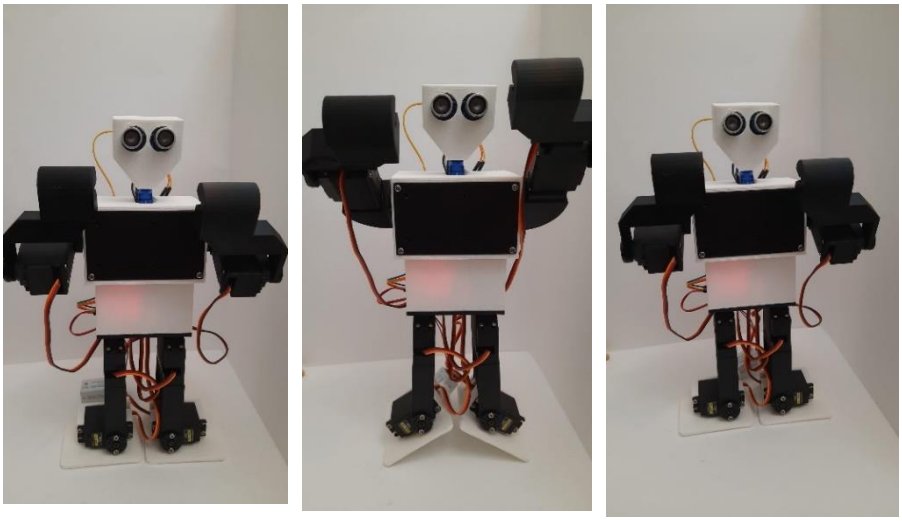


**Figura 68:** Baile de victoria

### 7.6.3. Baile de derrota

El baile de derrota (figura 69) consiste en un balanceo del robot subiendo y bajando los brazos. Para ello se han definido dos configuraciones que varían entre una y otra. En estas configuraciones se ha tenido en cuenta no perder la estabilidad en el balanceo además de que los componentes no colisionen. La primera configuración inclina el robot hacia la derecha y baja los brazos, para ello se configuran los servos en los siguientes ángulos:

- Pies: 110º
- Piernas: 90º
- Brazo izquierdo: 120º
- Brazo derecho: 65º
- Puños 90º

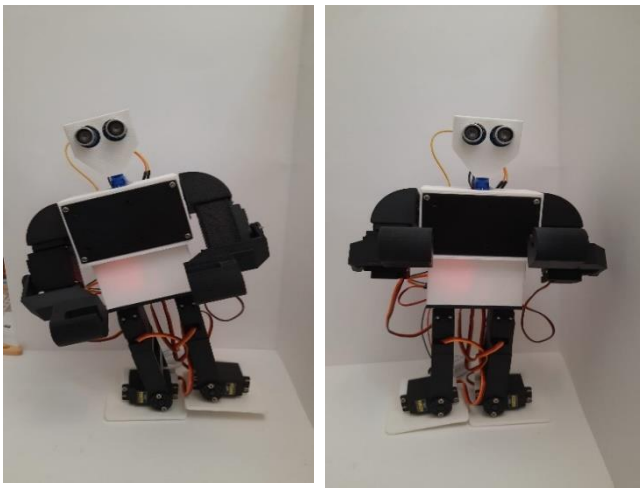


**Figura 69:** Baile de derrota

# 8. Resultados

Una vez realizado el montaje y la programación se procede a la comprobación de los resultados para ello se comprobarán diversos movimientos que puede realizar el robot además de la comprobación del funcionamiento de las aplicaciones autónomas.

Para la demostración de los movimientos se realizará un paso hacia adelante y un tipo de golpe. Uno de los movimientos se muestra en la figura 63 donde se muestra la correcta ejecución de la secuencia de movimientos calculados con anterioridad. Se puede observar que el robot avanza correctamente sin que haya colisiones entre las articulaciones ni se pierda estabilidad. Por otra parte, en la figura 67 se muestra una secuencia en la que el robot realiza el movimiento de gancho de derechas. Como se puede observar se realiza la secuencia apropiada sin ninguna colisión con ninguna de las piezas del robot. Cabe destacar el hecho de la dificultad de elevar la pierna derecha debido a la diferencia de fuerza entre el servo derecho y servo izquierdo. Sin embargo, se han calculado los ángulos de tal manera que el movimiento se ejecute efectivamente de igual manera.

A continuación, se comprueba el funcionamiento de las aplicaciones autónomas. El correcto funcionamiento de la primera aplicación se demuestra en la figura 70 donde se puede observar que avanza correctamente hasta detectar el objeto. Seguidamente se observa una de las series de golpes, y por último se observan tanto el baile de derrota como el de victoria.

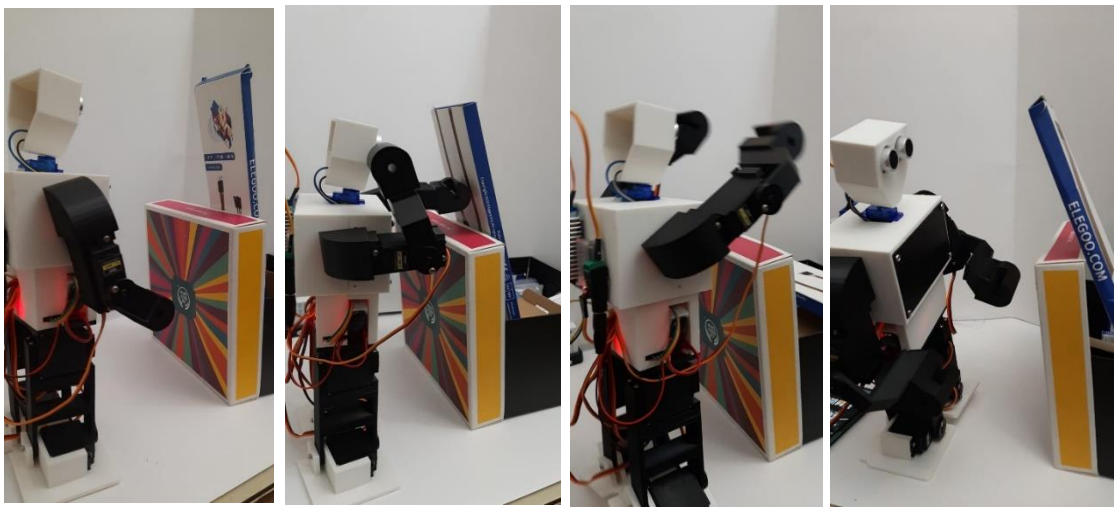



**Figura 70:** Ejecución de la aplicación automática 1

Por último, se comprueba el funcionamiento de la segunda aplicación autónoma. En la secuencia mostrada en la figura 71 se puede observar el correcto funcionamiento de la aplicación. En primer lugar, se observa al robot en la posición de guardia correcta, a continuación, se observa como el robot retrocede al detectar un objeto. Finalmente se puede observar tanto el baile de derrota en el caso de que el objeto permanezca ahí y un baile de victoria en el caso de que el objeto ya no sea detectado.

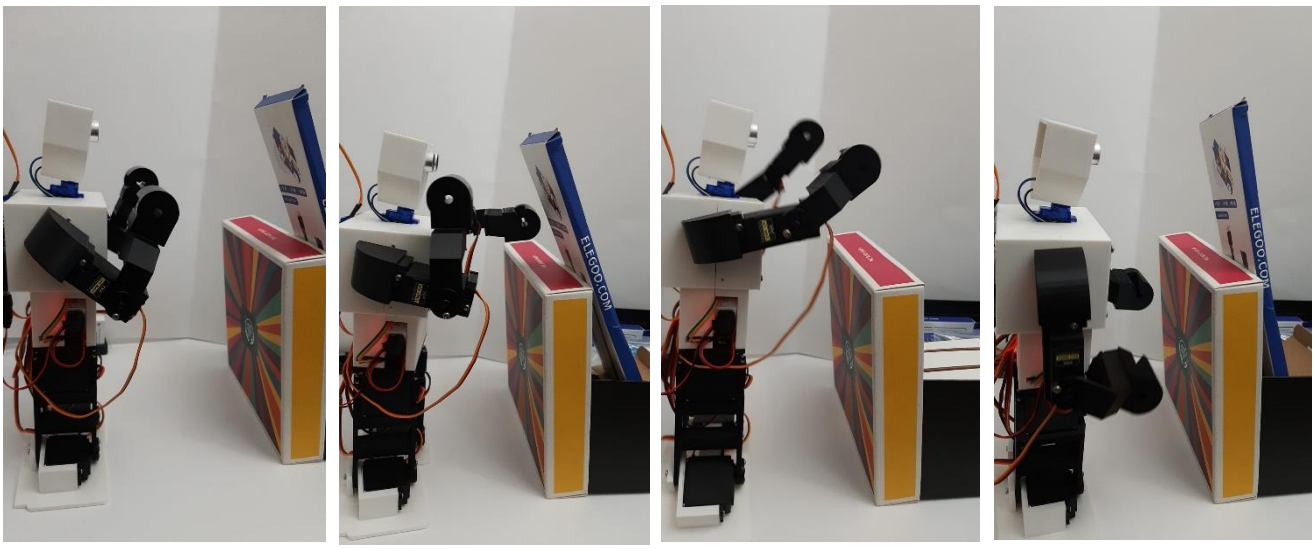


**Figura 71:** Comprobación del funcionamiento de la aplicación autónoma 2

# 9. Conclusiones

En este apartado se analiza el trabajo realizado durante el proyecto y se evaluarán los resultados obtenidos mostrados en el apartado anterior. Por otra parte, ser realizará una valoración personal sobre los conocimientos adquiridos y las problemáticas encontradas.

En lo referente al cumplimiento de los objetivos, dados los resultados mostrados se puede concluir que se han cumplido de manera óptima la mayoría de los objetivos. Por lo que a impresión y diseño 3D se refiere se ha conseguido diseñar e imprimir con éxito la totalidad de la estructura del robot, sin embargo, debido a la falta de conocimientos iniciales esta fase del proyecto ha ocupado más tiempo del esperado. Por otra parte, el diseño puede ser mejorado en cuanto a estética y rendimiento. Pese a ello se ha conseguido una estructura estable y funcional cumpliendo así el objetivo.

En lo que a programación se refiere se puede concluir que se han cumplido la totalidad de los objetivos. En primer lugar, se ha programado con éxito la forma de movimiento requerida para esta tipología de robot. Esto se puede concluir dado que el robot es capaz de moverse en ocho direcciones distintas con una precisión aceptable teniendo en cuenta el diseño y la electrónica. Además, el robot es capaz de mantener la estabilidad en todas sus configuraciones y sus movimientos no provocan colisiones entre sus componentes y extremidades.

Por otro lado, se han programado con éxito las tres aplicaciones propuestas en el proyecto. Esto se observa en el apartado de resultados del proyecto debido a que el comportamiento en las dos aplicaciones autónomas es el esperado y el control remoto funciona adecuadamente. Sin embargo, existe margen de mejora debido al sensor utilizado. Esto se debe a que el sensor de ultrasonidos escogido tiene una muy baja precisión y estabilidad provocando mediciones erróneas que afectan al comportamiento del robot. Pese a ello el comportamiento es el esperado y los resultados satisfactorios.

En lo referente a la programación de la aplicación se concluye que se han cumplido los objetivos relacionados con esta. Esto se debe al correcto funcionamiento de la aplicación y a la correcta coordinación con el robot que provoca el comportamiento deseado. Sin embargo, la aplicación

también contiene margen de mejora en lo referente a diseño y optimización puesto que no se tenían conocimientos previos del programa utilizado.

Durante la programación del proyecto se han encontrado diversas problemáticas que han tenido que ser solucionadas. La primera problemática encontrada está relacionada con la memoria ocupada por los programas realizados. Dada la cantidad de funciones y variables implicadas en todos los programas estos utilizan mucho espacio y se ha tenido que optimizar todos ellos con tal de ocupar menor espacio de memoria. Por otra parte, debido a la inexperiencia en el uso del programa empleado para la creación de la aplicación, la programación ha requerido más tiempo del esperado.

El diseño e implementación de la electrónica también se ha implementado con éxito debido al correcto funcionamiento y eficiencia del robot. Pese a ello se encontraron diversas problemáticas que han provocado que la electrónica de potencia ocupe más espacio del inicialmente planteado. La principal problemática es la potencia requerida por los servomotores dado que al comienzo del proyecto se dimensiono erróneamente debido a la falta de acceso a material de laboratorio. Esto derivo en una adquisición de un convertidor de mayor tamaño provocando el sobredimensionamiento descrito.

Por último, se puede concluir el correcto diseño e implementación del robot planteado dados los resultados. Esto se debe al correcto funcionamiento de este y a la solución de todas las problemáticas surgidas. Tanto el movimiento como la imitación de los golpes se realiza de manera satisfactoria cumpliendo así el objetivo planteado. Por otra parte, el robot no tiene colisiones entres sus articulaciones y mantiene la estabilidad en todo momento.

En lo referente a los conocimientos adquiridos se ha cumplido con la expectativa. Esto se debe a que se ha conseguido manejar correctamente tanto software como hardware de impresión y diseño 3D, conocimientos que al comienzo del proyecto no se poseían. Por otro lado, se han aplicado de manera práctica muchos de los conocimientos y métodos adquiridos en el grado cursado, como por ejemplo la programación y la resolución de problemas. Además, se ha utilizado tanto electrónica como software no utilizado anteriormente, en específico, el programa Thunkable X y servomotores de una gama con estas características.

En conclusión, se puede afirmar que todos los objetivos planteados al inicio del proyecto han sido cumplidos satisfactoriamente. Por otra parte, todas las problemáticas encontradas durante la realización del proyecto han sido solventadas con éxito resultando en un óptimo funcionamiento del robot. Además, se han adquirido y puesto en práctica diversos conocimientos en varias áreas del conocimiento relacionadas con la robótica

# 10. Referencias


Dougherty, D. (2012). The Maker Movement. *Innovations: Technology, Governance, Globalization*, *7*(3), 11–14. https://doi.org/10.1162/inov_a_00135

Guizzo, E. (2019). By leaps and bounds: An exclusive look at how Boston dynamics is redefining robot agility. *IEEE Spectrum*, *56*(12), 34–39. https://doi.org/10.1109/MSPEC.2019.8913831

Rosa, P., Ferretti, F., Guimarães Pereira, Â., Panella, F., & Wanner, M. (2017). *Overview of the Maker Movement in the European Union*. https://doi.org/10.2760/227356

Ruiz-de-garibay, J., & Estado, R. (n.d.). *Robótica : Estado del arte*.

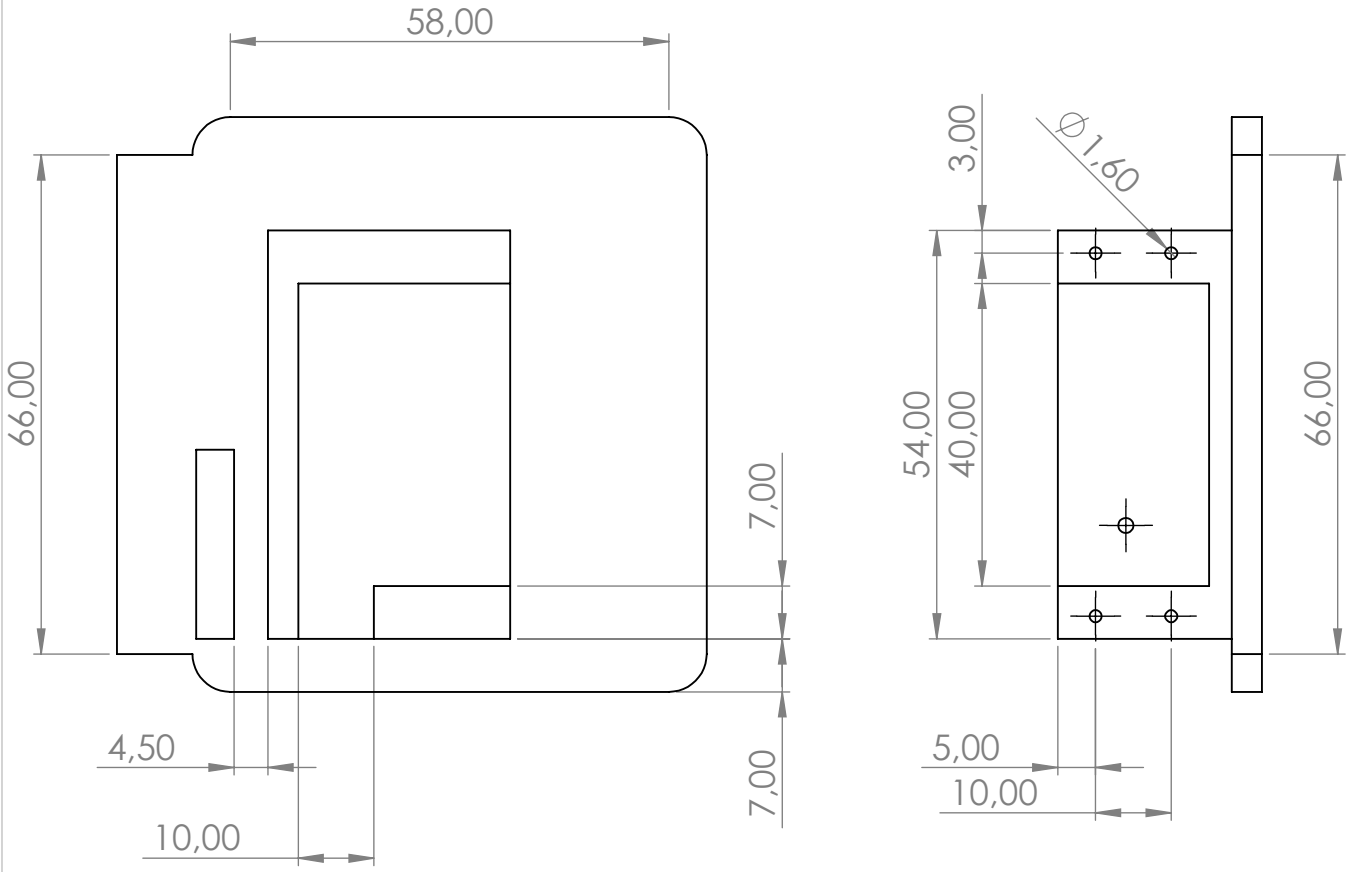
Savini, A., & Savini, G. G. (2015). A short history of 3D printing, a technological revolution just started. *Proceedings of the 2015 ICOHTEC/IEEE International History of High-Technologies and Their Socio-Cultural Contexts Conference, HISTELCON 2015: The 4th IEEE Region 8 Conference on the History of Electrotechnologies*. https://doi.org/10.1109/HISTELCON.2015.7307314

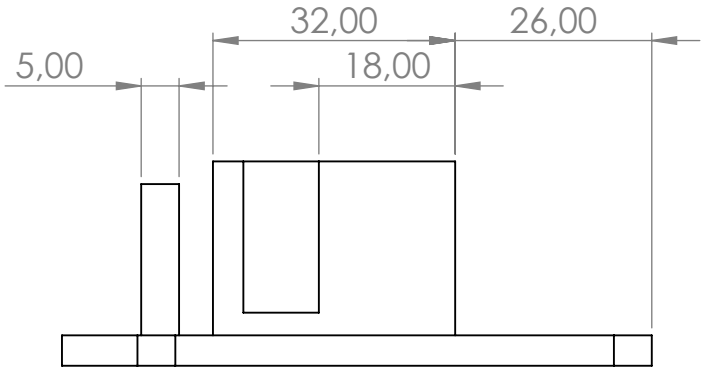
Sotelo, V. R. B., Sánchez, J. R. G., & Ortigoza, R. S. (2007). Robots Móviles: Evolución y Estado del Arte. *Polibits*, *35*, 12–17.

Villagomez, M. L. Q., Caracheo, L. A. A., Mondragón, G. R., & Castro, R. R. (2017). Revisión Del Estado Del Arte De La Fabricación De Multimateriales Por Medio De Impresión 3D. *Pistas Educativas*, *39*(125), 441–451. http://www.itcelaya.edu.mx/ojs/index.php/pistas/article/view/885

Zavaleta, L. (2019). *Memoria de Prácticas Robótica e Impresión 3D aplicadas al contexto educativo*. http://diposit.ub.edu/dspace/bitstream/2445/155177/1/TFM_Zavaleta_Paola_2019.pdf

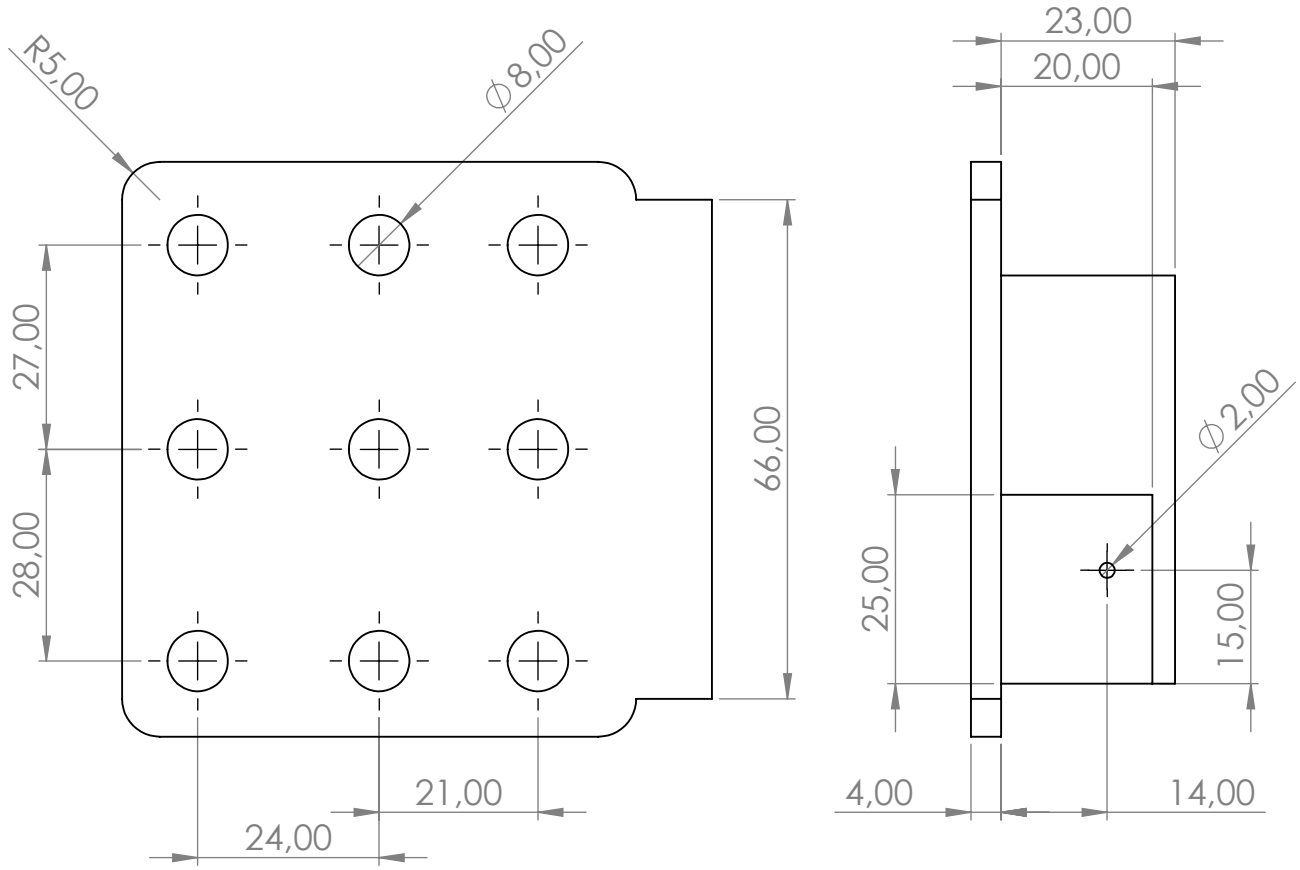
# DOCUMENTO 2: PLANOS



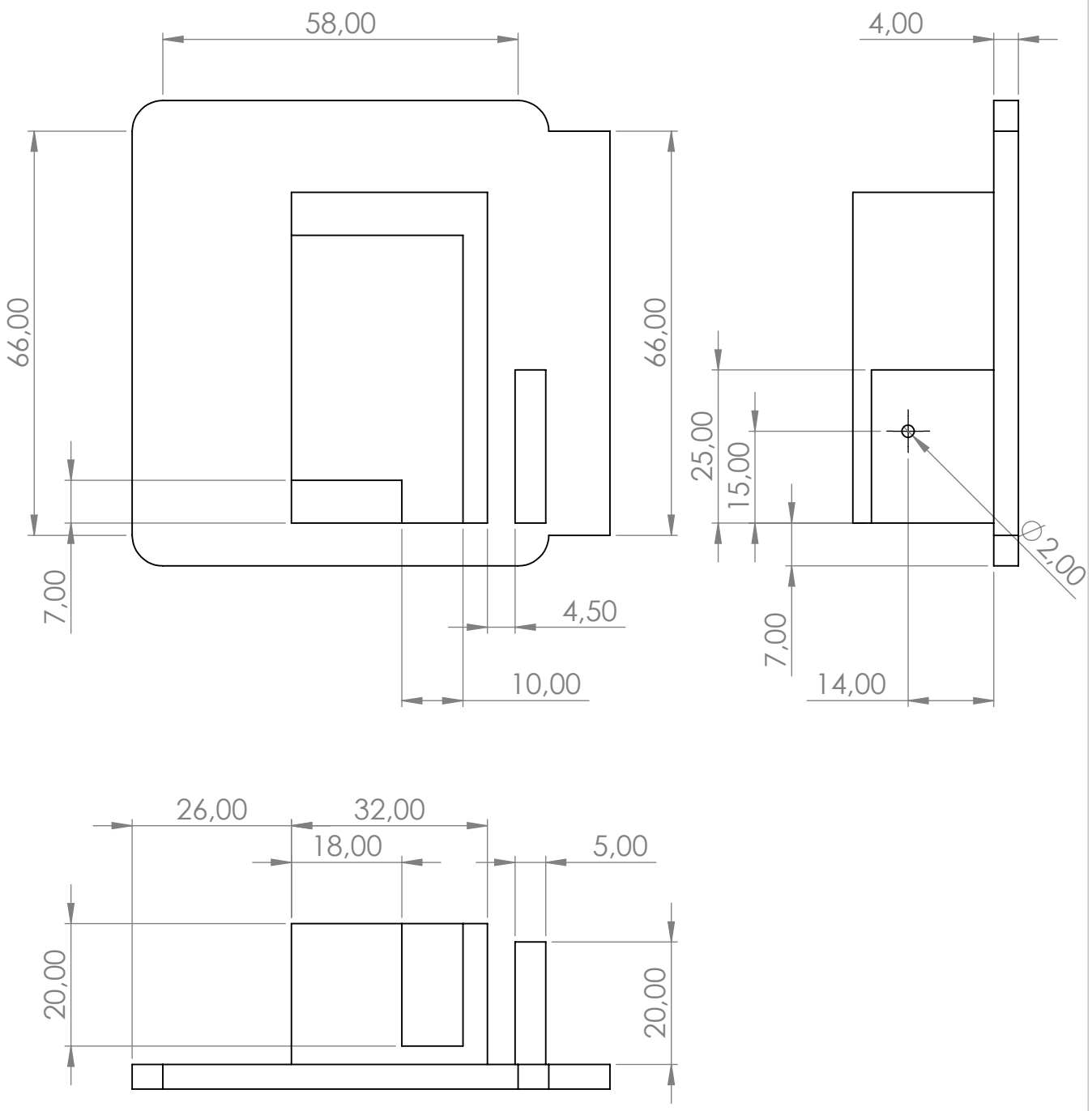

| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | PIE IZQUIERDO | 01 |



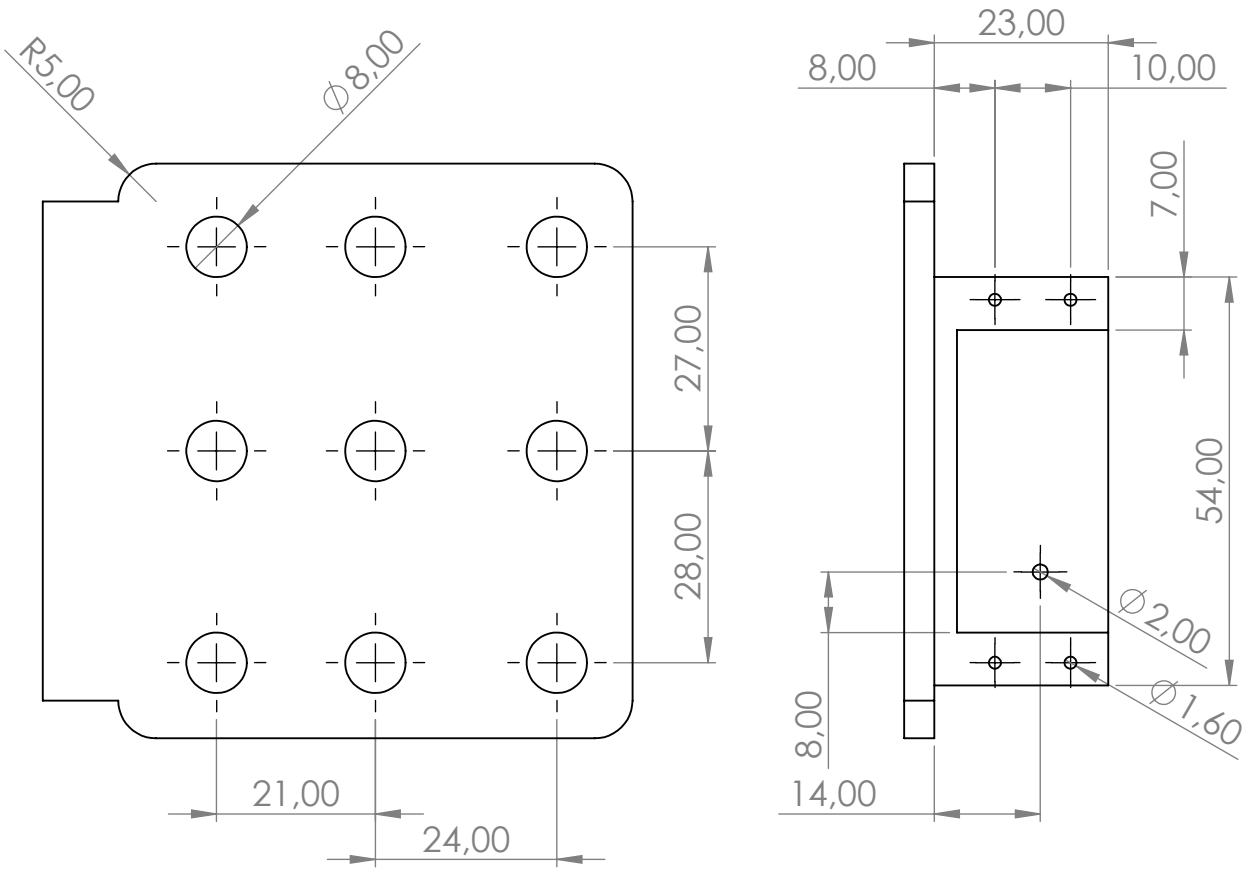

| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | PIE IZQUIERDO 2 | 02 |



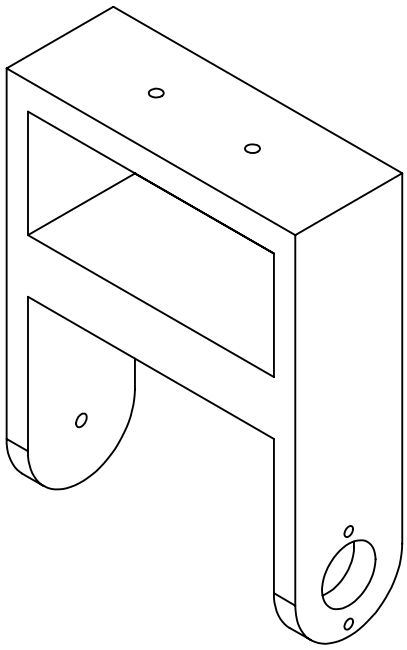

| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

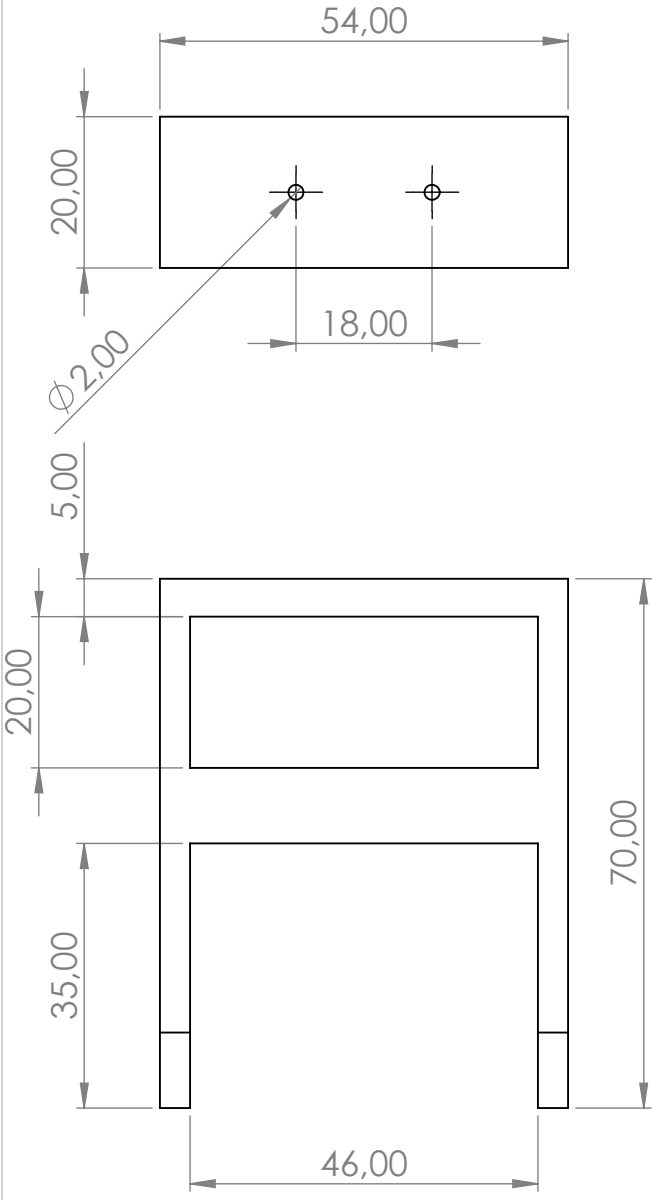
| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | PIE DERECHO 1 | 03 |

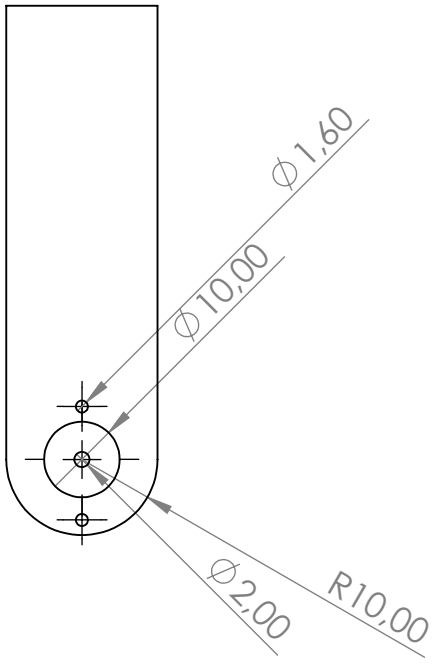



| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | PIE DERECHO 2 | 04 |

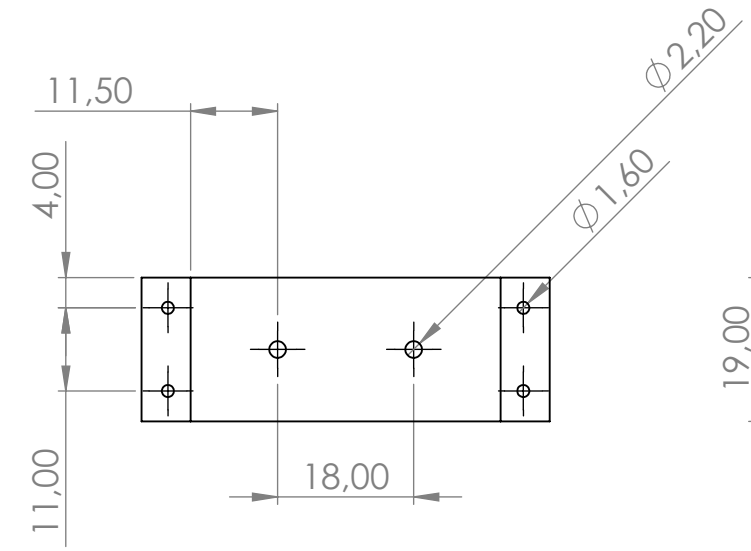


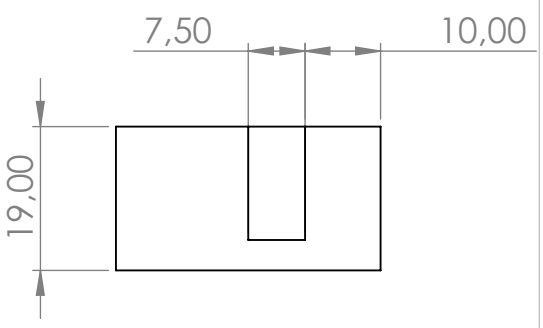


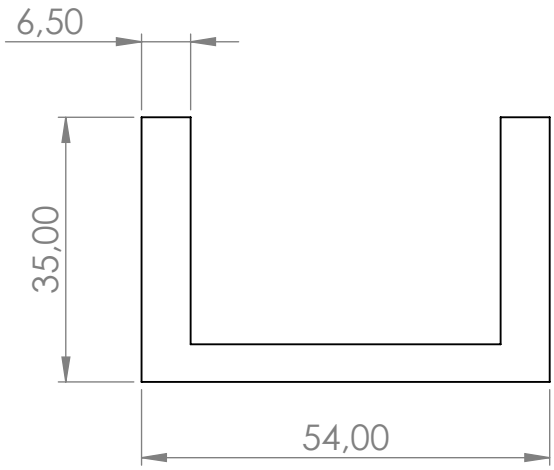



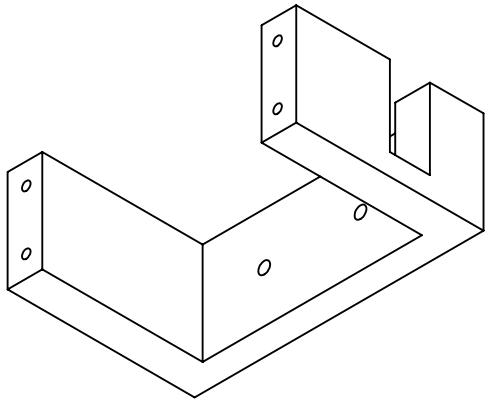

| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | PIERNA | 05 |



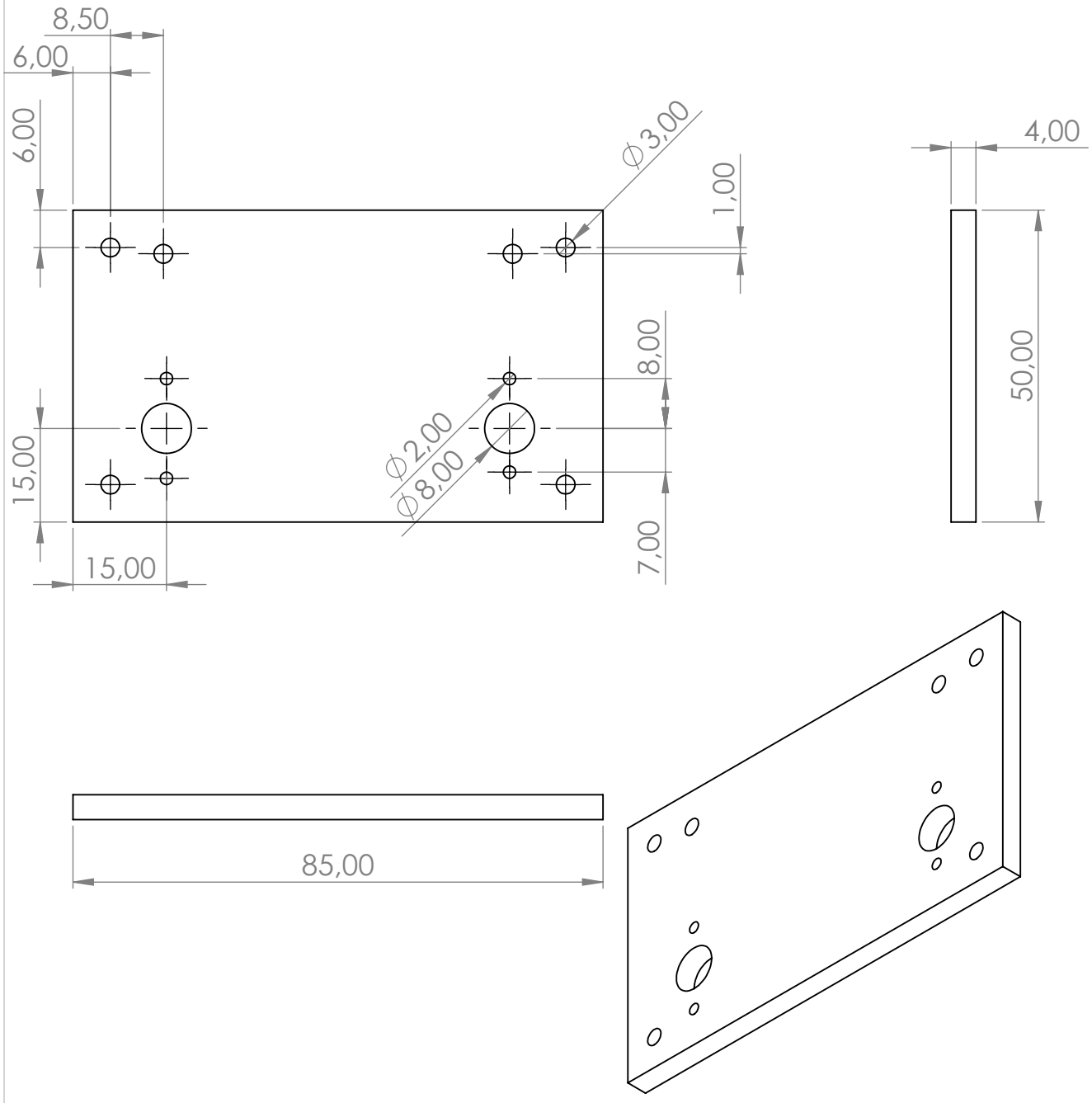



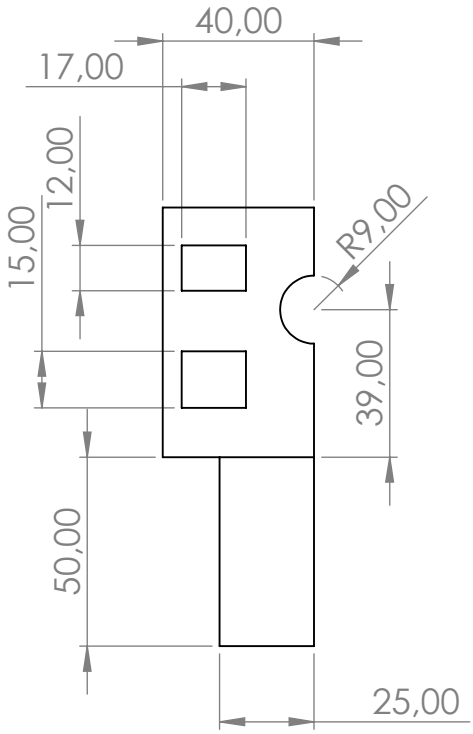
| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

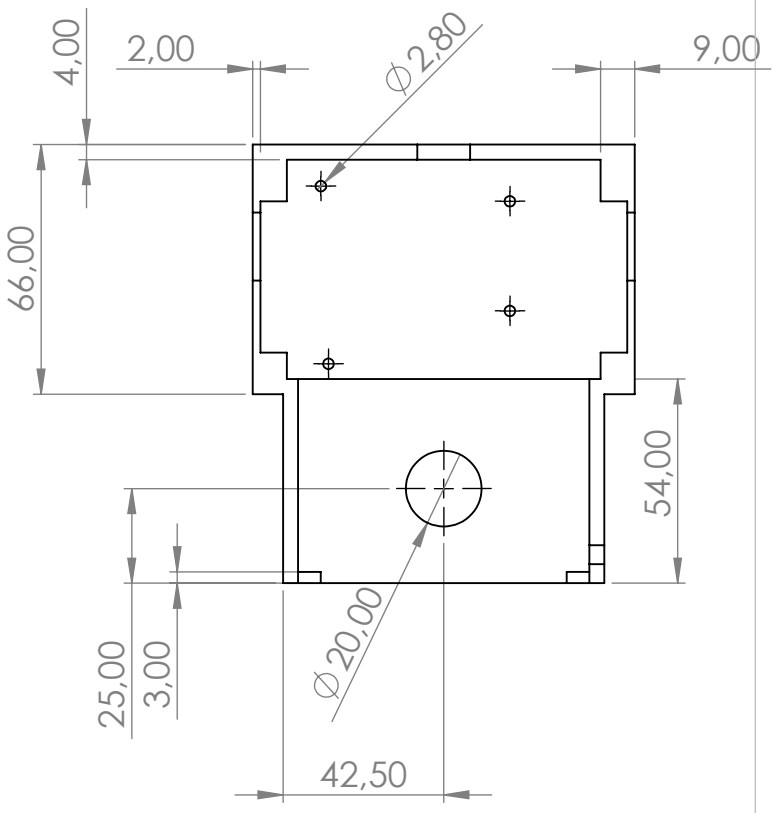
| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | SOPORTE CADERA | 06 |

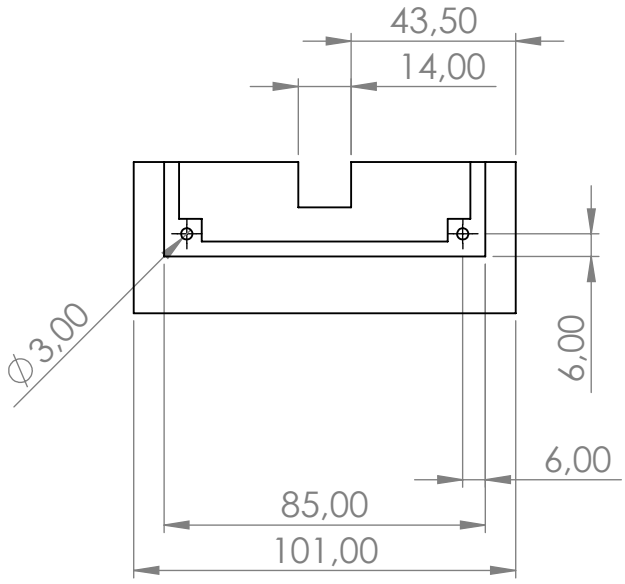



| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | CADERA | 07 |

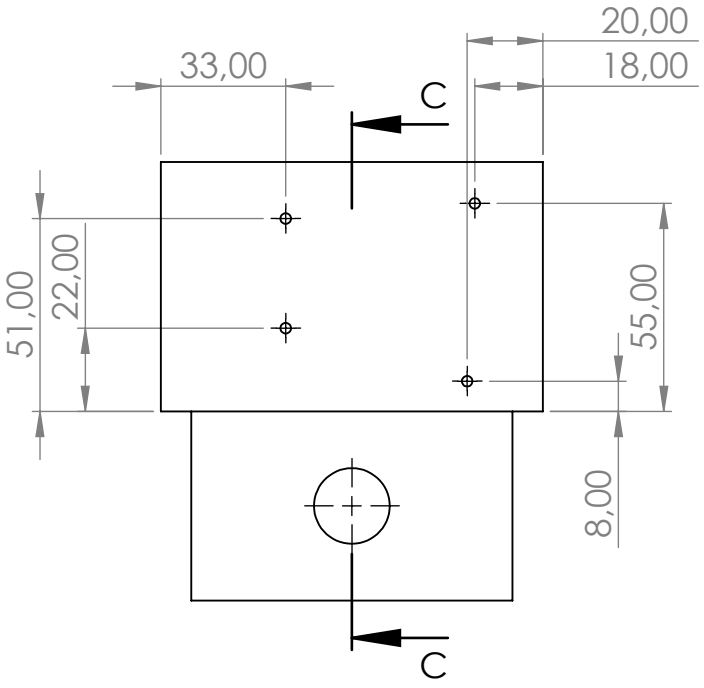


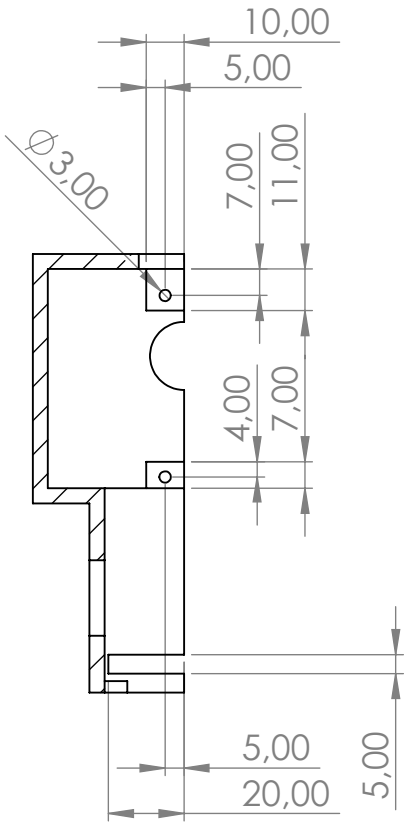



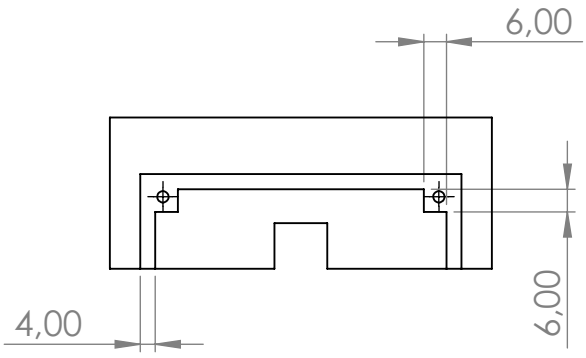




| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:2 | Espalda 1 | 08 |

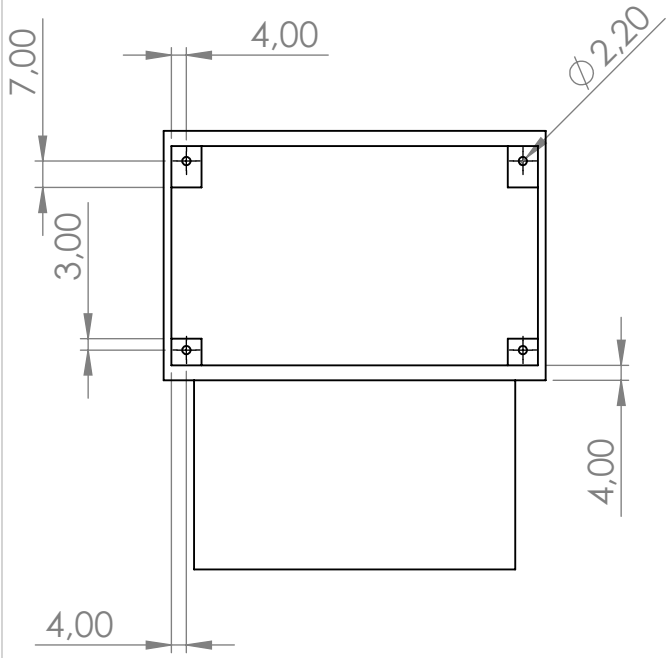


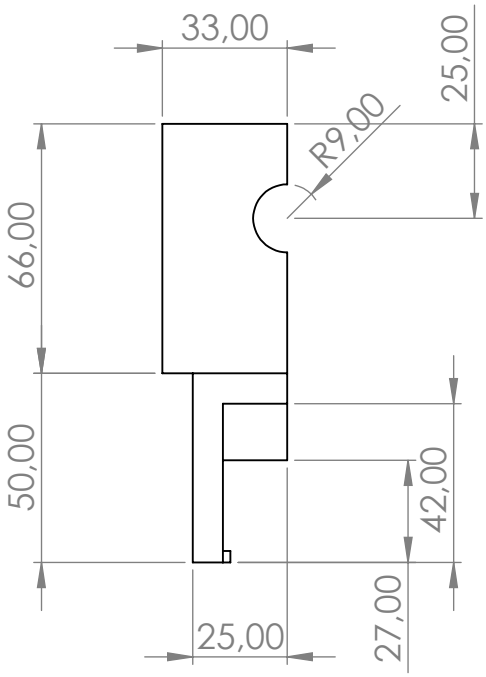



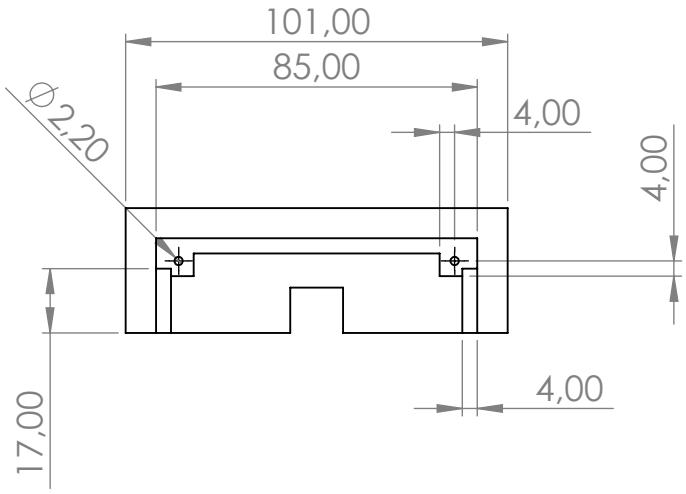




| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:2 | ESPALDA 2 | 09 |

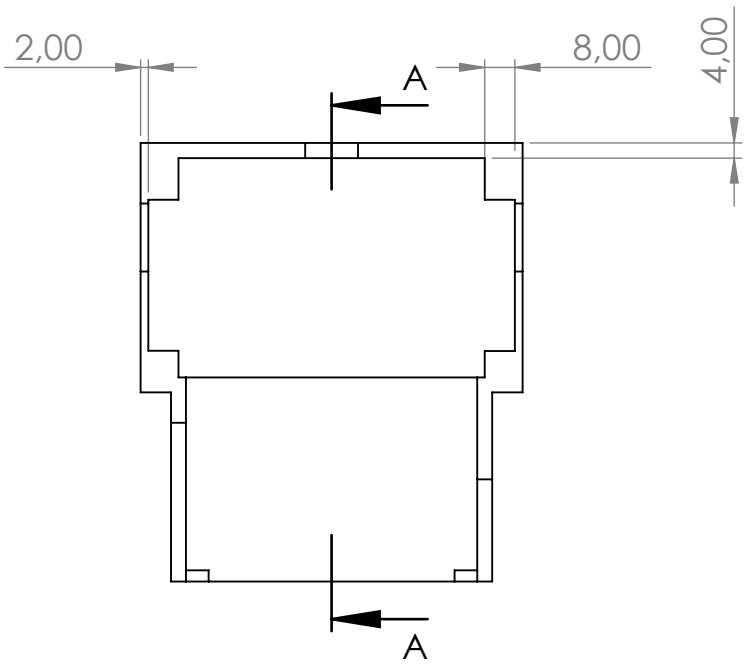


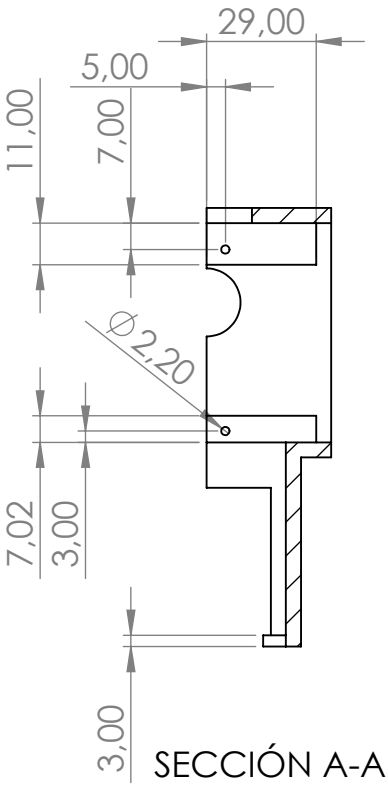



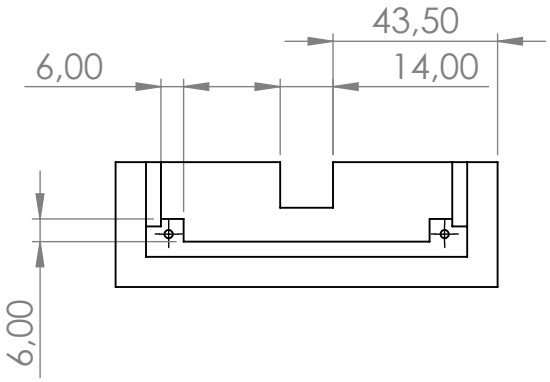

| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:2 | TORSO | 10 |



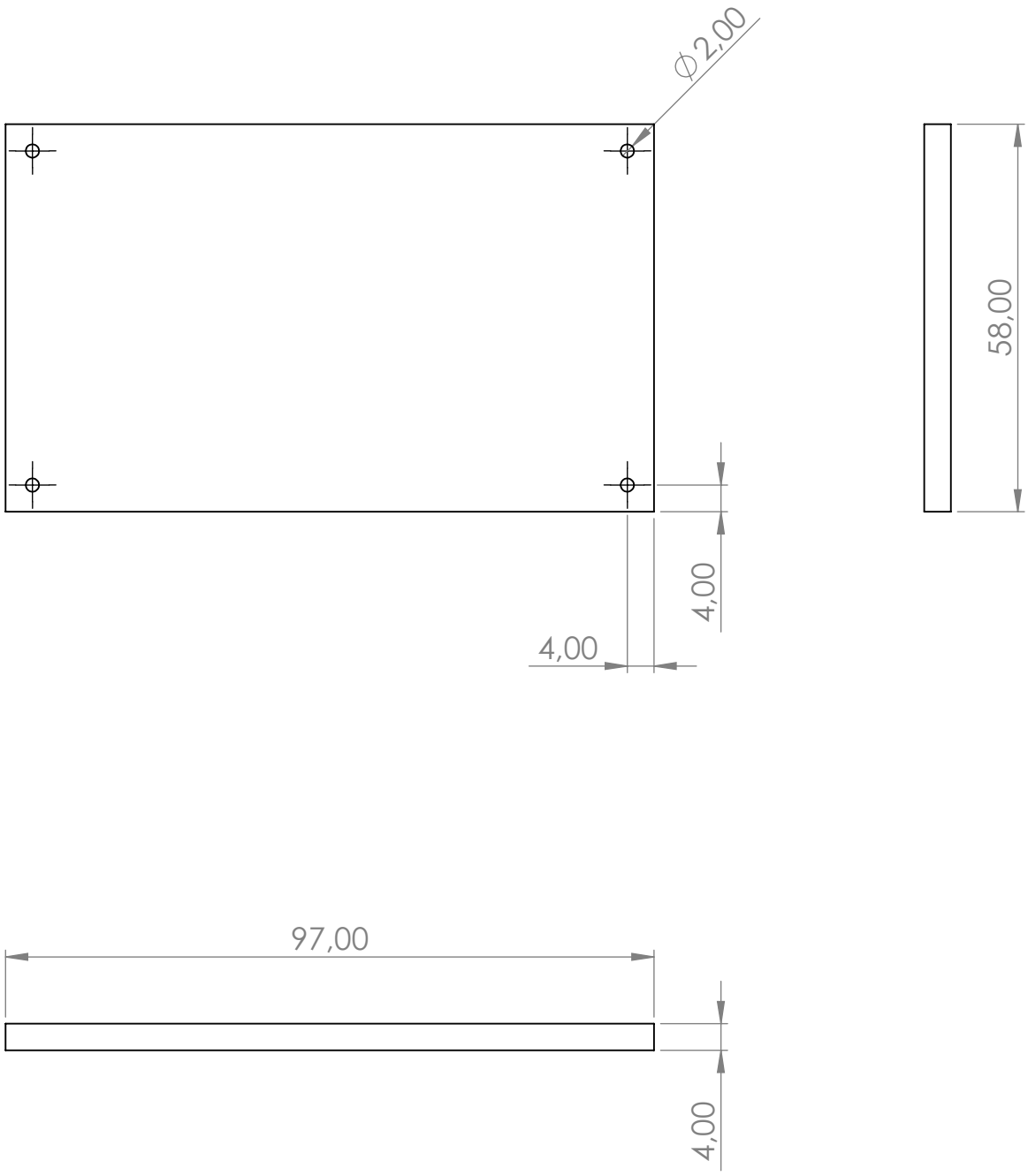




SECCIÓN A-A



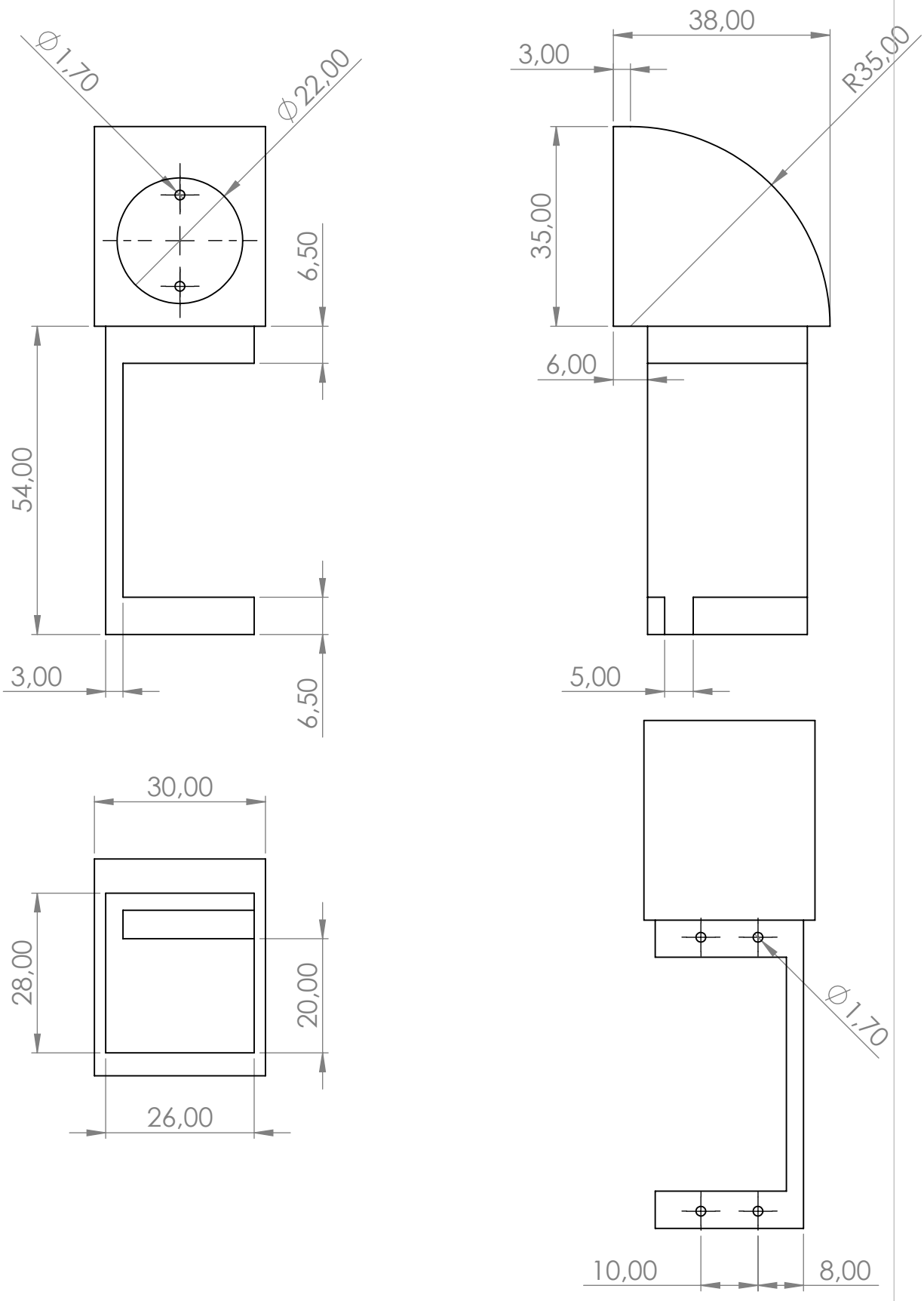

| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:2 | TORSO 2 | 11 |



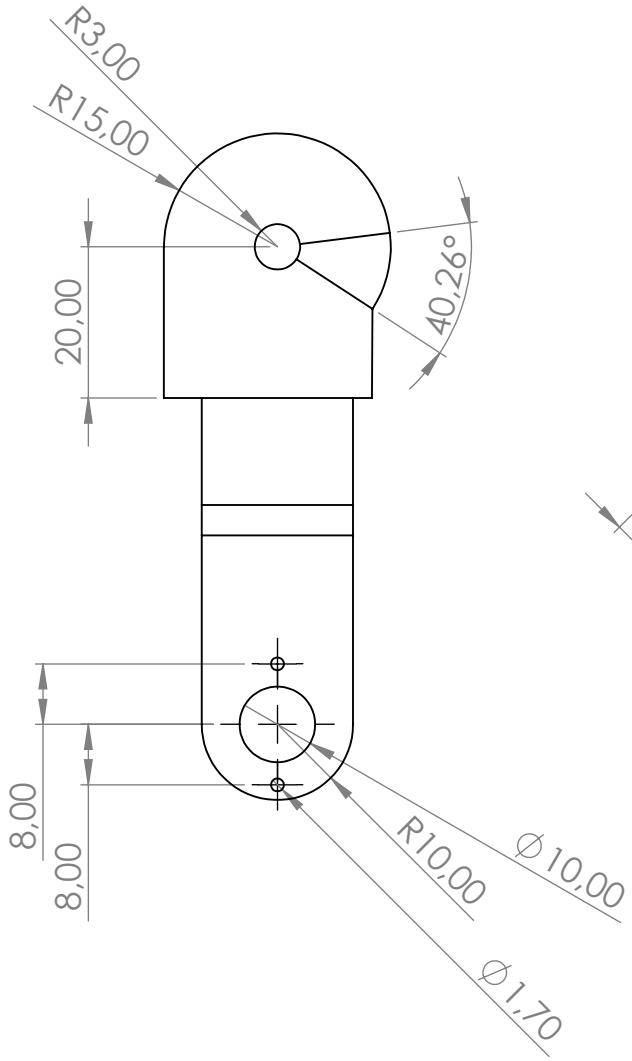

| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

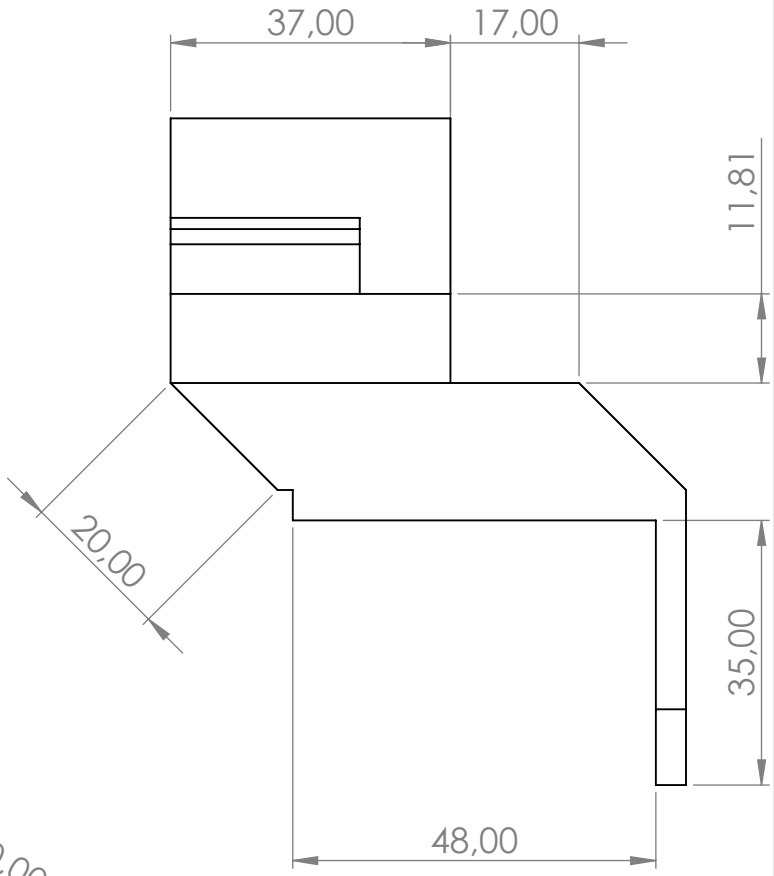
| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | CUBIERTA TORSO | 12 |

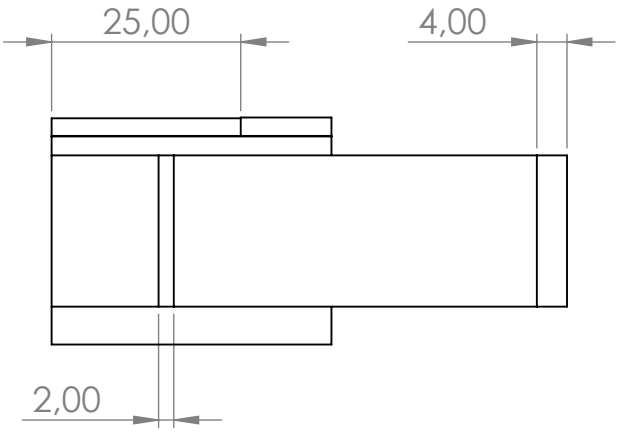



| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | BRAZO | 13 |

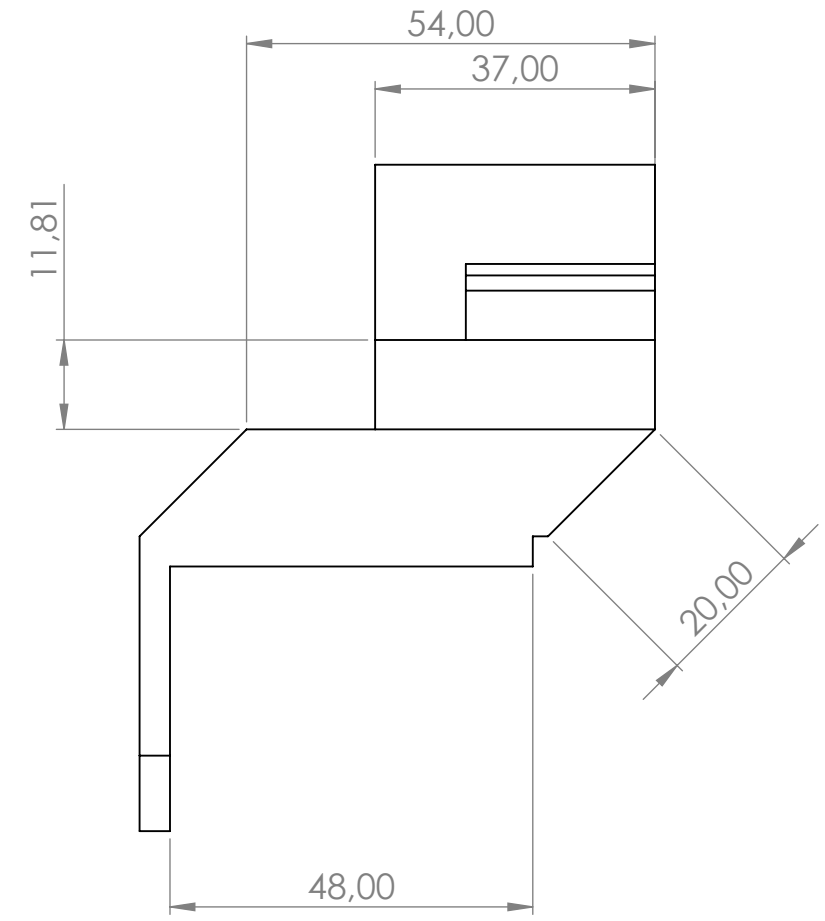


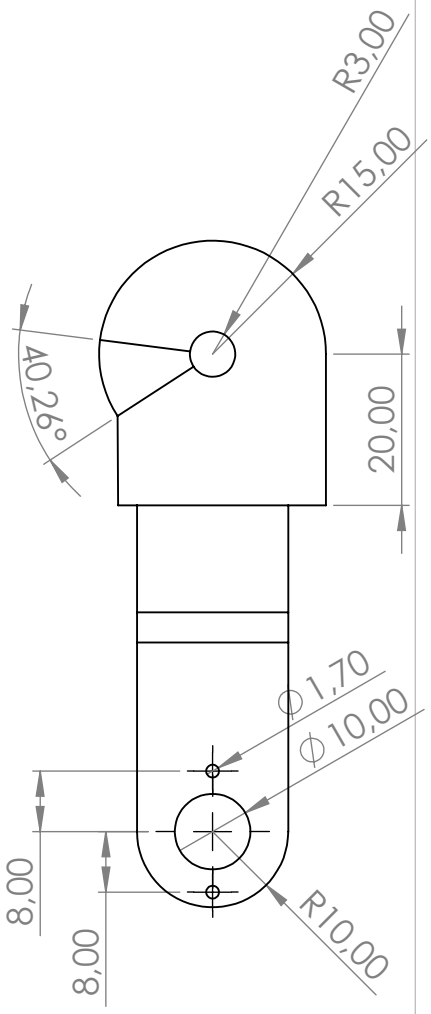



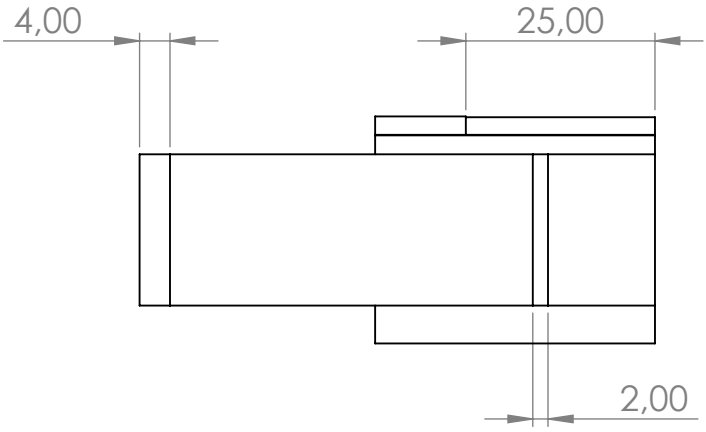




| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | PUÑO DERECHO | 14 |

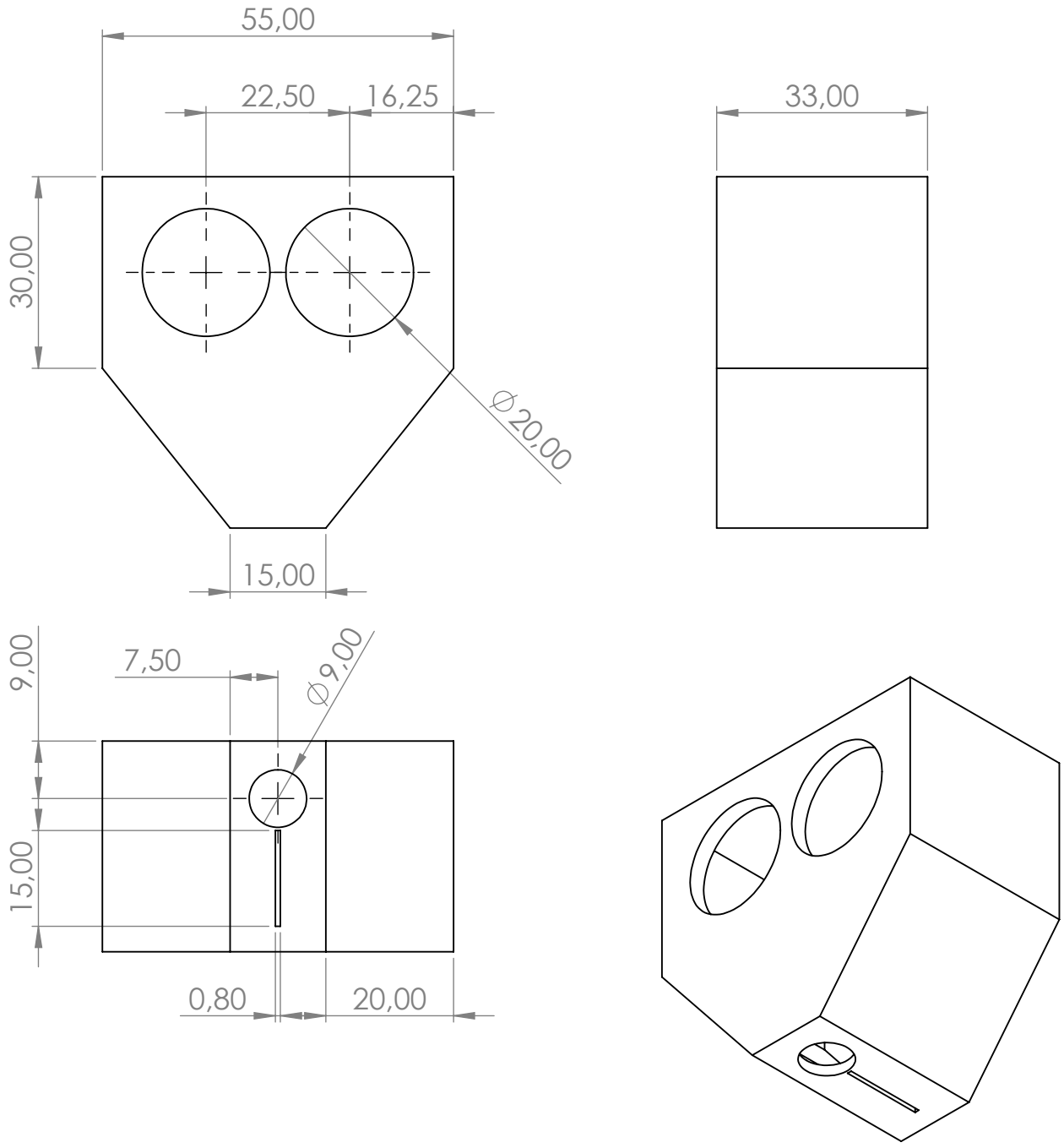



| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | PUÑO IZQUIERDO | 15 |




| | Fecha | Nombre | Firma | ESCUELA TÉCNICA SUPERIOR DE LA INGENIERIA DEL DISEÑO<br>TRABAJO FIN DE GRADO<br>Título: Robot boxeador multifunción<br>Autor: Adrià Bosch Serra |
|---|---|---|---|---|
| Dibujado | 15/06/21 | A.Bosch | | |
| Comformado | 15/06/21 | A.Bosch | | |

| Escala: | | Nº de plano: |
|---|---|---|
| 1:1 | CABEZA | 16 |

# DOCUMENTO 3: PRESUPUESTO

| Materiales | | | | | |
|---|---|---|---|---|---|
| **Alimentación** | | | | | |
| Componente | Descripción | Unidad | Cantidad | Precio Und | Precio total |
| Convertidor DC-DC | Convertidor DC-DC con salida ajustable. | Ud | 1 | 8,99 € | 8,99 € |
| Adaptador Jackx10 | Pack de adaptadores Jack | Ud | 1 | 9,99 € | 9,99 € |
| Bateria | Bateria Lipo 9V 600mAh | Ud | 4 | 5,00 € | 20,00 € |
| Cargador | Base y cargador de baterias tipo Lipo | Ud | 1 | 20,00 € | 20,00 € |
| cable adaptador | Cable adaptador de la bateria a salida jack | Ud | 1 | 3,20 € | 3,20 € |
| | | | | Total sin IVA | 49,12 € |
| | | | | Total | 62,18 € |
| **Electrónica** | | | | | |
| Módulo Bluetooth | Módulo bluetooth HM-10 | Ud | 1 | 9,99 € | 9,99 € |
| PCA9685 | Servo Shield 16 pines | Ud | 1 | 6,99 € | 6,99 € |
| Cableado | Pack cableado macho-macho, macho-hembra,hembra-hembra | Ud | 1 | 6,99 € | 6,99 € |
| ELEGOO UNO R3 | Microcontrolador | Ud | 1 | 9,99 € | 9,99 € |
| | | | | Total sin IVA | 26,83 € |
| | | | | Total | 33,96 € |
| **Sensores y actuadores** | | | | | |
| Servo motor MG996R | Servomotor de engranaje metálico con 13 Kg-cm | Ud | 12 | 5,00 € | 60,00 € |
| Servomotor SG90R | Servomotor | Ud | 1 | 5,00 € | 5,00 € |
| HC-SR04 | Sensor de ultrasonidos | Ud | 1 | 2,00 € | 2,00 € |
| | | | | Total sin IVA | 52,93 € |
| | | | | Total | 67,00 € |
| **Estructura** | | | | | |
| Tornilleria | Pack de tornilleria M3 de 5 a 20 mm | Ud | 1 | 12,99 € | 12,99 € |
| Plástico PLA Impresión 3D | Rollo de plástico PLA para impresión 3D | Ud | 1 | 24,99 € | 24,99 € |
| | | | | Total sin IVA | 30,00 € |
| | | | | Total | 37,98 € |
| | | | | Total materiales: | 201,12 € |

| Mano de obra | | | | | |
|---|---|---|---|---|---|
| Diseño | Diseño de la estructura y electrónica del robot | h | 20 | 13,33 € | 266,60 € |
| Montaje | Supervision de la impresión de las piezas y montaje del robot | h | 20 | 13,33 € | 266,60 € |
| Programación | Programación de las diversas aplicaciones del robot | h | 15 | 13,33 € | 199,95 € |
| Testeo | Comprobación del correcto funcionamiento del robot | h | 5 | 13,33 € | 66,65 € |
| Escritura | Escritura de la memoria técnica del proyecto | h | 20 | 13,33 € | 266,60 € |
| Revisión | Revision de la memoria técnica del proyecto | h | 15 | 40,00 € | 600,00 € |
| | | | | Total materiales: | 1.666,40 € |

| Maquinaria y equipos | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Componente | Descripción | Und | cantidad | Consumo (W) | Energia | Amortización | Mantenimiento | Coste |
| Computadora | Utilización de computadoras en la etapa de diseño y programación | h | 50 | 250 | 3,13 € | 0,02 € | 0,00 € | 3,15 € |
| Impresora 3D | Utilización de la impresora 3D | h | 150 | 150 | 5,63 € | 0,01 € | 2,00 € | 7,63 € |
| | | | | | | | Total Maquinaria y equipos: | 10,78 € |

| Medios Auxiliares | | | | |
|---|---|---|---|---|
| | | | | |
| Medios Auxiliares | % | 1.878,30 € | 10% | 187,83 € |
| | | | | |
| | | | Total Proyecto: | 2.066,13 € |

# ANEXO I: CÓDIGO

## 1.Funciones básicas

```
void Back_Right()
/*
 * Entrada: Ninguna
 * Salida: Ninguna
 *
 * Ejecuta un paso en direccón atras derecha
 */
{
  //Establezco el vector de la configuración deseada
  Change_vector(qn, 80, 80, 100, 95, 120, 65, 90, 90);
  //Movimiento del robot
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  //Se establece la posición actual en un vector para el siguiente
movimiento
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 80, 90, 65, 75, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 65, 75, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 110, 105, 65, 75, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 105, 100, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
```

```
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 100, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
}
void Back_Left()
/*
 * Entrada: Ninguna
 * Salida: Ninguna
 *
 * Ejecuta un paso en dirección atrás Izquierda
 */
{
  Change_vector(qn, 80, 80, 95, 100, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 80, 90, 75, 65, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 75, 65, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 110, 105, 75, 165, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
```

```
  Change_vector(qn, 90, 105, 95, 100, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 95, 100, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
}
void Forward_Right()
{
 /*
 * Entrada: Ninguna
 * Salida: Ninguna
 *
 * Ejecuta un paso en dirección adelante derecha
 */
  Change_vector(qn, 80, 80, 65, 75, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 80, 90, 100, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 100, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 110, 105, 100, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
```

```
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 105, 65, 75, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 65, 75, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
}
void Forward_Left()
{
 /*
 * Entrada: Ninguna
 * Salida: Ninguna
 *
 * Ejecuta un paso en dirección adelante izquierda
 */
  Change_vector(qn, 80, 80, 80, 65, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 80, 90, 95, 100, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 95, 100, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
```

```
    q0[i] = qn[i];
  }
  Change_vector(qn, 110, 105, 95, 100, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 105, 80, 65, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 80, 65, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
}
void Win_Dance()
{
 /*
 * Entrada: Ninguna
 * Salida: Ninguna
 *
 * Ejecuta un baile imitando una victoria
 */
  Change_vector(qn, 130, 50, 90, 90, 0, 180, 150, 40);
  coordinateAbsJ(full_robot, q0, qn, 0.5);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  delay(500);
  guard();
}
void Lose_Dance()
{
 /*
```

```
 * Entrada: Ninguna
 * Salida: Ninguna
 *
 * Ejecuta un baile de derrota
 */
 for (int i = 0; i < 3; i++)
 {
   Change_vector(qn, 110, 110, 90, 90, 120, 65, 90, 90);
   coordinateAbsJ(full_robot, q0, qn, 1);
   for (int i = 0; i < JOINTS; i++)
   {
     q0[i] = qn[i];
   }
   delay(500);
   Change_vector(qn, 80, 80, 90, 90, 120, 65, 40, 150);
   coordinateAbsJ(full_robot, q0, qn, 1);
   for (int i = 0; i < JOINTS; i++)
   {
     q0[i] = qn[i];
   }
 }
}
void Jap_Izquierda()
{
 /*
 * Entrada: Ninguna
 * Salida: Ninguna
 *
 * Ejecuta una imitación de un Jap Izquierda
 */
 Change_vector(qn, 90, 90, 75, 75, 30, 150, 150, 150);
 coordinateAbsJ(full_robot, q0, qn, 0.2);
 for (int i = 0; i < JOINTS; i++)
 {
   q0[i] = qn[i];
 }
 delay(500);
 guard();
}
void Gancho_Izquierda()
{
```

```
 /*
 * Entrada: Ninguna
 * Salida: Ninguna
 *
 * Ejecuta una imitación de un gancho de izquierdas
 */
  Change_vector(qn, 90, 90, 90, 90, 120, 120, 130, 150);
  coordinateAbsJ(full_robot, q0, qn, 0.2);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  delay(300);

  Change_vector(qn, 90, 90, 75, 75, 30, 150, 10, 150);
  coordinateAbsJ(full_robot, q0, qn, 0.2);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  delay(300);
  guard();
}
void Gancho_Derecha()
{
 /*
 * Entrada: Ninguna
 * Salida: Ninguna
 *
 * Ejecuta una imitación de un gancho de derechas
 */
  Change_vector(qn, 90, 90, 90, 90, 80, 85, 40, 50);
  coordinateAbsJ(full_robot, q0, qn, 0.2);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  delay(300);

  Change_vector(qn, 90, 90, 105, 105, 60, 180, 40, 160);
  coordinateAbsJ(full_robot, q0, qn, 0.2);
```

```
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  delay(300);
  guard();
}
void Jap_Derecha()
{
 /*
 * Entrada:Ninguna
 * Salida Ninguna
 *
 * Ejecuta una imitación de un Jap de derechas
 */
  Change_vector(qn, 90, 90, 105, 105, 60, 180, 40, 40);
  coordinateAbsJ(full_robot, q0, qn, 0.2);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  delay(500);
  guard();
}
void Back()
{
 /*
 * Entrada:Ninguna
 * Salida Ninguna
 *
 * Ejecuta un paso hacia atrás
 */
  Change_vector(qn, 75, 80, 102, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 75, 90, 102, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
```

```
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 75, 75, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 110, 105, 75, 75, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 105, 102, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 102, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
}
void Left()
{
 /*
 * Entrada:Ninguna
 * Salida Ninguna
 *
 * Ejecuta un paso a la izquierda
 */
  Change_vector(qn, 110, 105, 90, 80, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
```

```
  }
  Change_vector(qn, 90, 105, 80, 90, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 80, 90, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 80, 80, 80, 90, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 80, 90, 80, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 90, 80, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
}
void guard()
{
 /*
 * Entrada:Ninguna
 * Salida Ninguna
 *
 * Ejecuta una imitación de guardia
 */
  Change_vector(qn, 90, 90, 90, 90, 80, 120, 40, 150);
```

```
  coordinateAbsJ(full_robot, q0, qn, 0.2);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
}
void Right()
{
 /*
 * Entrada:Ninguna
 * Salida Ninguna
 *
 * Ejecuta un paso a la derecha
 */
  Change_vector(qn, 80, 80, 100, 90, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 80, 90, 90, 100, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 90, 100, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 110, 105, 90, 100, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 105, 100, 90, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
```

```
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 100, 90, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
}
void Forward()
{
 /*
 * Entrada: Ninguna
 * Salida Ninguna
 *
 * Ejecuta un paso hacia adelante
 */
  Change_vector(qn, 80, 80, 75, 71, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 80, 90, 102, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 102, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 110, 105, 102, 95, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
```

```
  }
  Change_vector(qn, 90, 105, 75, 71, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
  Change_vector(qn, 90, 90, 75, 71, 120, 65, 90, 90);
  coordinateAbsJ(full_robot, q0, qn, 0.7);
  for (int i = 0; i < JOINTS; i++)
  {
    q0[i] = qn[i];
  }
}
void Change_vector(double qn[], double j1, double j2, double j3, double
j4, double j5, double j6, double j7, double j8)
{
 /*
 * Entrada: Un vector vacío a rellenar, los ángulos deseados de cada
servomotor
 * Salida: Ninguna
 *
 * Renueva el vector a la siguiente configuración deseada
 */
  qn[0] = j1;
  qn[1] = j2;
  qn[2] = j3;
  qn[3] = j4;
  qn[4] = j5;
  qn[5] = j6;
  qn[6] = j7;
  qn[7] = j8;
}
void coordinateAbsJ(RobotServo_t robotServos[JOINTS], const double
q0[JOINTS], const double qT[JOINTS], const double T)
{
  double a[JOINTS], b[JOINTS], c[JOINTS], d[JOINTS], q, t, t0;
  /*Entrada: servos, posición de origen, posición final, tiempo de
ejecución
    Salida: Ninguna
```

```
   Con los datos proporcionados se calcula una trayectoria y se ejecuta
  */
  t0 = millis() / 1000.0;
  t = 0.0;
  while (t < T)
  {
    for (int i = 0; i < JOINTS; i++)
    {
      //TODO: cálculo de la trayectoria.
      a[i] = -(2 * (qT[i] - q0[i])) / pow(T, 3);
      b[i] = (3 * (qT[i] - q0[i])) / pow(T, 2);
      c[i] = 0;
      d[i] = q0[i];
      q = a[i] * pow(t, 3) + b[i] * pow(t, 2) + c[i] * t + d[i];
      //Ejecución
      writeServo(robotServos[i], (int)q);
    }
    delay(20);
    t = millis() / 1000.0 - t0;

  }
}
void writeServo(RobotServo_t &servo, int angle)
{
  /*
   * Entrada: Servo a mover,ángulo deseado
   * Salida: ninguna
   *
   * Mueve el servo al ángulo deseado
   */
  int pulse_width;
  angle = constrain(angle, servo.min_pos, servo.max_pos);
  pulse_width = map(angle + servo.offset, 0, 180, MIN_PWM,
MAX_PWM);
  servos.setPWM(servo.pin, 0, pulse_width);
}
void detachServo(RobotServo_t &servo)
{
  servos.setPWM(servo.pin, 0, 0);
}
```

# 2.Aplicación de control remoto

```
/*
 * En este código se ejecuta el control mediante una aplicación que envía caracteres en función del botón pulsado
 * Además se crean diversas funciones genéricas que permiten el movimiento del robot
 */

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

// Configuración de servos y Bluetooth
Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x40);
SoftwareSerial mySerial(7, 6); //RX, TX

// Definición de constantes
#define MIN_PWM 130
#define MAX_PWM 570
#define JOINTS 8

//Definición de variables de los servos
typedef struct
{
  uint8_t pin;
  int offset;
  int min_pos;
  int max_pos;
} RobotServo_t;

//Definición de posiciones
double q0[JOINTS] = {90, 90, 90, 90, 80, 120, 40, 150};
double qn[JOINTS];
//Definicion de los servos
RobotServo_t full_robot[JOINTS] = {{10, 10, 0, 140}, {11, 3, 40, 180}, {12, -5, 0, 180}, {13, 0, 0, 180}, {15, 0, 0, 180}, {14, 0, 0, 180}, {8, 0, 0, 180}, {9, 0, 0, 180}};

//Configuración inicial
void setup() {
  //Configuración de la frecuencia de los servos
  servos.begin();
  servos.setPWMFreq(60);
  //Configuración del puerto serie para visualizar los datos
  Serial.begin(9600);
  mySerial.begin(9600);
  Serial.println("Empiezo a recibir caracteres!");
  //Posición inicial
  guard();
  delay(3000);
}
```

```
void loop() {
  if (mySerial.available())
  {
    //Recepción de caracteres mediante Bluetooth
    char c = (char)mySerial.read();
    Serial.println(c);
    // Ejecución de un movimento en función del caracter recibido
    switch (c)
    {
      case 'f':
        Forward();
        break;
      case 'l':
        Right();
        break;
      case 'c':
        guard();
        break;
      case 'r':
        Left();
        break;
      case 'b':
        Back();
        break;
      case 'o':
        Gancho_Derecha();
        break;
      case 'z':
        Jap_Derecha();
        break;
      case 'k':
        Gancho_Izquierda();
        break;
      case 'j':
        Jap_Izquierda();
        break;
      case 'y':
        Lose_Dance();
        break;
      case 't':
        Win_Dance();
        break;
      case 'x':
        Jap_Derecha();
        delay(500);
        Jap_Izquierda();
        break;
      case 'p':
        Jap_Derecha();
        delay(500);
        Gancho_Derecha();
        break;
```

```
      case 'v':
        Jap_Derecha();
        delay(500);
        Gancho_Derecha();
        delay(500);
        Jap_Izquierda();
        delay(500);
        Gancho_Izquierda();
        break;
      case 'q':
        Jap_Derecha();
        delay(500);
        Jap_Derecha();
        delay(500);
        Gancho_Derecha();
        delay(500);
        Jap_Izquierda();
        delay(500);
        Jap_Izquierda();
        delay(500);
        Gancho_Izquierda();
        break;
      case 'g':
        Forward_Left();
        break;
      case 'h':
        Forward_Right();
        break;
      case 'n':
        Back_Left();
        break;
      case 'm':
        Back_Right();
        break;
    }
  }
}
```

# 3. Aplicación autónoma 1

```
/*
 * Ejecuta una aplicación en la que el robot se moverá hacia adelante hasta detectar un obstáculo
 * Una vez detectado ejecutará una serie de golpes
 * En el caso de dejar de detectar el objeto realizará un baile de victoria
 * En el caso de seguir detectando el objeto realizará un baile de derrota
 */

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

// Configuración de servos y Bluetooth
Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x40);
SoftwareSerial mySerial(7,6); //RX, TX
```

```
// Definición de constantes
#define MIN_PWM 130
#define MAX_PWM 570
#define JOINTS 8
#define echoPin 4 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 5 //attach pin D3 Arduino to pin Trig of HC-SR04
#define D 10

//Definición de variables de los servos
typedef struct
{
  uint8_t pin;
  int offset;
  int min_pos;
  int max_pos;
} RobotServo_t;

//Definición de posiciones
double q0[JOINTS] = {90,90,90,90,80,120,40,150};
double qn[JOINTS];
//Definicion de los servos
RobotServo_t full_robot[JOINTS] = {{10,10,0,140},{11,3,40,180},{12,-
5,0,180},{13,0,0,180},{15,0,0,180},{14,0,0,180},{8,0,0,180},{9,0,0,180}};

long duration; // variable for the duration of sound wave travel
int d,e; // variable for the distance measurement


//Configuración inicial
void setup() {
  servos.begin();
  servos.setPWMFreq(60);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
  guard();
  e = 0;
  delay(3000);
}

void loop() {
   Serial.println(e);
  // Aplicación automática
  if ((distancia() < D) && (e == 0))
  {
    Jap_Derecha();
    delay(500);
    Jap_Izquierda();
    delay(100);
    e++;
  }
```

```
  else if ((distancia() < D) && (e == 1))
  {
   Jap_Izquierda();
   delay(500);
   Gancho_Derecha();
   delay(100);
   e++;
  }
  else if ((distancia() < D) && (e == 2))
  {
   Jap_Izquierda();
   delay(500);
   Gancho_Derecha();
   delay(500);
   Jap_Derecha();
   delay(500);
   Gancho_Izquierda();
   delay(500);
   e++;
  }
  else if ((distancia() < D) && (e == 3))
  {
   Lose_Dance();
   delay(300);
   e = 0;
  }
  else if ((distancia() > D) && (e > 0))
  {
   Win_Dance();
   delay(300);
   e = 0;
  }
  else
  {
   Forward();
   e = 0;
  }
}
```

# 4. Aplicación autónoma 2

```
/*
 * Ejecuta una aplicación en la que el robot se moverá hacia adelante hasta detectar un obstáculo
 * Una vez detectado ejecutará una serie de golpes
 * En el caso de dejar de detectar el objeto realizará un baile de victoria
 * En el caso de seguir detectando el objeto realizará un baile de derrota
 */

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

```
#include <SoftwareSerial.h>

// Configuración de servos y Bluetooth
Adafruit_PWMServoDriver servos = Adafruit_PWMServoDriver(0x40);
SoftwareSerial mySerial(7,6); //RX, TX

// Definición de constantes
#define MIN_PWM 130
#define MAX_PWM 570
#define JOINTS 8
#define echoPin 4 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 5 //attach pin D3 Arduino to pin Trig of HC-SR04

//Definición de variables de los servos
typedef struct
{
  uint8_t pin;
  int offset;
  int min_pos;
  int max_pos;
} RobotServo_t;

//Definición de posiciones
double q0[JOINTS] = {90,90,90,90,80,120,40,150};
double qn[JOINTS];
//Definicion de los servos
RobotServo_t full_robot[JOINTS] = {{10,10,0,140},{11,3,40,180},{12,-
5,0,180},{13,0,0,180},{15,0,0,180},{14,0,0,180},{8,0,0,180},{9,0,0,180}};

long duration; // variable for the duration of sound wave travel
int d,e; // variable for the distance measurement


//Configuración inicial
void setup() {
  servos.begin();
  servos.setPWMFreq(60);
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed
  Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
  Serial.println("with Arduino UNO R3");
  guard();
  e = 0;
  delay(3000);
}

void loop() {
  if ((distancia() < 12)&&(0 < e < 3))
  {
    Back();
    delay(100);
```

```
    Jap_Derecha();
    Jap_Derecha();
    delay(100);
    Back();
    e++;
  }
  else if((distancia() < 12)&& (e == 3))
  {
    Jap_Izquierda();
    delay(500);
    Gancho_Derecha();
    delay(500);
    Jap_Derecha();
    delay(500);
    Gancho_Izquierda();
    delay(500);
    e++;
  }
  else if((distancia() < 12)&& (e == 4))
  {
    Lose_Dance();
    delay(300);
  }
  else if ((distancia() > 12) && (e > 0))
  {
    Win_Dance();
    delay(300);
  }
  else
  {
    guard();
    e = 0;
  }
}
```