Escuela Politécnica Superior de Jaén
Grado en Ingeniería Informática

Universidad de Jaén
Escuela Politécnica Superior
de Jaén

TRABAJO FIN DE GRADO

# DISEÑO Y PROTOTIPADO DE UN ROBOT AUTÓNOMO Y SENSIBLE AL CONTEXTO.

Alumno

**Manuel Valero Valero**

Tutor

**José María Serrano Chica**
(Departamento de Informática)

**Septiembre, 2021**


Universidad de Jaén

Departamento de Informática

Don José María Serrano Chica, tutor del Trabajo Fin de Grado titulado: '**DISEÑO Y PROTOTIPADO DE UN ROBOT AUTÓNOMO Y SENSIBLE AL CONTEXTO.**', que presenta Don Manuel Valero Valero, otorga el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2021

El alumno: El tutor:

Manuel Valero Valero José María Serrano Chica

| FICHA DEL TRABAJO FIN DE TÍTULO | |
|---|---|
| **Titulación** | Grado en Ingeniería Informática |
| **Modalidad** | Proyecto de Ingeniería |
| **Especialidad** (solo TFG) | Sistemas de Información |
| **Mención** (solo TFG) | Sin mención |
| **Idioma** | Español |
| **Tipo** | General |
| **TFT en equipo** | No |
| **Fecha de asignación** | 11/09/2020 |
| **Descripción corta** | Hoy en día, la disponibilidad de las últimas tecnologías, unida al abaratamiento de los componentes, facilitan que, cada vez más, dispositivos inteligentes de última generación se introduzcan en nuestros hogares para asistirnos en multitud de tareas cotidianas. En este trabajo, se pretende crear desde cero un agente inteligente y autónomo, basado en microcontroladores, con capacidad para interactuar con el ambiente y desplazarse por él. |

*“No hay deber más necesario que el de dar las*

*gracias” (Marco Tulio Cicerón)*

---

En primer lugar, quiero agradecer a mi familia, sin ellos no habría podido vivir esta etapa de mi vida que decidí emprender, a mis compañeros y ahora parte de mi vida, que he conocido en estos duros años, donde siempre he tenido su ayuda, motivación y paciencia. También dar las gracias a todo el conjunto docente, interno y externo, que ha contribuido a mi formación, y por último y no por ello menos importante, dar las gracias a D. José María Serrano Chica, por todo su apoyo y ayuda para la realización de este TFG

# Contenido

# Índice de ilustraciones

# Índice de tablas

## Abreviaciones

TFG: trabajo fin de grado.

SCAI: Servicios Centrales de Apoyo a la Investigación.

FDM: modelado por deposición fundida.

IR: ingeniería de requisitos.

LCD: pantalla de cristal líquido (liquid-crystal display).

IDE: entorno de desarrollo integrado (integrated Drijive Electronics).

I2C: circuito integrado (Inter-Integrated Circuit).

SCL: es el pin de la señal de reloj de la interfaz I2C.

SDA: es el pin de la señal de datos de la interfaz I2C.

SVG: Gráficos vectoriales escalables (Scalable Vector Graphics).

STL: Biblioteca de plantilla estándar (Standard Template Library).

# 1. INTRODUCCIÓN

El presente proyecto consiste en el diseño y fabricación de un prototipo de robot bípedo autónomo que interactúa con el entorno y, que servirá como base para futuros estudios.

La fabricación del robot parte de cero, desde la impresión de las piezas que forman el prototipo en impresora 3D, hasta su posterior montaje. Además del montaje mecánico del prototipo, también se realiza el diseño y fabricación de toda la electrónica del mismo, así como la adecuación e implantación de los módulos necesarios.

El bípedo se ha diseñado mediante el programa FreeCAD, montado mediante el sistema electrónico en Arduino y programado el sistema software con el lenguaje C++. Pudiendo decir que este prototipo es de bajo coste y asequible, ya que cualquiera con impresora 3D podría fabricarlo. Además, es un proyecto abierto a varias posibilidades de evolución, ya que la robótica es un área de investigación que está en constante desarrollo y crecimiento.

## 1.1. Antecedentes

Para la elección del presente trabajo, donde me he inclinado más en el desarrollo del diseño desde cero del bípedo y la interconexión de diferentes módulos en Arduino, ha tenido que ver mucho mi etapa de trabajo, donde ejercí la profesión de mecánico de automóviles. En ella, siempre estaba manipulando los diferentes componentes del automóvil, así como resolviendo problemas que, en varias ocasiones no podían solventarse de la manera convencional.

Pero no me llenaba como para dedicarme el resto de mi vida a ello, por lo que decidí iniciar mi otra pasión, la informática. En la que recorrí unos duros años que me han hecho comprender la inmensidad que abarca este campo de la ciencia, y he podido disfrutar a la vez que sufrir, mientras me iba dando poco a poco cuenta de la meta que quería alcanzar.

Como cualquier otro ser humano, nos genera satisfacción el poder ver los resultados después de un duro trabajo y esfuerzo, en mi caso, me resulta gratamente satisfactorio el ver como he podido crear un robot, capaz de interactuar en los comportamientos más básicos, como un ser humano.

Gracias a estas dos etapas, he podido adquirir los conocimientos necesarios para poder desarrollar este proyecto, en el cual, he disfrutado y me he nutrido de muchas tecnologías que antes había podido probar.

## 1.2. Objetivos

El presente proyecto tiene como meta el estudio del arte en la robótica y robots bípedos, para luego diseñar y fabricar un prototipo de robot autónomo imprimible y capaz de desplazarse caminando y esquivar los obstáculos de su entorno.

Para cumplir esta meta, se han desarrollado una serie de objetivos, en los cuales, el resultado final sería la capacidad de diseñar y crear un prototipo de robot autónomo, esto se conseguirá trabajando desarrollando varios objetivos, los cuales se describirán a continuación:

- Identificación de escenarios de funcionamiento del prototipo.
- Identificar los requisitos hardware y software necesarios.
- Adquisición de los elementos necesarios para la construcción del prototipo.

## 1.3. Estado del arte

A día de hoy, poca gente no sabrá que es el término robot, y dependiendo de si lo han leído en libros, visto en películas o en la vida real, darán una opinión diferente para referirse a ellos y a lo que pueden hacer.

Sin embargo, mucho antes de que la palabra robot fuera usada por primera vez, el ser humano ya creaba máquinas y dispositivos capaces de imitar movimientos o funciones de los seres vivos, siendo en el siglo XVIII en Europa, se desarrolló el famoso autómata, pato de Vauconson, en la Figura 1, una creación que representaba un pato y era capaz de realizar funciones básicas de alimentación, como ingerir comida, digerir y defecar [1].

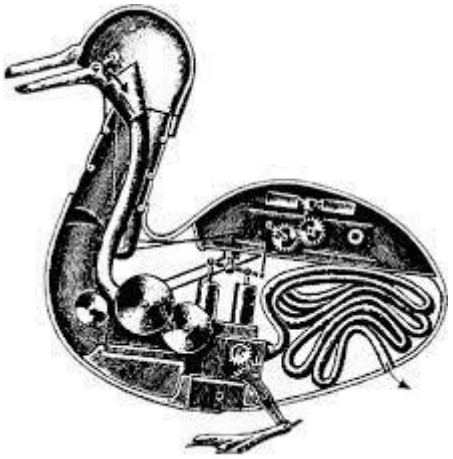


***Ilustración 1. Pato de Vaucanson***

Autómatas o robots, parecidos, pero con una gran diferencia. Técnicamente, un autómata es cualquier entidad artificial que se mueva por sí misma, sin asistencia humana, con este criterio, para que una entidad artificial sea un robot, no basta con ser automático. Además, es necesario que se autorregule, lo que significa que debe contar con tres capacidades: percibir lo que sucede a su alrededor, tomar decisiones con respecto a su entorno, y ejecutar acciones sobre el entorno en base a estas decisiones [2].

Actualmente existen una infinidad de robots, desde el primer robot de la historia llamado Elektro, en la figura 2, apodo que fue concebido por Joseph Barnett. Este humanoide media dos metros de altura y pesaba 120kg, podía caminar, fumaba cigarros, inflaba globos y movía la cabeza y brazos, todo esto unido a un vocabulario de 700 palabras. Su cuerpo era un engranaje de acero, ojos fotoeléctricos podían distinguir la luz roja y verde, con un cerebro elaborado con 48 relés [3].


***Ilustración 2. Humanoide Elektro***

Hasta Erica, construida en 2018 como parte de un proyecto entre dos de las universidades más importantes del mundo, la de Osaka y Kyoto. El comportamiento de Erica dejó perplejo a todo el mundo. [4]

***Ilustración 3. Humanoide Erica***

Con apariencia y textura similares a las del ser humano, tiene 15 sensores con los que es capaz de seguir con la mirada a las personas, y una inteligencia artificial que le ha llevado a trabajar como actriz robótica.

Hoy en día los robots son una parte indispensable de la sociedad, habiendo en España una demanda en la industria que supone una estimación de 113 unidades por cada 10.000 empleados, situando a España la treceava en el ranking mundial, en el cual, podemos destacar con diferencia a Singapur, con 918 unidades por cada 10.000 trabajadores, cifras que claramente no dejarán de subir. [5]

Sabiendo con los datos anteriormente nombrados, podemos observar que hay diferentes tipos de robots, los cuales se pueden clasificar en generaciones, teniendo un total de 5 generaciones:

**1ª generación**, son los conocidos como robots manipuladores, se utilizan principalmente para mover objetos, pero están limitados por sus movimientos, siendo su principal función la de realizar tareas repetitivas programadas.

**2ª generación,** capaz de obtener información limitada del entorno y de mayor tamaño que los anteriores, teniendo movimientos complejos y controlados con secuencias numéricas. Se usan en la industria automotriz.

**3ª generación,** siendo los primeros robots inteligentes debido a que son capaces de obtener información del entorno mediante deferentes sensores, son controlados por ordenador, e iniciando el desarrollo de lenguajes de programación.

**4ª generación,** estos son capaces de captar y controlar su entorno en tiempo real, poseen mejores sistemas sensoriales y conceptos de conducta.

**5ª generación,** son los que se usan en la época actual, directamente relacionados con la inteligencia artificial. Son cada vez más autónomos gracias a los avances tecnológicos, y capaces de imitar las funciones cognitivas de la mente humana.

Una vez descritas brevemente las diferentes generaciones que ha habido desde los comienzos de los robots, podemos ver al sector que pertenecen aplicando las funciones que desarrollan:

**Industriales,** encargados de realizar las tareas que se desarrollan en las fases de producción industrial, siendo normalmente manipuladores, trabajando de manera repetitiva y procesos complejos, situados en entornos controlados.

**De servicio,** son dispositivos móviles y generalmente autónomos controlados por ordenador, se utilizan para la realización de las tareas en entornos no controlados. Principalmente su función es la de asistir a las personas en trabajos repetitivos, peligrosos, etc. Habiendo varios tipos dentro de esta categoría de robots:

*Domésticos,* robots encargados de las tareas del hogar, haciendo trabajos rutinarios, como podemos destacar al más extendido en este ámbito, Roomba.

*Investigación,* son los usados en laboratorios, suelen ser específicos y con morfologías distintas.

*Exploración,* los móviles que se usan para la inspección y rastreo en entornos, siendo los principales de oruga o con cadenas, y aéreos o submarinos entre otros.

*Militares,* abarcan distintas morfologías según su función. Usados para las operaciones del ejército, pueden ser desde robots de transporte a gestionaos con control remoto, rastreadores, artificieros, etc.

*Médicos,* usados en cirugías, siendo robots con movimientos muy precisos donde el ser humano requiere de una mayor precisión. También podemos encontrar robots que ayudan a personas con dependencia. En esta misma categoría están los robots nano, los cuales están formados por componentes moleculares y suelen ser introducidos en el cuerpo para tratar diversas enfermedades. Actualmente se encuentran en fase de investigación y desarrollo. [6]

## 1.4. Fundamentos teóricos

### *1.4.1. Arduino*

Arduino, ¿qué es exactamente el proyecto Arduino?, es una plataforma de creación en código abierto, flexible y fácil de utilizar para los desarrolladores y creadores, permitiendo crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores le dan diferentes usos. [8]

El proyecto nació en 2003, cuando varios estudiantes del Instituto de Diseño Interactivo de Ivrea, Italia, con el fin de facilitar el acceso y uso de la electrónico y programación. Lo hicieron para que los estudiantes de electrónica tuviesen una alternativa más económica a las populares. El resultado fue Arduino, capaz de conectar periféricos a las entradas y salidas de un microcontrolador, y que podía ser programado en GNU/Linux, macOS y Windows, promoviendo la idea filosófica 'learning by doing'. [9]

Arduino es una placa con microcontroladores integrados en los que grabar las instrucciones, las cuales se escriben utilizando el entorno Arduino IDE. Los microcontroladores poseen interfaz de entrada, pudiendo conectar diferentes periféricos a la placa, pudiendo ser desde cámaras hasta sensores. También cuenta con interfaz de salida, procesando la información de Arduino a otros periféricos como pantallas o altavoces.

Por último, destacar que tenemos diferentes modelos de placas para la adaptación de las necesidades de cada proyecto, siendo las más comunes las placas de Arduino y ordenadas por su tamaño: Arduino Mega, Arduino Uno y Arduino Nano, siendo esta última la elegida para el desarrollo de este prototipo.

Arduino uno ha sido la elegida para este desarrollo ya que se ha intentado eliminar el mayor peso posible al bípedo. Para el uso de Arduino nano, hacía falta el uso de una board para conectar el Arduino, ya sea una comercial o diseñada por mi nosotros mismos, y Arduino mega es la que más prestaciones tiene, pero su tamaño, dificulta el poder acoplarla en este proyecto. No por elegir Arduino uno se ha reducido la calidad, ya que, tiene una gran cantidad de posibilidades en cuanto a los proyectos que puede soportar, teniendo las siguientes características que cumplen con lo que se demanda en este proyecto:

- Microcontrolador: ATmega328
- Voltaje de operación: 5V
- Voltaje de entrada (recomendado): 7-12V
- Voltaje de entrada (límites): 6-20V
- Pines de E/S digitales: 14 (de los cuales 6 proporcionan salida PWM)
- Pines de entrada analógica: 6
- Corriente DC por pin de E/S: 40 mA
- Corriente DC para 3.3V Pin: 50 mA
- Memoria Flash: 32 KB de los cuales 0,5 KB utilizados por el bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Velocidad de reloj: 16 MHz

Y aquí se ve el esquema pinout de la placa:

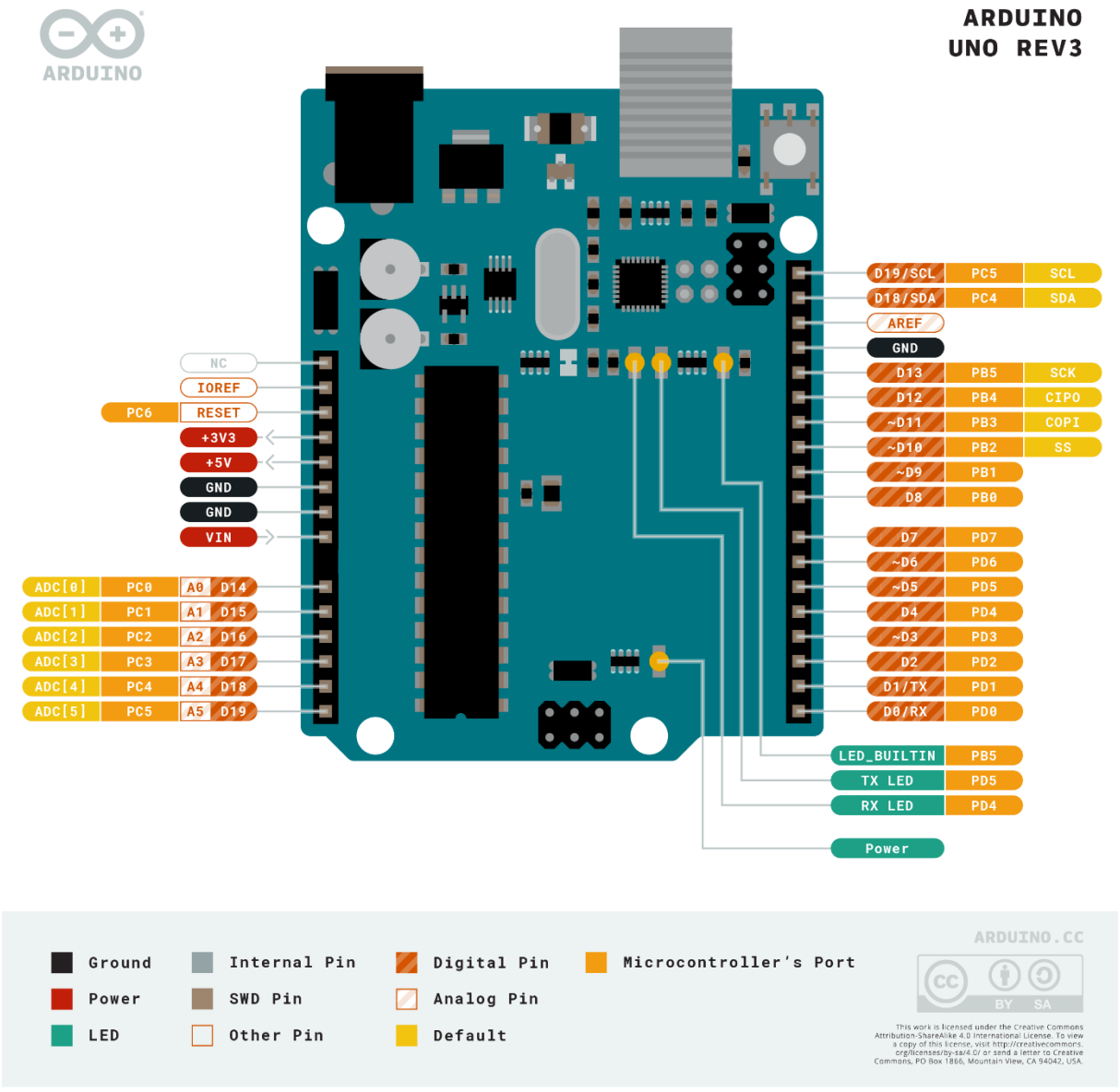



*Ilustración 4. Pinout Arduino Uno*

En esta imagen se pueden ver los pinout o la predisposición de pines y conexión que se encuentran en la placa.

### *1.4.2. Estudio mecánico*

En el presente apartado, se mostrarán los precedentes de la locomoción bípeda, conceptos sobre esta disciplina de la biomecánica y como solventarla con el fin de llegar al movimiento de nuestro bípedo.

La locomoción es la capacidad de un cuerpo para desplazarse de un lugar a otro, Esto se logra mediante la manipulación del cuerpo con respecto al entorno, teniendo muchas formas como pueden ser el volar de los pájaros, nadar de los peces o el caminar de los humanos. Los robots actualmente pueden caminar sobre sus propias piernas en entornos de pendientes y obstáculos, pudiendo argumentar que el medio más versátil y adecuado para la locomoción de los robots son las piernas.

Las piernas permiten superar discontinuidades del entorno, por lo que los robots bípedos son el aspecto practico de la locomoción con patas mecánicas, además, las piernas son una opción obvia para la locomoción en entornos de marcha humana: correr y trepar.

Los robots bípedos forman una de las clases de los robots con patas, hay que reconocer que gran parte del interés actual en robots bípedos, deriva de la existencia de máquinas que operan en forma animal (cuadrúpedos). La motivación del estudio de robots bípedos surge de los diversos intereses sociológicos y comerciales, que ven el deseo de reemplazar a los humanos en tareas hostiles, así como la restauración del movimiento en personas con discapacidades.

El proceso de caminar es más complejo de lo que parece, para el ser humano es ha terminado por ser un acto de memoria muscular, no nos damos cuenta de lo que hay detrás de cada paso, pies, cadera, torso, rodilla, brazos, etc. Tomando en cuenta el objetivo principal de este proyecto, reconocer los principios básicos que forman parte de la caminata humana es un objetivo fundamental para realizar el movimiento del robot bípedo.

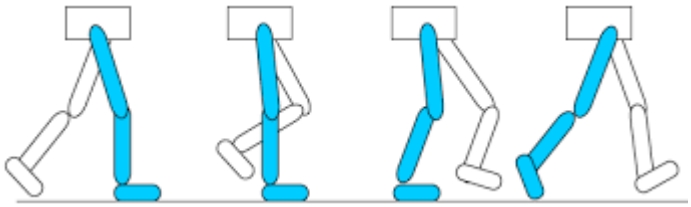


*Ilustración 5. Fases de la marcha bípeda*

En la Figura 4 se muestra las fases de la marcha bípeda fijando el objetivo en la pierna derecha, la caminata empieza con los dos pies extendidos sobre el suelo, donde obtenemos el mayor estado de equilibrio de la marcha, el problema comienza cuando levantamos una de las piernas para realizar el movimiento, ya que se está generando una alta probabilidad de que el bípedo pierda el equilibrio, para evitar que el robot bípedo caiga se deben realizar correctivos a los movimientos, para obtener una estabilidad mayor en la marcha bípeda.

El robot bípedo se basa en un diseño, en el cual, mediante una secuencia de movimientos mediante su cinemática directa, se logra una forma de caminar con un enfoque similar a la locomoción humana, Los movimientos son diferentes estados, generados por el movimiento de las articulaciones en ciertos ángulos determinados y posiciones finales, de tal manera, que permite al robot moverse. [7]

## 1.5. Recursos utilizados

Aquí se mostrará un listado y un breve resumen de los componentes que he utilizado para la elaboración del proyecto:

- **Placa Arduino uno R3,** basada en el microcontrolador mega, voltaje de entrada 7-12v, con 14 pines digitales, 6 salidas pwm y 8 entradas analógicas
- **Sensores de navegación,** sensor de ultrasonido HCSR04

- **Servos para rotación,** modelo MG90S micro servo con 180º y 4,8V.
- **Módulos,** tales como zumbador y pantalla led Diymore 1,3”
- **Alimentación,** se necesitan 2 baterías 18650 de iones de litio de 4,2 V y un portapilas para la conexión con la placa.
- **Ordenador,** necesario para el uso de los programas e instalación de librerías, diseño de impresión y programación del bípedo.
- **Impresora 3D FDM,** Witbox BQ proporcionada por (SCAI) Servicios Centrales de Apoyo a la Investigación, Área de Ingeniería y Computación Científica de la Universidad de Jaén.
- **Otros materiales,** tales como tornillos, tuercas, juntas tóricas, mini placa de pruebas, alargadores de cables, conector de barril para alimentación del Arduino.

## 1.6. Puntos a destacar

En este primer capítulo, se ha podido desarrollar algunos apartados que nos explican cómo se va a desarrollar este prototipo a lo largo del documento. A parte, se hace una introducción en algunos conceptos teóricos para entender mejor porqué se ha desarrollado dicho prototipo de la forma que se expone en el presente documento.

# 2. ANÁLISIS

En este capítulo de análisis, se va a exponer diferentes datos para que ayuden a su comprensión y conocimientos sobre el proyecto que se va a llevar a cabo en este documento. A parte, se hablarán de diversos apartados, dando una explicación del porque se han usado, como pueden ser, la metodología, los requisitos iniciales y el alcance del proyecto.

## 2.1. Metodología

La metodología es un conjunto de tecnologías y métodos integrados que permiten resolver de alguna manera, cada actividad en el ciclo de vida de un proyecto de desarrollo es homogénea y abierta, siendo un proceso de software detallado y completo.

Estos métodos se basan en una combinación de modelos de procesos comunes. Definen, artefactos, roles y actividades, junto con prácticas y técnicas recomendadas. La metodología de desarrollo de software es una implementación, gestión y la posibilidad de éxito en la gestión de un proyecto, para tener altas posibilidades de éxito. Una metodología para el desarrollo de software comprende el seguimiento sistemático para diseñar, implementar y mantener los productos de software desde el surgimiento de la demanda del producto hasta que lo satisfagamos.

Aplicar esto a la ingeniería del software, destacamos una metodología que:

- Optimiza en el proceso y en el producto software.
- Métodos que guían en la planificación y en el desarrollo del software.
- Define qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto.

Una metodología define una estrategia global para afrontar con el proyecto, los principales elementos que forman la parte de una metodología pueden ser:

- Fases: tareas a realizar en cada etapa.
- Productos: E/S de cada etapa, documentos.
- Procedimientos y herramientas: apoyo a la realización de cada tarea.

- Criterios de evaluación: del proceso y del producto, saber si se han logrado los objetivos.

En cuanto al desarrollo del software, es la clave de todo, se ha convertido en una industria con crecimiento vertical en los últimos años, pero también tiene problemas. La planificación y estimación del costo suelen ser imprecisas, falta de productividad y la calidad del software es a veces inaceptable.

Estos problemas generan insatisfacción y falta de confianza con el cliente, y como es evidente, los problemas anteriores generan otras dificultades:

- No hay tiempo para la recopilación de los datos necesarios para el desarrollo del software.
- Los proyectos se realizan con una vaga indicación de los requisitos del cliente.
- La calidad del software es cuestionable.
- Mantener el software es costoso y no se considera un aspecto importante.

Los problemas anteriormente nombrados se solucionan dándole un enfoque de ingeniería de software. [10]

Siendo la ingeniería del software, la recopilación de todos los conceptos que se han dado sobre la ingeniería del software, definiéndola como la disciplina o área de la informática, formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de programas informáticos, más conocidos como software. [11]

Sabiendo lo anteriormente nombrado, tenemos lo necesario para introducirnos en la metodología de trabajo y para que se utiliza. Pues bien, la metodología hace referencia al conjunto de procedimientos utilizados para alcanzar los objetivos que requerían de habilidades y conocimientos específicos. Siendo la metodología tradicional y la ágil las dos más usadas.

La metodología tradicional consta del desarrollo de un buen software que depende de un gran número de actividades y etapas, donde el impacto de elegir metodología para la realización no supone un proceso transcendental para el éxito del mismo. Las metodologías tradicionales son llamadas a veces de formas despectiva, como metodologías pesadas.

Centran su atención en llevar una documentación, exhaustica de todo el proyecto, planificación y control del mismo. Estas metodologías tradicionales imponen una disciplina más rigurosa de trabajos sobre el proceso de desarrollos del software, consiguiendo un software más eficiente, siendo las principales características

La metodología ágil nace como respuesta a los problemas que surgen en algunos casos en las metodologías tradicionales, basándose en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa, basando su fundamento en la adaptabilidad de los procesos de desarrollo.

Los modelos de desarrollo ágil suelen ser incrementales, entregas frecuentes, cooperativo, refiriéndonos, a que el cliente y desarrolladores trabajan con una constante comunicación, también es sencillo, lo que se traduce en que es fácil de aprender y modificable por el equipo de trabajo y adaptativo, siendo capaz de permitir cambios de última hora.

Con esto podemos decir, que las metodologías tradicionales tienen como principal problema el no conseguir una planificación con el esfuerzo requerido a seguir, pero si se consiguen definir métricas que apoyen la estimación de las actividades, muchas prácticas de esta metodología, podrían ser apropiadas. Pero tener diferentes metodologías para la aplicación de acuerdo a los proyectos que se vayan a desarrollar es una idea interesante, pudiendo involucrar practicas tanto de metodologías agiles como de las tradicionales, el problema vendría al definir cada una de ellas en el momento preciso y definir parámetros para saber cuál usar.

Es importante ser consciente de que a pesar de que a priori, la metodología ágil sería la más adecuada para cualquier proyecto, no es así. Pero sin embargo cabe destacar que dicha metodología tiene como virtud su curva de aprendizaje, siendo los métodos agiles inicialmente ligeros y por ello las personas que no están acostumbradas a seguir procesos, la encuentran bastante agradable.

Las principales diferencias entre estas dos metodologías se mostrarán en una tabla comparativa a continuación.

| Metodologías ágiles | Metodologías tradicionales |
|---|---|
| Basadas en heurísticas provenientes de prácticas de producción de código | Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo |
| Especialmente preparados para cambios durante el proyecto | Cierta resistencia a los cambios |
| Impuestas internamente (por el equipo) | Impuestas externamente |
| Proceso menos controlado, con pocos principios | Proceso mucho más controlado, con numerosas políticas/normas |
| No existe contrato tradicional o al menos es bastante flexible | Existe un contrato prefijado |
| El cliente es parte del equipo de desarrollo | El cliente interactúa con el equipo de desarrollo mediante reuniones |
| Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio | Grupos grandes y posiblemente distribuidos |
| Pocos artefactos | Más artefactos |
| Pocos roles | Pocos roles |
| Menos énfasis en la arquitectura del software | La arquitectura del software es esencial y se expresa mediante modelos |
| Poca documentación | Documentación exhaustiva |
| Muchos ciclos de entrega | Pocos ciclos de entrega |

***Tabla 1. Comparativa metodología ágil y tradicional***

Como se muestra en la tabla comparativa, se puede apreciar como las metodologías ágiles, son más baratas en tiempo y recursos, siendo los mismos o mejores los resultados ante las metodologías tradicionales.

En nuestro proyecto se han estudiado ambas metodologías y ambas serían más que válidas para la implementación del mismo, por lo que me he decantado por la metodología tradicional, en el cual, se estructura la realización del proyecto en varias etapas, con la finalización de una etapa, se continua con la siguiente, siguiendo la estructura que ya se indicó en el apartado Objetivos de este mismo documento.

## 2.2. Análisis de algunos de los robots bípedos

En este apartado, se mostrarán deviseros proyectos donde se asemeja con la idea de este proyecto y que, a su vez, han servido para entender mejor el extenso mundo de la robótica.


*Ilustración 6. Robot bípedo*

Boris, diseñado por Sebastian Coddington, es un robot bípedo muy bien ensamblado y utilizando el movimiento de los motores de una forma diferente a las que normalmente se ven. Poniendo en disposición los servos de las caderas de forma que el giro que realice, sea de izquierda a derecha en vez de arriba y abajo. Lo cual le hace que los giros estén muy bien depurados.

Este proyecto consta de 6 servos, una pantalla, sensor ultrasónico, brújula digital, modulo wifi y Arduino nano como componentes más importantes, pudiendo ser controlado mediante un mando controlador. [21]

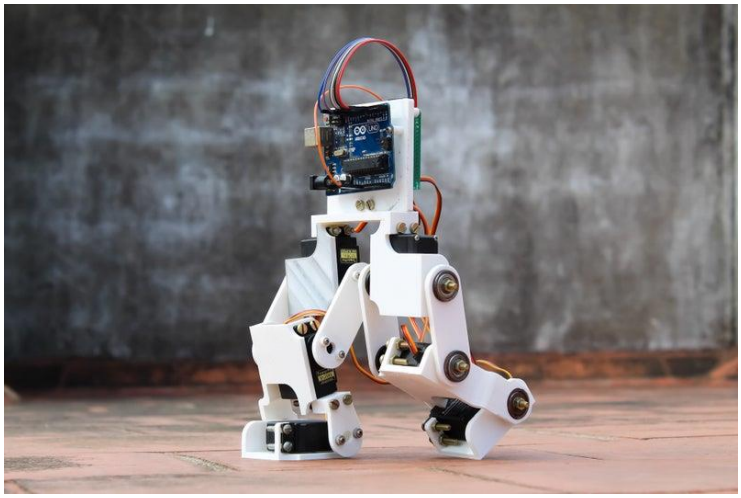


*Ilustración 7. Robot bípedo Technovation*

Este robot bípedo, ha sido desarrollado por Technovation. El diseño se asemeja más al común en los bípedos conforme la disposición de los servos. A parte, su diseño carece de parte superior o cara, lo que lo hace mucho más ligero y, el uso de los servos más potentes que los convencionales, además de las piezas que generan la rotación ha sido muy cuidadas, añadiendo el uso de rodamientos. Este conjunto hace que el movimiento sea muy fluido y la forma que ha sido diseñado, ayuda a que se pueda actualizar con mucha facilidad.

Este proyecto está formado por: 6 servos de grandes prestaciones, sensor ultrasónico y Arduino Uno como componentes más destacables. [22]

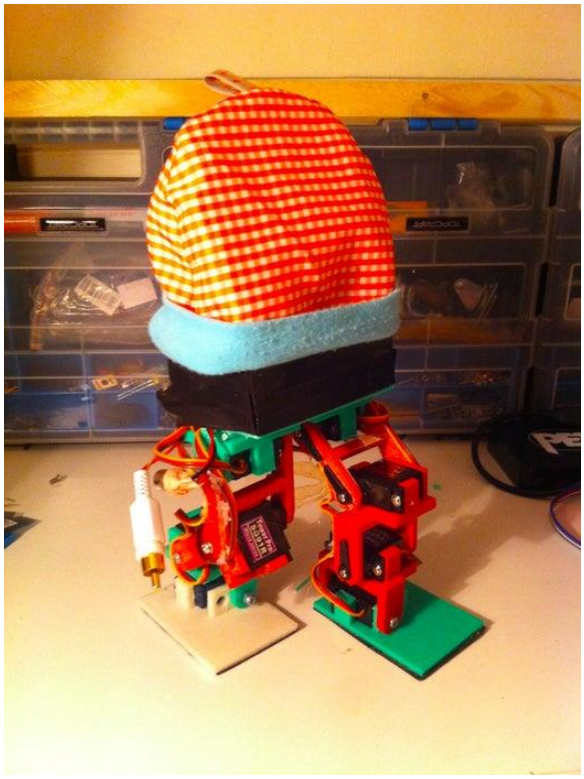


*Ilustración 8. Robot bípedo 4 servos*

Este bípedo esta desarrollado por Bonnetx, utiliza 4 servos para darle movimiento a pesar de llevar 2 servos en cada lado de la cadera haciendo un total de 6 servos. Su diseño es ligero y como el anterior bípedo, carece de parte superior lo cual le hace menos pesado, ayudándole a un mejor desplazamiento y equilibrio.

Los componentes más destacados de este proyecto han sido: 6 servos, aunque solo funcionan 4 de ellos, una placa Arduino nano y una placa casera para Arduino.[23]

## 2.3. Requisitos iniciales

La ingeniería de requisitos, es un conjunto de procesos, tareas y técnicas que permiten la definición y gestión de los requisitos de un producto de modo sistemático. Facilitando los mecanismos para comprender las necesidades del cliente, analizando, confirmando su viabilidad, especificando la solución sin ambigüedad para transformarla en un sistema operacional.

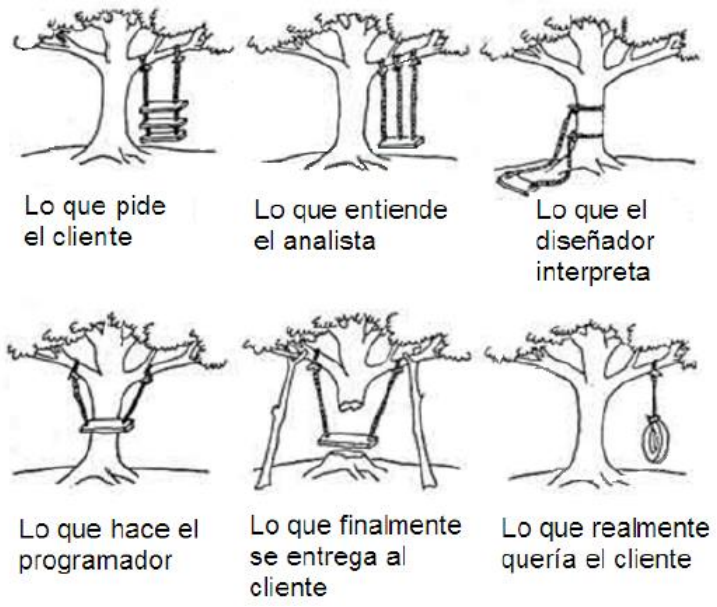



*Ilustración 9. IR diferentes puntos de vista*

La especificación de requisitos no es una tarea trivial, los usuarios no tienen claro lo que necesitan, no hay visión de conjunto, tampoco se sabe expresar de manera precisa lo que quieren y se considera algunos requisitos como asumidos y evidentes y no se expresan de manera explícita.

Tras esta breve introducción, podemos continuar con los tipos de requisitos que tenemos, pudiendo destacar los requisitos funcionales y no funcionales, siendo los primeros, los que describen cómo debe comportarse el sistema y, los no funcionales están relacionados con la calidad del sistema en desarrollo. [12]

| | |
|---|---|
| **RF-1** | Gestión de obstáculos. |
| **Autor** | Manuel Valero Valero. |
| **Descripción** | El sistema será capaz de evitar los obstáculos autónomamente. |

*Tabla 2. Requisito: gestión de obstáculos*

| | |
|---|---|
| **RF-2** | Gestión de sonido. |
| **Autor** | Manuel Valero Valero. |
| **Descripción** | El sistema será capaz de reproducir una melodía. |

*Tabla 3. Requisito: gestión de sonido*

| | |
|---|---|
| **RF-3** | Gestión de imagen. |
| **Autor** | Manuel Valero Valero. |
| **Descripción** | El sistema será capaz de mostrar diversas expresiones. |

*Tabla 4. Requisito: gestión de imagen*

| | |
|---|---|
| **RNF-1** | Estabilidad |
| **Autor** | Manuel Valero Valero. |
| **Descripción** | El sistema debe funcionar de forma estable y evitar, en la medida de lo posible, fallos durante su ejecución que puedan ocasionar la inmovilización del prototipo. |

*Tabla 5. Requisito: estabilidad*

| | |
|---|---|
| **RNF-2** | Disponibilidad |
| **Autor** | Manuel Valero Valero. |
| **Descripción** | El sistema podrá ser usado en cualquier momento. Esto se conseguirá gracias a una batería externa. |

*Tabla 6. Requisito: disponibilidad*

| | |
|---|---|
| **RNF-3** | Usabilidad |
| **Autor** | Manuel Valero Valero. |
| **Descripción** | El prototipo no entraña dificultad para la puesta en marcha del usuario. |

*Tabla 7. Requisito: usabilidad*

## 2.4. Alcance del proyecto

De acuerdo a las características del proyecto, en las que se enmarca este trabajo, se estableció un alcance adecuado tanto académicamente como desde el punto de vista del presupuesto disponible, siendo este último el más ajustado debido al costo de la impresión.

Se ha desarrollado, por lo tanto, un prototipo de robot bípedo simple, que camina utilizando un enfoque quizás algo más complejo a lo que normalmente se encuentra por la red, pero sin dejar de ser sencillo. Todos los materiales utilizados son de muy bajo costo y, los programas usados no implican desembolso para su utilización, por lo que no suponen dificultades para conseguir el resultado final de este prototipo.

## 2.5. Puntos a destacar

Para finalizar este capítulo, cabe destacar los puntos vistos anteriormente, en los cuales se estudia las metodologías más comunes, teniendo la tradicional y la metodología ágil como candidatas para este proyecto, también se ha hecho un seguimiento de los diferentes proyectos de robots bípedos y a mi parecer, más destacables que he encontrado por la red y, que se han estudiado.

A parte, se han estudiado los requisitos iniciales que se contemplan en este proyecto, teniendo varios funcionales y no funcionales, además, se ha desarrollado un alcance del proyecto donde se resume los motivos del porque se ha planteado el diseño que se va a desarrollar, siendo los académicos y económicos los principales motivadores en el resultado final del prototipo.

# 3. PLANIFICACIÓN

Planificación, en este capítulo del documento, se hablará sobre cómo se ha desarrollado el proyecto en su totalidad, mediante varias fases las cuales han sido nombradas anteriormente y se desglosaran y explicaran con sus correspondientes hitos en este capítulo.

A parte, se obtendrá el presupuesto del desarrollo total del TFG, dividido en diferentes partes, para así obtener información sobre el coste de las mismas y, los componentes que han sido adquiridos para el desarrollo del prototipo.

## 3.1. Planificación temporal

En este punto se van a describir las diferentes etapas de elaboración del presente TFG, detallando la duración en cada una de ellas, como se han llevado a cabo y, la duración completa de del TFG que, sumando el total, darían 263 horas divididas de la siguiente manera:

1º **FASE:** análisis (43 horas)

- ➔ Elección del tema.
- ➔ Definir el problema.
- ➔ Resolución del problema.

En esta primera fase, se escoge el tema del que tratará el TFG, en este caso, diseño y prototipado de un robot autónomo y sensible al contexto. Posteriormente se buscará el problema existente y la resolución del mismo con el prototipo, ideas iniciales, planificación, estudios previos, adquiriendo los conocimientos necesarios para el desarrollo, así como los aspectos económicos y viabilidad del proyecto.

**2º FASE:** construcción del marco teórico (96 horas)

- ➔ Búsqueda de fuentes de información.
- ➔ Diseño 3D del prototipo.
- ➔ Diseño del sistema software.

Para la segunda fase, se buscará información sobre cómo se diseña y crea el prototipo, estudiando las diferentes posibilidades que hay, para obtener la manera más eficaz y sencilla de cumplir los objetivos. Sin lugar a duda, esta fase ha sido la más tediosa, el desconocimiento hacía las tecnologías de desarrollo 3D, ha generado que tuviera que dedicarle la mayor parte del tiempo de esta fase.

**3º FASE:** codificación y diseño (35 horas)

- ➔ Construcción del prototipo.
- ➔ Implementación del software.

En esta fase, se desarrollará la construcción y ensamblaje de todas las piezas impresas, así como la implementación del software, uniendo todo para su correcto funcionamiento, además de darle la programación necesaria para que realice la funcionalidad deseada. Impresión de las piezas diseñadas, adaptación de los componentes previamente seleccionados, así como las diferentes versiones del desarrollo software que se implementarán, para una elaboración y diseño funcional.

**4º FASE:** pruebas (47 horas)

- ➔ Pruebas.
- ➔ Evaluación de los resultados.
- ➔ Depuración de errores.

A continuación, la parte crucial del TFG, donde se pone a prueba el prototipo, comprobando la funcionalidad, así como el ensamblaje del mismo, resolviendo y depurando los errores que vayan surgiendo, para dejarlo sin ningún tipo de complicación. Pruebas de las diferentes versiones del código software desarrollado, así como las prueba para obtener un resultado optimo y final del prototipo. Añadiendo ajustes de los componentes electrónicos para obtener un correcto funcionamiento.

**5º FASE:** finalización (42 horas)

- ➔ Documentación del TFG.
- ➔ Presentación del TFG.
- ➔ Defensa del TFG.

Última fase, documentar en la memoria de trabajo, todas las fases anteriormente nombradas junto a los pasos necesarios para la creación y uso del prototipo, una vez realizado esto, se pasará a presentar el TFG y a defenderlo.

Para entender las fases anteriormente nombradas y la planificación temporal del TFG, se mostrarán los hitos de una manera más visual mediante las siguientes graficas correspondientes a cada fase.

| Fase | Mes | Julio'20 | | Octubre'20 | | | Noviembre'20 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Semana | 2 | 3 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| Análisis | Elección del tema | | | | | | | | | |
| | Definir el problema | | | | | | | | | |
| | Resolución del problema | | | | | | | | | |

***Tabla 8. Fase análisis***

| Fase | Mes | Febrero'21 | | | | Marzo'21 | | | | Abril'21 | | | | Mayo'21 | | | | Junio'21 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Semana | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| Construcción marco teórico | Búsqueda de fuentes de información | | | | | | | | | | | | | | | | | | | | |
| | Diseño 3D del prototipo | | | | | | | | | | | | | | | | | | | | |
| | Diseño del sistema software | | | | | | | | | | | | | | | | | | | | |

*Tabla 9. Fase construcción marco teórico*

| Fase | Mes | Julio'21 | | | |
|---|---|---|---|---|---|
| | Semana | 1 | 2 | 3 | 4 |
| Codificación y diseño | Construcción del prototipo | | | | |
| | Implementación del software | | | | |

*Tabla 10. Fase codificación y diseño*

| Fase | Mes | Agosto'21 | | |
|---|---|---|---|---|
| | Semana | 1 | 2 | 3 |
| Pruebas | Pruebas | | | |
| | Evaluación de resultados | | | |
| | Depuración de errores | | | |

*Tabla 11. Fase pruebas*

| Fase | Mes | Agosto'21 | | Septiembre'21 | | |
|---|---|---|---|---|---|---|
| | Semana | 3 | 4 | 1 | 2 | 3 |
| Finalización | Documentación del TFG | | | | | |
| | Presentación del TFG | | | | | |
| | Defensa del TFG | | | | | |

Tabla 12. Fase finalización

## 3.2. Presupuesto

En este apartado que a continuación se desarrolla, se pretende realizar un presupuesto económico tanto de los esfuerzos realizados como de los materiales utilizados para la realización de este TFG. Aunque el ámbito en el que se mueve este proyecto es académico, no hay que olvidar que todo proyecto que quiera ser tratado

como tal, y más aquellos que son de carácter ingenieril, deben incluir junto con la memoria el correspondiente presupuesto en el que se contabilicen los gastos que supondría la realización del mismo prototipo en un futuro.

### *3.2.1. Costes de hardware*

A continuación, se muestra el presupuesto del coste del hardware, parte importante en este proyecto por las piezas 3D y los diferentes componentes del prototipo.

El primero y más costoso serían los diseños 3D, con un total de 22 piezas impresas, que se muestran en la siguiente tabla que se usó para solicitar la impresión:

| Nombre pieza | Peso | Dimensiones(mm) | Cantidad |
|---|---|---|---|
| BaseBateria | 8gr | 45x29x22 | X1 |
| BaseBoardV2.1 | 9gr | 58x60x43 | X1 |
| BaseFijacionPantalla | 8gr | 35.3x56.6x6 | X1 |
| BaseYcaraInferiorRPRotada | 99gr | 153x110x52 | X1 |
| CaraSuperiorV3.1 | 82gr | 152.1x102x46 | X1 |
| PieDerV2.0 | 12gr | 48.5x71.8x19 | X1 |
| PieIzqV2.0 | 12gr | 48x71.8x19 | X1 |
| PuenteVersionFinal | 8gr | 15x39x48.5 | X8 |
| SejeccionServoFinal | 7gr | 21x18x33.5 | X6 |
| SujeccionModuloApoximidad | 13gr | 55.7x70x26.5 | X1 |

*Ilustración 10. Tabla descriptiva diseños 3D*

Estos diseños han sido impresos en (SCAI) Servicios Centrales de Apoyo a la Investigación, Área de Ingeniería y Computación Científica de la Universidad de Jaén. Con unas tarifas de impresión que se muestras a continuación:

**TARIFAS FABLAB UJAEN**

| Equipo | | Unidad | Miembros Comunidad Universitaria | Empresas Externas | |
|---|---|---|---|---|---|
| Impresora 3D FDM | BQ Witbox | €/g | 0,21 | 0,42 | |
| | AirWolf 3D HD | €/g | 0,22 | 0,44 | |
| Impresora 3D SLA | Formlabs Form1+ | €/g | 0,72 | 1,44 | |
| Impresora 3D PolyJet | Stratasys Objet30 | €/g | 1,13 | 2,26 | |
| Termoformadora | Formech 508DT | €/plancha | 3,16 | 6,32 | *El coste final puede variar en funcion del material |
| Cortadora Vinilo | Roland Camm-1 GS-24 | €/h | 10,71 | 21,42 | |
| Fresadora PCB | LPKF Protomat S63 | €/placa | 37,83 | 75,66 | |
| | | €/(1/2 placa) | 18,92 | 37,84 | |
| | | €/(1/4 placa) | 9,46 | 18,92 | |
| Escaner 3D | HDI Advance 3D Scan | €/h | 19,52 | 39,04 | |
| Fresadora | Alarsis FR210 | €/h | 18,50 | 37 | *Coste sin material |
| Cortadora Laser | BCN3D IGNIS | €/h | 14,24 | 28,48 | *Coste sin material |
| Reometro | DHR HR-2 | €/ensayo | 15,22 | 30,44 | |
| Inyectadora de Plastico | Babyplast | €/h | 20,58 | 41,16 | *Coste sin material ni molde |
| Tronzadora | TLG-352-A | €/h | 12,04 | 24,08 | |
| Lijadora | Optimum DBS 75 | €/h | 11,59 | 23,18 | |
| Sierra | Holzkraft HBS 633 S | €/h | 17,63 | 35,26 | |
| Taladradora | Optimum Optidrill B32 | €/h | 12,62 | 25,24 | |

**DESCUENTOS**
1) Impresiones superiores a 100 gramos obtienen un 30% de descuento.
2) Impresiones superiores a 200 gramos obtienen un 50% de descuento.
3) Encargos de más de 4 piezas (con peso superior a 25 g) tienen una reducción del 21% IVA.

*Ilustración 11. Tarifa impresión 3D*

Como se muestra en la Figura 7, tenemos dos costes diferentes, la comunidad universitaria y empresas externas. Para este prototipo se ha elegido la impresión en FDM, siendo la más económica pero la más idónea para el prototipo que hemos desarrollado, por lo que teniendo estos datos podemos obtener el presupuesto para la impresión de los diseños.

Para el peso total de los diseños tenemos 349gr, lo que supone un coste total de 73,29€ para el miembro de la comunidad universitaria, si lo calculamos para una empresa externa sería un total de 146,58€. Esto sería el coste total, pero como se muestra en las tarifas, al tener un peso superior a 200gr, se realiza un descuento del 50%, por lo que los anteriores presupuestos se quedarían en:

- Comunidad universitaria 36,65€.
- Empresa externa 73,29€.

*Ilustración 12. Piezas impresas 3D*

El siguiente componente hardware es la placa de Arduino, en este prototipo se ha usado la versión Uno R3, teniendo un coste de 9,99€


*Ilustración 13. Precio Arduino Uno R3*

Ahora se verá el precio de los servos, modelo owootecc MG90S de 10 unidades con un precio total de 35,99€.

*Ilustración 14. Precio servos MG90S*

Los siguientes componentes usados en este TFG son:

- Conector de barril para Arduino 1,74€

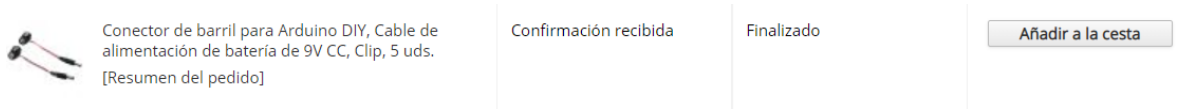



*Ilustración 15. Precio conector corriente*

.

- Módulo de pantalla LCD 1,3” 4,32€

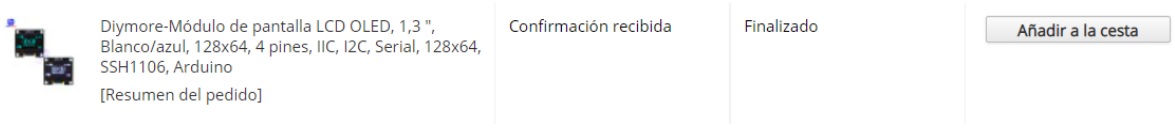



*Ilustración 16. Precio módulo pantalla LCD*

- Cables puente 2,67€ y mini placas 1,40€


*Ilustración 17. Precio Cable puente y mini placas*

- Modulo ultrasónico 1,56€

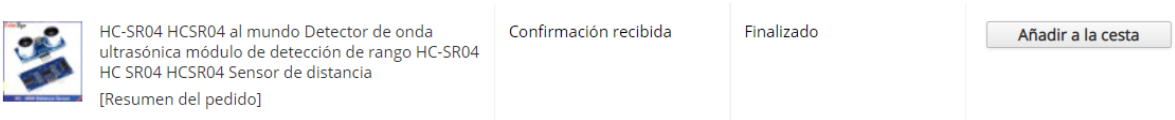



*Ilustración 18. Precio módulo ultrasónico*

- Tornillos de acero inoxidables 2,78€

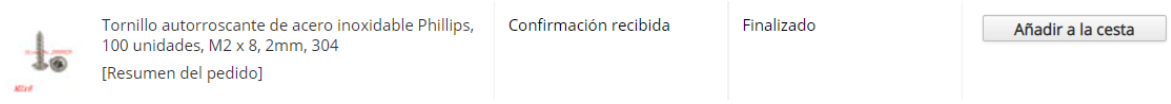



*Ilustración 19. Precio tornillos*

- Módulo zumbador pasivo 2,66€

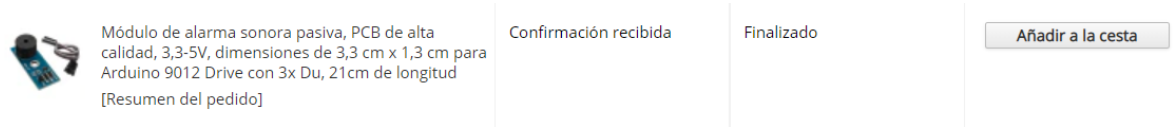



*Ilustración 20. Precio zumbador pasivo*

- Caja porta baterías 2,36€

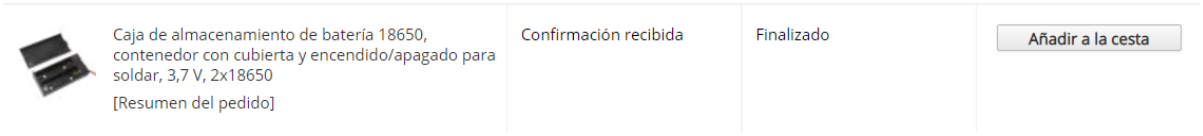



*Ilustración 21. Precio caja porta baterías*

- Baterías 18650 16,80€


*Ilustración 22. Precio baterías*

Con estos últimos componentes y sin añadir la impresión, el presupuesto es de 82,88€, ahora dependiendo de si somos de la comunidad universitaria o empresa externa, debemos añadírselo a este presupuesto, en mi caso al ser de la comunidad universitaria, le sumaria 36,66€, y si somos empresa externa sería 73,29€. Esto se resume en:

| Concepto | Coste (€) |
|---|---|
| Impresión diseños 3D comunidad universitaria | 36,65 |
| Servos MG90S | 9,99 |
| Arduino Uno R3 | 35,99 |
| Conector de barril para Arduino | 1,74 |
| Módulo de pantalla LCD 1,3” SSH1106 | 4,32 |
| Cable puente dupont 10cm | 2,67 |
| Mini placas YZ-25 | 1,40 |
| Módulo ultrasónico HCSR04 | 1,56 |
| Tornillos acero inoxidable M2 x 8, 2mm | 2,78 |
| Módulo zumbador pasivo | 2,66 |
| Caja porta baterías 18650 | 2,36 |
| Baterías recargables 18650 | 18,8 |
| | |
| **TOTAL** | **119,54** |

*Tabla 12. Coste hardware comunidad universitaria*

| Concepto | Coste (€) |
|---|---|
| Impresión diseños 3D empresas externas | 73,29 |
| Servos MG90S | 9,99 |
| Arduino Uno R3 | 35,99 |
| Conector de barril para Arduino | 1,74 |
| Módulo de pantalla LCD 1,3” SSH1106 | 4,32 |
| Cable puente dupont 10cm | 2,67 |
| Mini placas YZ-25 | 1,40 |
| Módulo ultrasónico HCSR04 | 1,56 |
| Tornillos acero inoxidable M2 x 8, 2mm | 2,78 |
| Módulo zumbador pasivo | 2,66 |
| Caja porta baterías 18650 | 2,36 |
| Baterías recargables 18650 | 18,8 |
| | |
| **TOTAL** | **156,17** |

*Tabla 13. Coste hardware empresas externas*

### 3.2.2. Costes de software

Para el coste del software, solo se ha tenido en cuenta lo relacionado con el código que se ha desarrollado, ya que los programas han sido libres, lo cual no se ha necesitado de licencias para el uso de los mismos.

En cuanto a la mano de obra por el desarrollo del código, se ha buscado el salario de un ingeniero informático, que oscila entre los 24.000€ de sueldo base al año según linkedin. A lo que debemos calcular las horas del desarrollo del software invertidas, por lo que teniendo lo anteriormente nombrado, tenemos un coste de 280€ por el desarrollo del software.

| Concepto | Coste (€) |
|---|---|
| Desarrollo del código | 280 |
| | |
| **Total** | **280** |

*Tabla 14. Coste software*

### 3.2.3. Costes de personal

En cuanto al personal, como se ha visto en el apartado anterior, se ha requerido de in ingeniero informático, por lo que el valor por los conocimientos sería de 300€.

Por otra parte, tenemos el desarrollo de las piezas 3D y la parte electrónica, lo que requiere de dos personales más, ingeniero mecánico e ingeniero electrónico. Como se ha consultado en el apartado anterior, el salario tanto para el personal electrónico como para el mecánico, se e ha mirado en linkedin y, cuyo salario oscila entre los 24.000€ de sueldo base al año, calculando las horas de desarrollo y conocimientos de este personal, tenemos un coste de 1080€ para el mecánico y para el electrónico sería de 410€.

| Concepto | Coste (€) |
|---|---|
| Ingeniero informático | 300,0 |
| Ingeniero mecánico | 1.080 |
| Ingeniero electrónico | 410,0 |
| | |
| **Total** | **1.790** |

*Tabla 15. Coste personal*

### 3.2.4. Costes totales

Por último, tenemos el coste total del prototipo, el cual hemos descrito en los puntos anteriores y mostraremos en una tabla resumen.

| Concepto | Coste (€) |
|---|---|
| Coste hardware universitaria | 119,54 |
| Coste software | 280,00 |
| Coste personal | 1.790,00 |
| | |
| **TOTAL** | **2.189,54** |

*Tabla 16. Coste total universitaria*

| Concepto | Coste (€) |
|---|---|
| Coste hardware empresas externas | 156,17 |
| Coste software | 280,00 |
| Coste personal | 1.790,00 |
| | |
| **TOTAL** | **2.226,17** |

*Tabla 17. Coste total empresa externa*

Con estos datos podemos decir que, el presupuesto para una persona que pertenezca a la comunidad universitaria, y sin conocimientos de ninguno de los que se necesitan para el desarrollo de este prototipo es de 2.189,54€. Mientras que para una persona externa a la universidad y con las mismas condiciones, tendría un presupuesto de 2.226,17€.

## 3.3. Puntos a destacar

Tras ver el contenido de este capítulo, cabe destacar la duración del desarrollo del TFG, que rondan las 260 horas de dedicación. Estas horas ha sido la suma de las diferentes fases las cuales, han tenido diferencia de horas por la complejidad de las mismas.

En cuanto al presupuesto, se ha explicado de la mejor manera posible y contabilizando todos los componentes u acciones que han podido generar un aumento del coste final, pudiendo destacar los dos valores finales que dependerían de un único factor, teniendo un coste total para los miembros universitarios de 2.189,54€ y, para el personal externo a la universidad de 2.226,17€.

# 4. ROBOT CON NAVEGACIÓN AUTÓNOMA

Llegamos al capítulo más extenso y donde se desarrollará el porqué de cada acción que se ha tenido en cuenta para el resultado final, así como los dibujos técnicos desarrollados y los componentes usados. Todo para que, con la comprensión lectora y las imágenes correspondientes, se pueda captar todo lo que ha influido en el diseño final del prototipo.

A parte se hará una breve introducción personal para entender mejor porque se ha elegido este TFG y el desarrollo de este prototipo. Además de la muestra del código y su explicación del porque se ha desarrollado de esta forma, así como las tecnologías usadas y los problemas que han ido surgiendo conforme se iba avanzando en el desarrollo del prototipo.

## 4.1. Robot

Siempre he tenido la curiosidad y ganas de tener un robot creado por mí mismo, gracias a los conocimientos obtenido, he podido iniciar un prototipo, que deseo continuar y perfeccionar.

La idea del diseño inicial era la de un humanoide, pero tras obtener la información de los componentes necesarios, usando un mínimo de 12 servos para la simulación de los movimientos y tener un humanoide lo más parecido a un humano, esto diseño generaba también un aumento de la calidad de los servos, lo que incrementaba el presupuesto, a parte, había que imprimir 4 extremidades, más costo al presupuesto final. Viendo lo que necesitaba para el desarrollo, decidí aparcar esa ida para este prototipo, dado que el presupuesto en estos momentos dispongo.

Esto me llevo a pensar en dos posibles opciones para el diseño, ¿cuadrúpedo o bípedo?, ambos accesibles, pero con diferencias en la movilidad y la más importante, el equilibrio.

El diseño de un cuadrúpedo reduciría significativamente la dificultad en cuanto a la estabilidad de los movimientos del robot, dado a sus cuatro apoyos, reparto del peso y menor altura, harían que este problema y junto al equilibrio, pasaran a desaparecer.

En cuanto al bípedo, acarrearía los problemas anteriormente mencionados que solventa la versión cuadrúpeda, pero personalmente me gustaba más el diseño del bípedo ya que se asemejaba más al humanoide que me gustaría desarrollar, además, era la primera idea que tenía en mente con este TFG y la cual más me ilusionaba realizar.

Teniendo claro el objetivo del bípedo, encontré un proyecto el cual me inspiro, Boris[1]. Este proyecto me ayudo a tomar los primeros pasos para el desarrollo de mi prototipo. Teniendo una ida de los componentes que necesitaría, faltaba tener un diseño el cual encajaran, esto me llevo tiempo ya que, no sabía nada sobre diseño 3D.

Una vez visto y desarrollado el diseño e informado sobre los componentes de este prototipo, decidí como quería desarrollar mi robot. La primera idea que tenía en mente y quería desarrollar era la de un bípedo con la forma las patas simulando a los terópodos[2]

[1] Boris: https://www.instructables.com/BORIS-the-Biped-for-Beginners-and-Beyond/

[2] Terópodo: “pie de bestia”, dinosaurios de gran tamaño, que se caracterizan por sus grandes patas traseras. https://www.asturnatura.com/articulos/fosiles/teropod.php

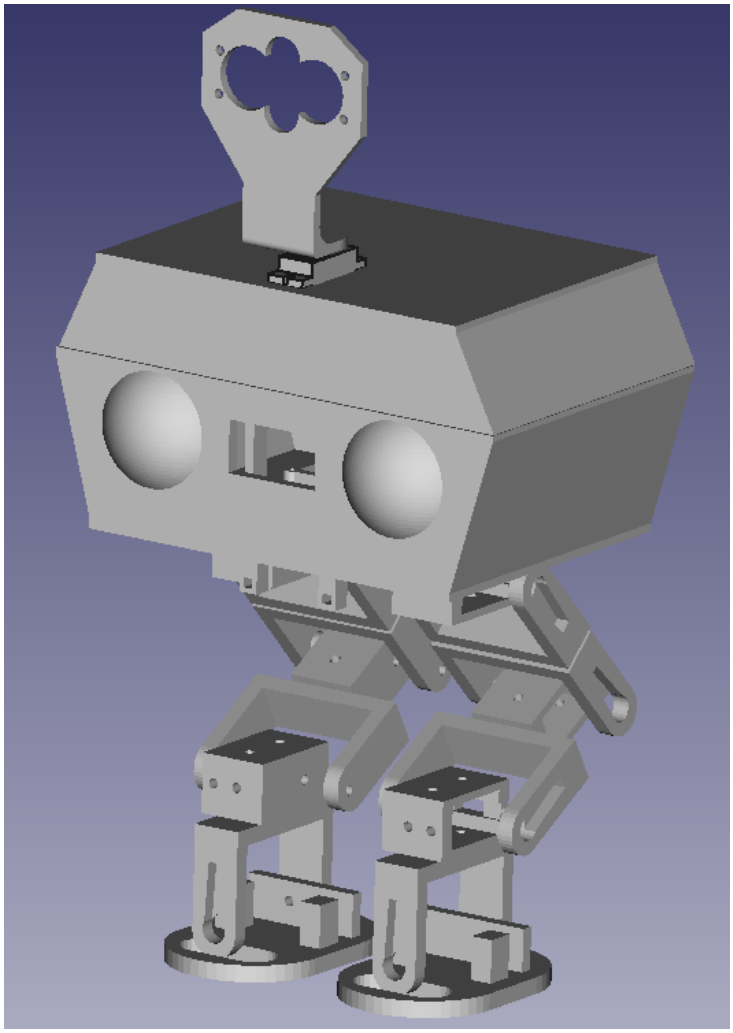


*Ilustración 23. 1ª versión prototipo bípedo*

Como se puede observar en la Figura 6, se intentó conseguir el rasgo característico de las extremidades de los dinosaurios terópodo, dándole una flexibilidad mayor al movimiento de la marcha bípeda. Este diseño incrementaba los servos utilizados en cada extremidad, lo cual, incrementaba el peso total, algo de lo que había que tener en cuenta dado que los servos utilizados, a pesar de ser la versión con engranajes metálicos, y no los de plástico, no podía exceder el peso, ya que tampoco usábamos los idóneos para los humanoides que soportan pares superiores de tensión.

Esto me hizo pensar en la segunda idea, el diseño e imitación de las piernas de los humanos, reduciendo ese peso extra al añadir más componentes, usando un servo por cada articulación, pie, rodilla y cadera. Un total de 6 servos, suficientes para generar el movimiento y el desarrollo del robot bípedo a bajo coste.

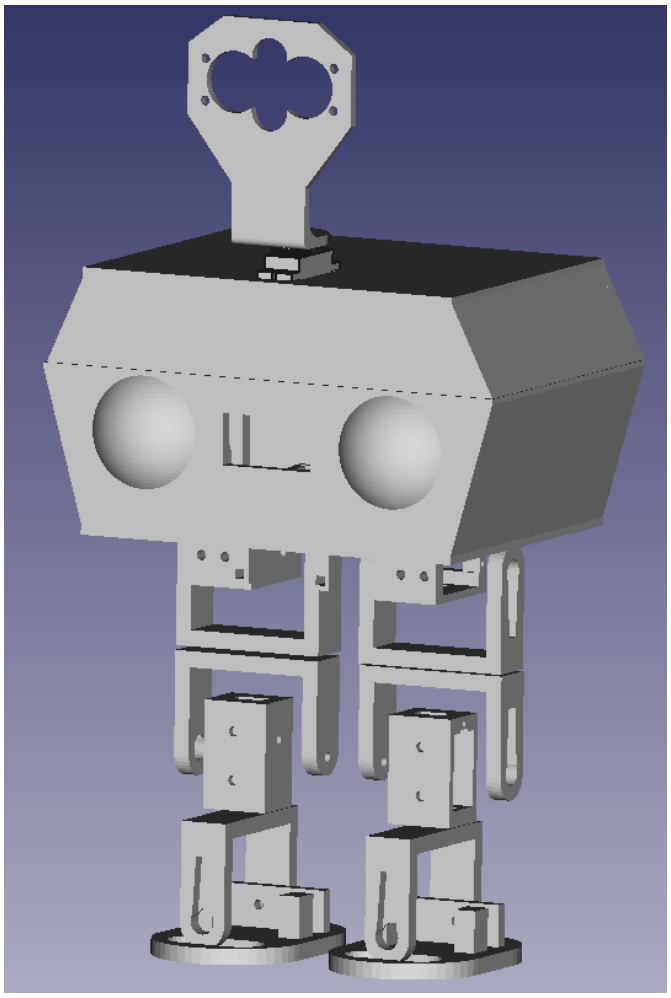


*Ilustración 24. 2ª versión prototipo bípedo*

## 4.2. Componentes

Para el desarrollo de este TFG, han sido necesarios varios componentes para simular y obtener lo más aproximadamente posible, el comportamiento de un humano al interactuar con obstáculos.

### *4.2.1. Arduino uno R3*

Basada en el microcontrolador basada en ATmega328P, voltaje de entrada 7-12v, con 14 pines digitales, 6 salidas pwm y 8 entradas analógicas, botón de reinicio, conector de alimentación y una conexión USB. También tiene incorporado un led controlado por el pin digital 13. La palabra uno se eligió por ser la primera placa con soporte USB.

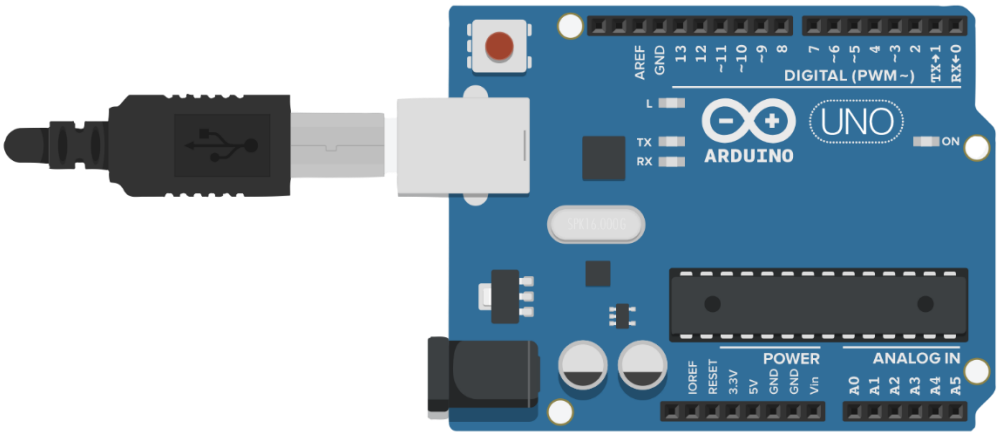



*Ilustración 25. Placa Arduino uno*

### 4.2.2. Sensor ultrasónico

El sensor ultrasónico HCSR04 es un sensor de distancia de bajo coste, su uso es muy frecuente en la robótica, utilizando transductores de ultrasonido para detectar objetos. Su funcionamiento consiste en emitir un sonido ultrasónico por uno de sus transductores y esperar el rebote del sonido, el eco es captado por su otro transductor.

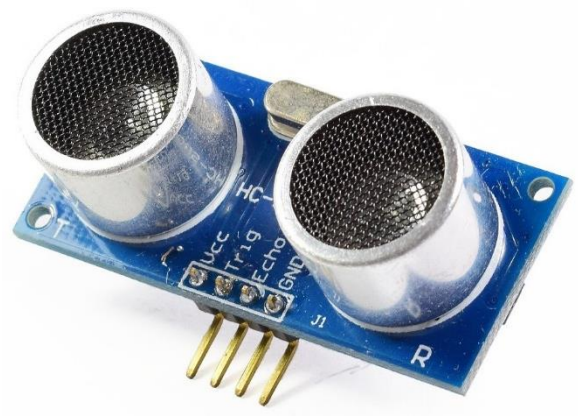



*Ilustración 26. Sensor ultrasónico HCSR04*

En nuestro proyecto se ha usado para cambiar el comportamiento del prototipo, cuando la distancia en centímetros llega o disminuye de 25cm.

### 4.2.3. Alarma sonora pasiva

Un buzzer o zumbador es un dispositivo capaz de enviar avisos a través del sonido, teniendo un circuito adicional que lo hace más fácil de usar. Este circuito es un oscilador, que hace que el zumbador suene al recibir alimentación.


*Ilustración 27. Módulo zumbador*

Existen dos tipos de zumbadores, pasivos y activos:

- Los activos solo necesitan ser alimentados por corriente continua (5V) para emitir un sonido debido a que disponen de un oscilador interno.
- Los pasivos necesitan que el pin de la placa proporcione una señal oscilatoria a una determinada frecuencia mediante la instrucción tone().

Siendo la principal ventaja de los pasivos, el poder reproducir melodías mediante la variación de la frecuencia, por este motivo se ha elegido este tipo de zumbador para su uso en el proyecto.

| TONO | FRECUENCIA EN QUINTA OCTAVA (Hz) |
|---|---|
| Do (C) | 523 |
| Re (D) | 587 |
| Mi (E) | 659 |
| Fa (F) | 698 |
| Sol (G) | 783 |
| La (A) | 880 |
| Si (B) | 987 |

***Tabla 18. Frecuencia notas zumbador***

### *4.2.4. Pantalla LCD*

Módulo de pantalla LCD 1,3'' modelo SSH1106. Se trata de una pantalla de diodos orgánicos de emisor de luz capaces de consumir muy poca energía, se comunican por I2c[3], con un tamaño de los pixeles de 128x64.

[3] I2C: https://programarfacil.com/blog/arduino-blog/comunicacion-i2c-con-arduino/

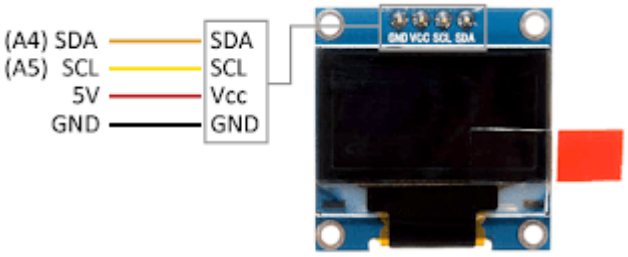



*Ilustración 28. Módulo pantalla LCD*

En el proyecto, se le ha dado el uso para mostrar dos tipos de imágenes diferentes, que varían entre si al encontrar un obstáculo o no.

### *4.2.5. Servos MG90S*

Servo motor tamaño micro, reforzado con engranajes metálicos y gran robustez. Usado principalmente en proyectos de robótica, tiene un conector tipo S que encaja con la mayoría de los receptores de radio control. Los cables están distribuidos de la siguiente forma: rojo → alimentación (+), marrón → alimentación (-) y naranja → señal PWM[4].

Sus principales características son:

- **Tipo de interfaz:** Analógica
- **Dimensiones:** 40.6 x 19.8 x 42.9 mm (1.60 x 0.78 x 1.69 pulgadas)
- **Peso:** 55 gramos
- **Torque a 4.8 volts:** 8.9 oz/in (10.00 kg/cm)
- **Voltaje de operación:** 4.0 a 7.2 volts
- **Velocidad de giro a 4.8 volts:** 0.2 s / 60 º
- **Conector:** universal para la mayoría de los receptores de radio control

[4] PWM: modulación de ancho de pulso.


*Ilustración 29. Servomotor MG90S*

Para el proyecto se han usado un total de 7 servos, 3 para cada pierna (pie, rodilla, cadera) y uno para el sensor ultrasónico.

### *4.2.6. Alimentación*

La alimentación se genera por parte de unas baterías 18650 de 8800mah Li-ion a 4,2v recargables. Gran parecido a la pila tipo AA, tiene una ventaja y es su bajo nivel de autodescarga y ausencia de efecto de memoria. Sus dimensiones están cifradas con su nombre, 18 indica el diámetro (18mm) y 650 es su longitud (65mm). Tienen una larga vida útil que va de los 500 ciclos de carga a los 1000 ciclos. Con una duración aproximada de 30 minutos en este proyecto, llegando a las 2 horas con baterías 18650 de mejor calidad.

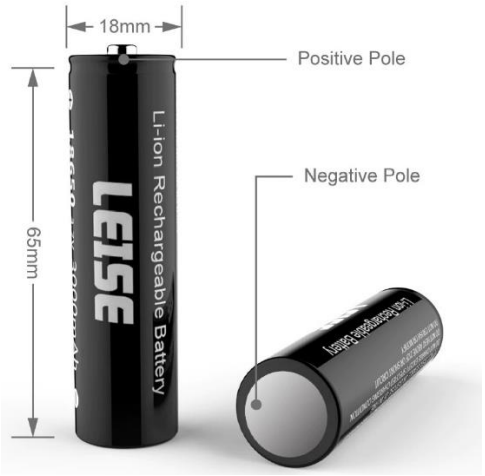



*Ilustración 30. Batería 18650*

En este proyecto se han usado dos baterías, para superar al voltaje mínimo que necesita Arduino para su alimentación externa, ya que nos pide 6v y una sola batería de este tipo llega a los 4.2v, dejando un total de 8,4v, más que suficiente y sin excederse del intervalo propuesto por el fabricante para alimentar por vía externa la placa Arduino, que sería de 6v hasta los 12v, superar estas indicaciones generarían picos que dañarían los componentes electrónicos alimentados.

### *4.2.7. Otros materiales*

Por último, tenemos los Dupont, cables puente macho-hembra, macho-macho y hembra macho, con medidas de 10cm. También las mini placas de pruebas, con 25 puntos de conexión, sin soldadura y dimensiones de 2,1 x 1,6cm. Además de conectores de barril para Arduino, tornillo autorroscante de acero inoxidable, M2 x 8, 2mm y, caja de almacenamiento de batería 18650, con cubierta y encendido/apagado.

*Ilustración 31. Caja porta baterías*

*Ilustración 32. Tornillos*

*Ilustración 34. Conector barril*

*Ilustración 33. Mini placa*

.

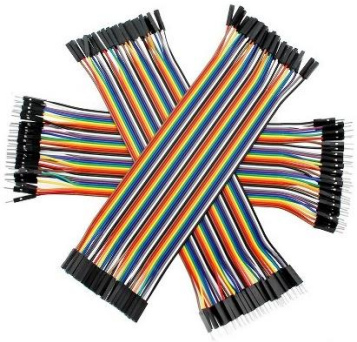


*Ilustración 35. Cables puente*

Estos últimos componentes se han usado para dar fijación a los diferentes componentes como servos en cuanto a los tornillos, las mini placas han sido usadas dos, que junto a los cables puente, han servido para poder puentear y dar más maniobrabilidad a los módulos y servos, y el conector barril ha sido usado para acoplar a la caja porta baterías y así poder conectarla a la placa Arduino Uno.

## 4.3. Esquema prototipo

Algunos de los componentes que se han visto en el anterior capítulo, tienen conexión con la placa Arduino, esto genera un gran cableado interno para todos los módulos y servos usados. A continuación, se intenta exponer detalladamente la disposición interna del cableado y así como una imagen esquemática, con el resultado final.

Para los servos, el cableado de toma a tierra (GND) y el de corriente (VCC), van conectados a las mini placas, ya que, por la falta de pines en Arduino, eran necesarios el juntar todos los cables que necesiten de corriente y de toma a tierra (GND). Dejando solo los pines de las señales para el uso de cada servo y, usando solo los pines que no son digitales. Empezando por el pieZ que se conecta a la señal 2, para rodZ usamos el 4, cadeZ el 7, pieD 8, rodD 12, cadeD 13.

Con los módulos pasa lo mismo para las conexiones de GND y VCC, por lo que solamente, se quedan los que van a las señales digitales, ya que para los módulos es necesario de estos pines para su correcto funcionamiento. Para el zumbador pasivo se ha usado el pin 6, el módulo ultrasónico, tiene como pines el trig y el echo, los cuales van conectados al 10 y al 11 respectivamente, por último, para la pantalla LCD necesitamos de dos conexiones, para los pines SCL y SDA, que van conectados al A4 para SDA y A5 para SCL.

Con esto tenemos los pines que hemos usado en la placa de Arduino Uno y que está conectado en cada uno de ellos, a continuación, se muestra la imagen del esquema final del prototipo

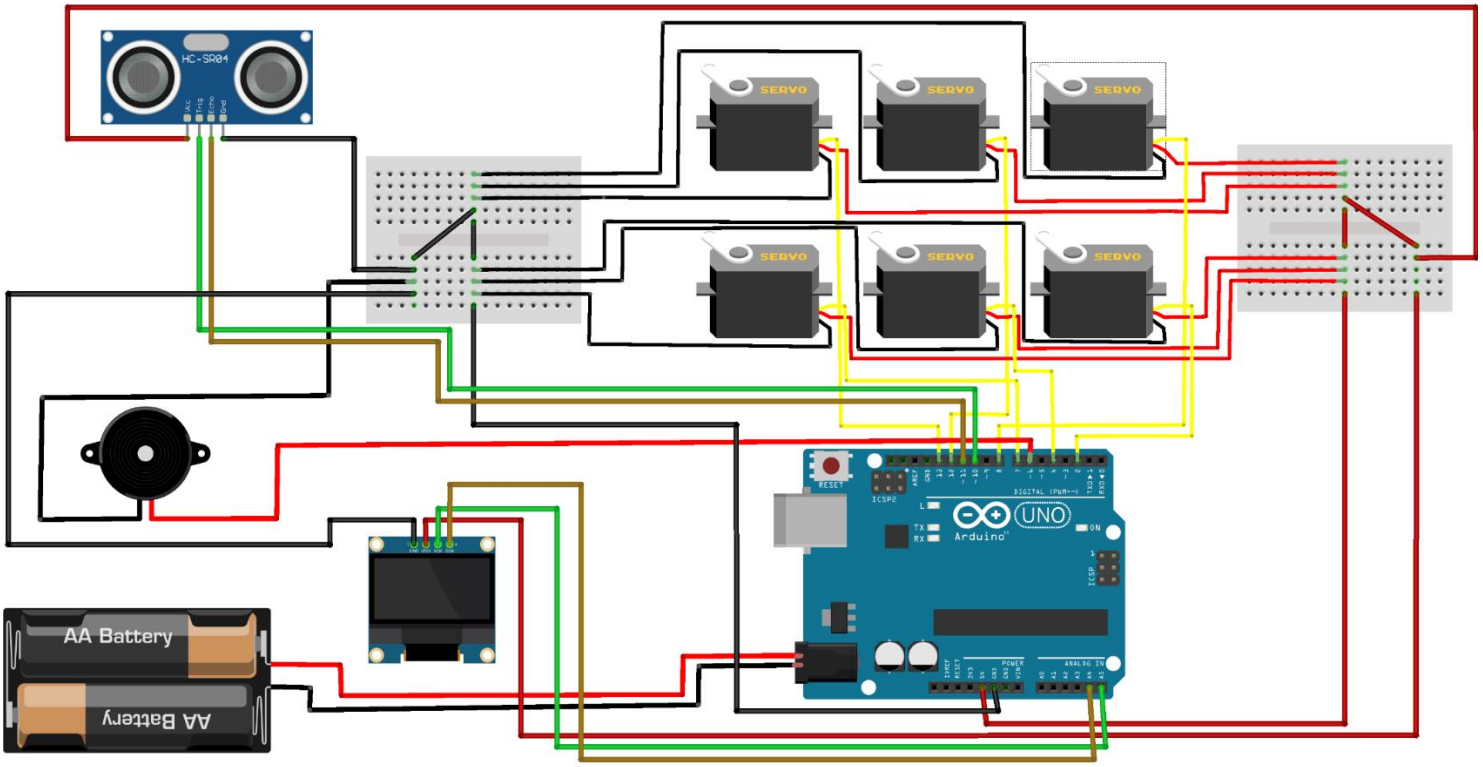



*Ilustración 36. Esquema prototipo*

## 4.4. Diagramas de piezas

Aquí se van a mostrar los planos de las piezas diseñadas en el uso del bípedo. Estas han sido elaboradas teniendo en cuenta las medidas de los servos y componentes usados, desde esa base se ha ido generando las formas que finalmente y, tras varias modificaciones conforme avanzaba en el prototipo, han ido dando la forma final del bípedo.

Las piezas que se han usado para este diseño han sido desarrolladas en FreeCAD, aplicación libre de diseño por ordenador 3D para la asistencia de elementos mecánicos en ingeniería mecánica.

### *4.4.1. SujeccionServoFinal*

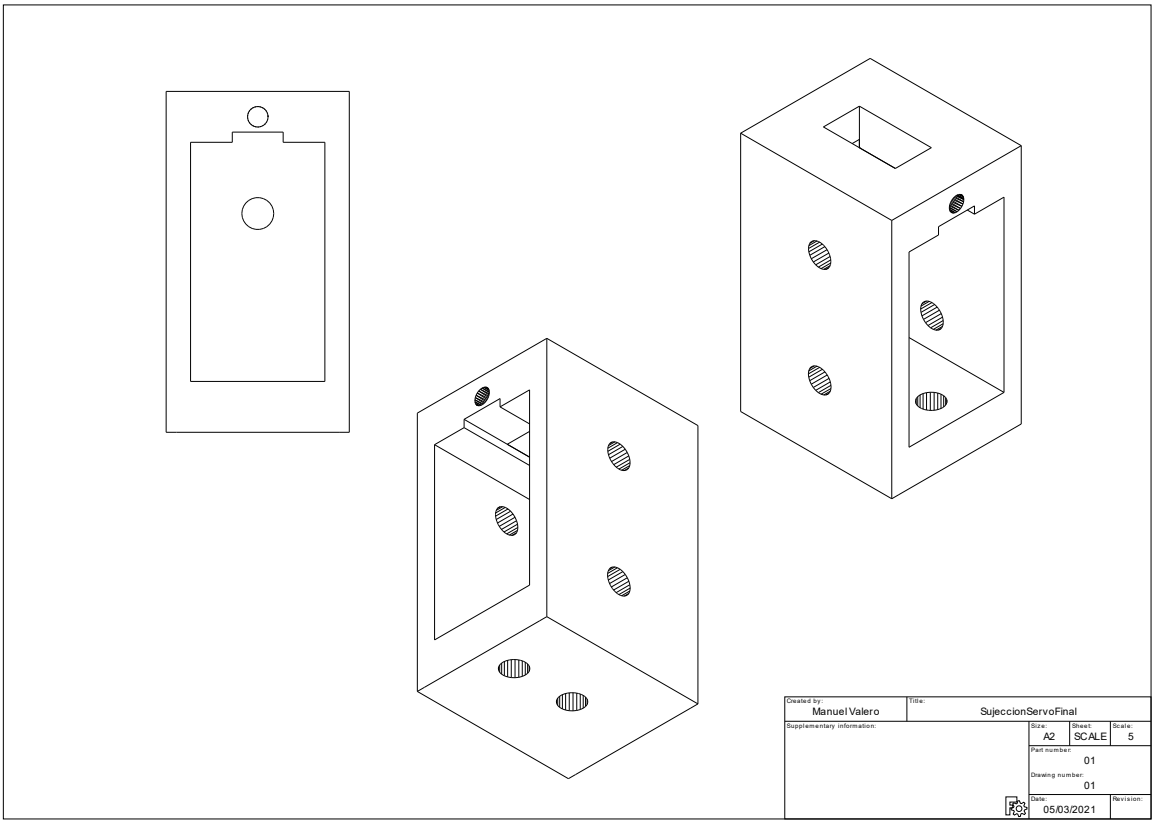



*Ilustración 37. SujeccionServoFinal*

Este diseño se ha conseguido con un modelo de servo MG90S con las medidas exactas a las del original usado en el TFG. Se ha usado un cuadrado modificando y ampliando sus medidas para obtener un rectángulo de 21mm de ancho x 33,5mm de alto x una profundidad de 21mm.

Posteriormente se ha vaciado parte de su interior, con las medidas un poco mayores a las del servo para no generar fricción y tener un acople lo más justo posible, este vacío tiene unas dimensiones de: 13,2mm de ancho x 23,5mm de alto x 18,6mm de profundidad, aparte, en una de su pared, se ha realizado una modificación con forma de rectángulo para el deslizamiento del cableado del servo, esta modificación tiene 5mm de ancho y ocupa toda la longitud de la pared. También en esta misma cara de la pieza, se ha realizado un vacío para sacar dichos cables al exterior de la pieza, este vacío tiene las dimensiones de 5mm de ancho x 10mm de altura.

Por último, he añadido varios orificios por los lados para poder modificar la posición de esta pieza en función de las posibles versiones que pudiera hacer del bípedo como se han descrito en los apartados anteriores, dependiendo si queríamos la versión de 4 servos por pata o la de 3 servos. Esto hace que tengamos unos orificios realizando un vacío en las caras laterales con diámetro de 3mm, a una distancia del borde de 7,6mm, y en la parte inferior otros dos orificios de las mismas medidas de su diámetro y a 2,4mm del borde el más próximo al mismo y 7,4mm el otro vacío.

### 4.4.2. PuenteVersionFinal

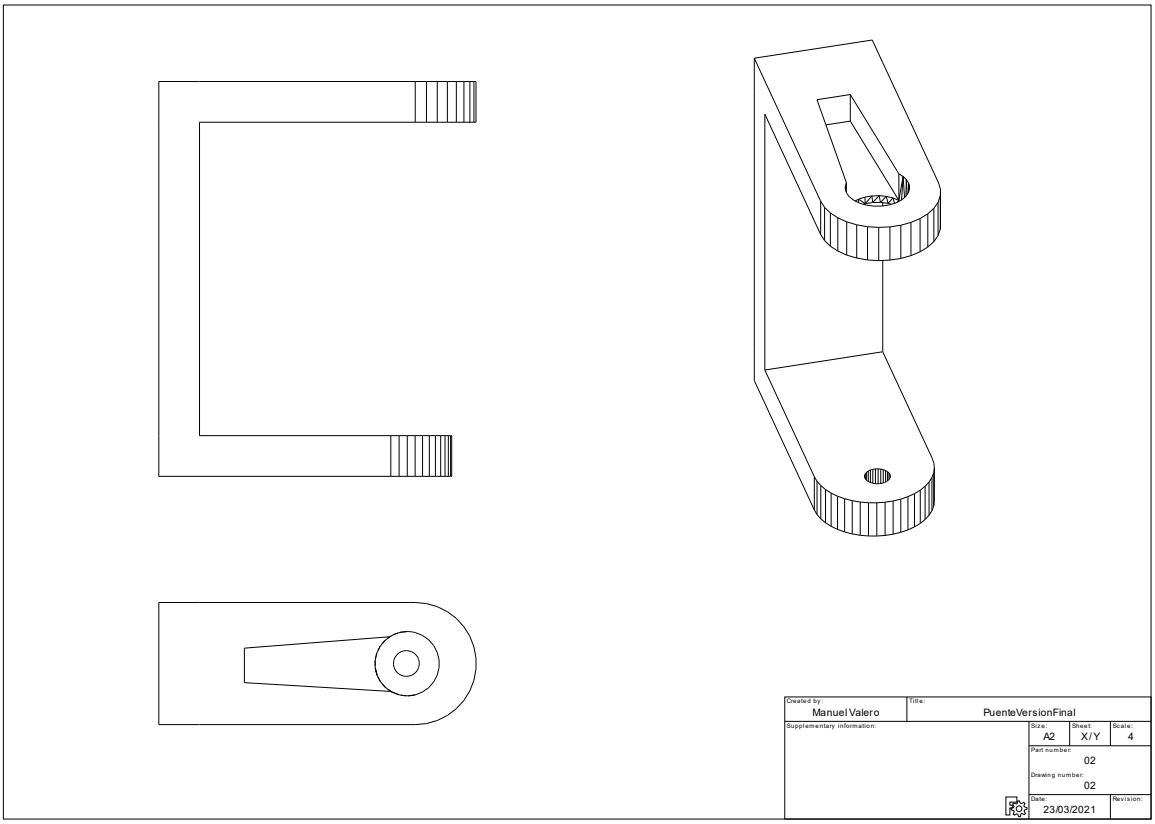



*Ilustración 38. PuenteVersionFinal*

La segunda pieza que diseñe, es la que extenderá el movimiento de rotación del servo, acoplando la hélice del mismo a uno de sus lados para una mayor fijación en el movimiento. Primero hice una pieza con forma rectangular de 48,5mm de ancho x 15mm de alto x 5 mm de grosor, está la cloné y situé en ambos lados adaptando la longitud deseada y, redondeando ambos extremos con forma de semicírculo.

Un lado tendrá un orificio que será el encargado de fijar esta pieza a la SujeccionServoFinal, con diámetro de 3mm, y la medición de este rectángulo es de 15mm de ancho x 35,9mm de alto.

Por último, el lado donde iría acoplada la hélice, guarda las mismas proporciones que el anterior salvo, que es ligeramente más largo, esto era necesario para poder realizar una buena fijación con el servo y su hélice. Como se ha nombrado

anteriormente, teníamos las medidas del servo y como es normal también de sus componentes, por lo que lo primero fue realizar un vacío que permitiera ajustar el engranaje del servo junto a la hélice, teniendo un vacío circular de 7,9mm de diámetro, posteriormente necesitamos ajustar la longitud de la hélice a la superficie con un vacío de 4mm para no sobrepasar por completo la base y conseguir una mayor fijación. Para las últimas medidas tenemos 6,7mm la parte más ancha de la hélice y donde comienza reducir su tamaño mientras que se alarga hasta unos 16mm de longitud y 4,2mm de ancho para este extremo. Con esto se consigue una fijación de la hélice lo más ajustada posible.

### *4.4.3. PieIzquierdoV2 y PieDerechoV2*

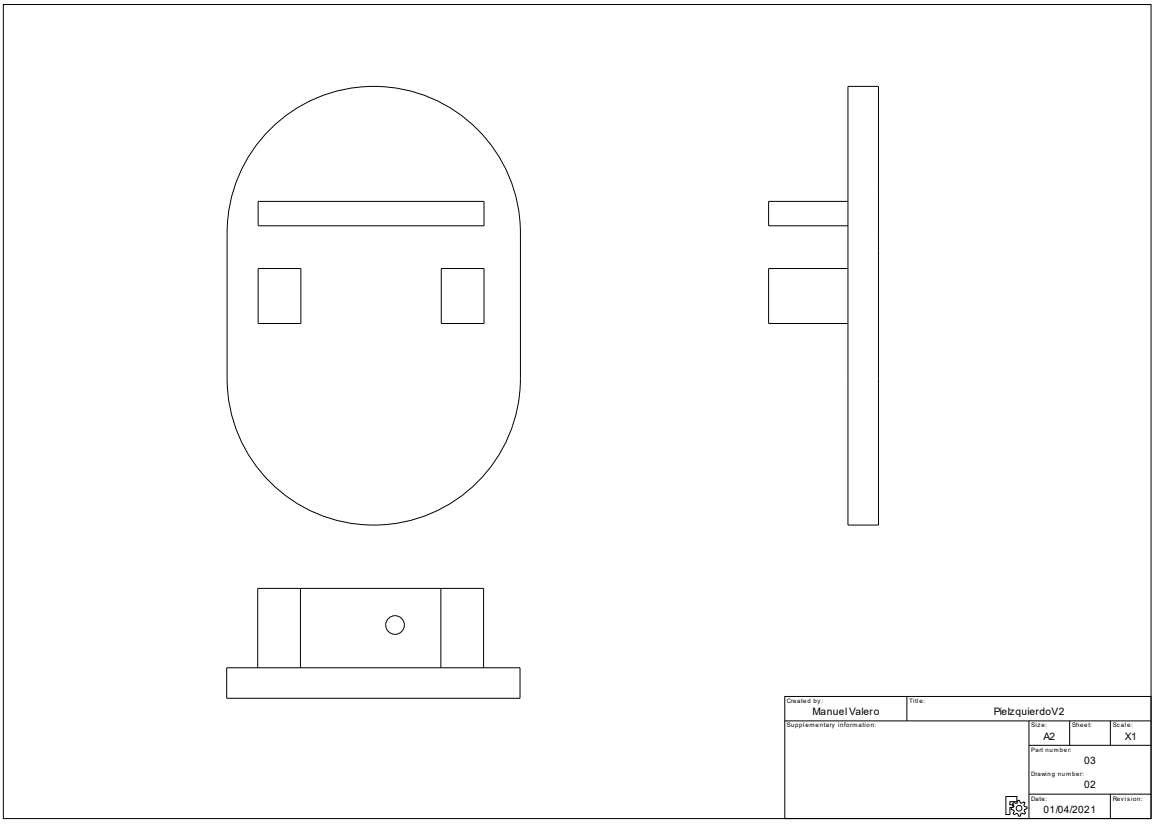



***Ilustración 39. PieIzquierdoV2***

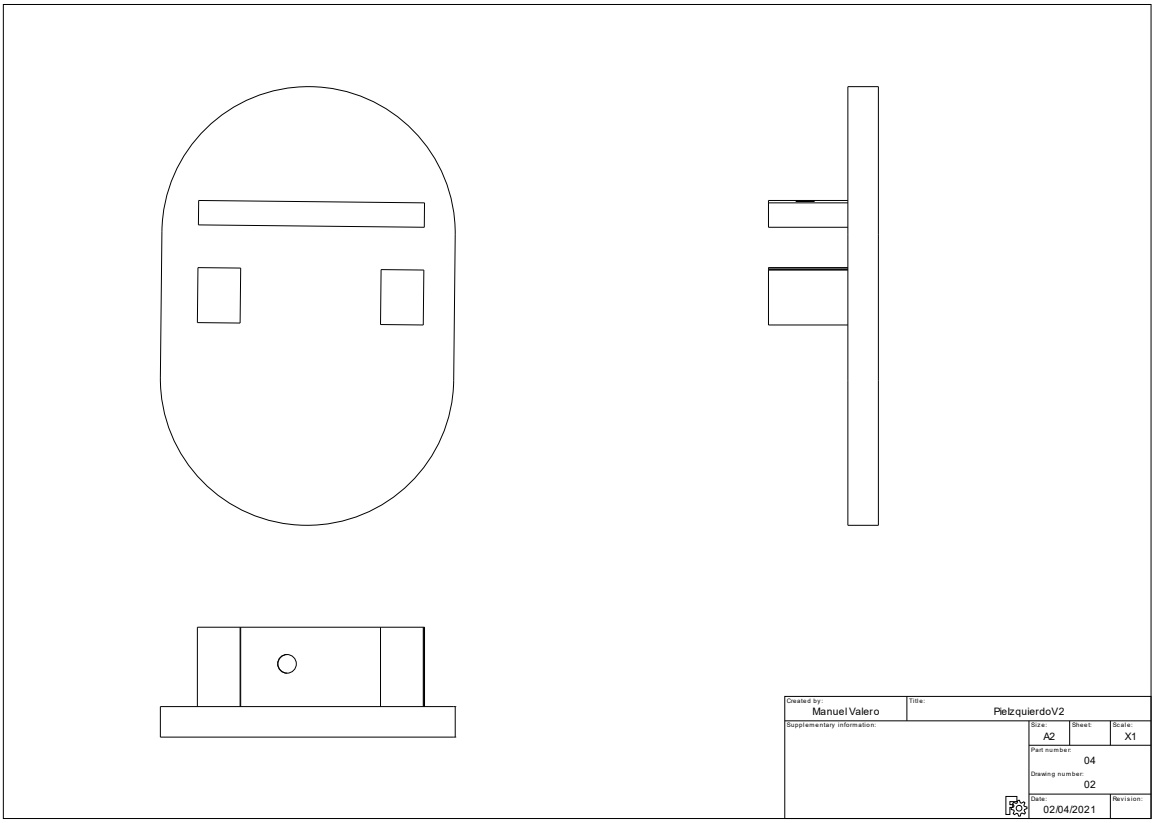



***Ilustración 40. PieDerechoV2***

El pie, empezó siendo un rectángulo de 48mm de ancho x 71,4mm de alto x 4 mm de grosor, con 3 piezas fusionadas para el acople y fijación del servo que, junto a el PuenteVersionFinal, hacen el movimiento del tobillo interno y externo. Estas piezas tenemos dos con medidas de 7mm de ancho x 8,9mm de largo x 13mm de alto, situadas ambas a 5mm de su lado lateral más próximo y a 34mm del opuesto y, a 29,7mm del lado trasero y 55,4mm frontal. Estas piezas tienen como función ser de base para el tornillo que usaremos para fijar el servo al pie.

Para la última pieza y que su función es la de fijar el PuenteVersionFinal y además de hacer de tope para el servo, con unas medidas de 37mm de ancho x 4 de alto x 13mm de alto. Esta pieza esta fusionada en la base a 5mm de ambos lados, a 7mm de las dos piezas anteriores descritas y a 18,7mm de la parte trasera de la base. Aparte tiene un vacío circular de 3mm de diámetro para el tornillo que fijara el puenteVersionFinal al pie y, dependiendo de si es el pie izquierdo o derecho, su

disposición cambiara para obtener una postura del conjunto de la pierna final más natural, dejando dicho vacío a 21mm de su lado más alejado y 13mm del más cercano.

Para el diseño de ambos pies solo fue necesario el realizar uno de ellos y con el modo espejo que disponemos dentro de la aplicación FreeCAD, obtenemos el PieIzquierdoV2 y el PieDerechoV2.

### *4.4.4. BaseYcaraInferior*

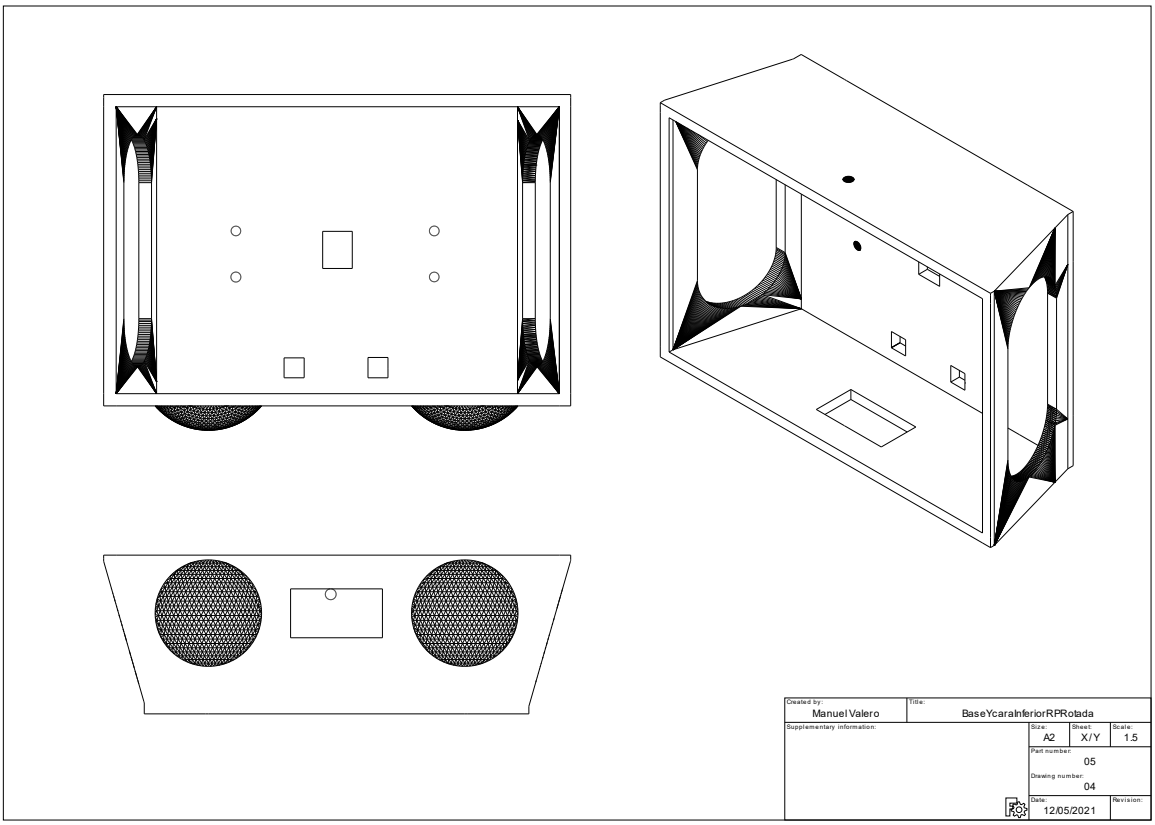



*Ilustración 41. BaseYcaraInferior*

El desarrollo fue bastante tedioso, pues requería de más técnicas que las anteriores piezas, además de ser la parte más importante del diseño ya que todo partiría de la cabeza. En la idea principal para la cara, decidí hacer una separación, dejando parte superior e inferior de la cara y, así poder manipular mejor los

componentes que irían en su interior. Teniendo claro como realizaría las manipulaciones para las pruebas del código, comencé por el desarrollo del diseño desde la base, en ella, tuve en cuenta las estructuras que habría para la sujeción de las otras partes y los componentes. Sabiendo esas medidas, dibuje una base para enfocarme y poder añadir las correspondientes manipulaciones.

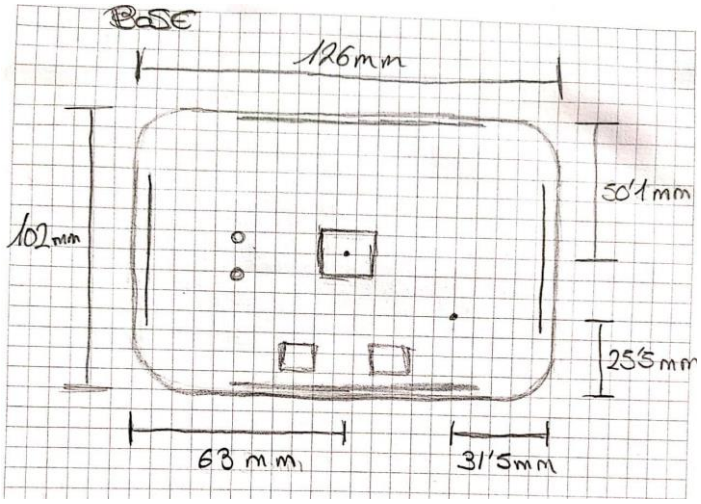



***Ilustración 42. Boceto base***

Como se aprecia en la Figura 19, era necesario hacer esas modificaciones para la sujeción del porta baterías, la pantalla y la sujeción de los servos, para estos últimos, teníamos la pieza hecha por lo que, fue solo cuadrar más o menos y realizar los vacíos de 3,3mm en la mitad más próxima al alto de la base y, a la mitad de la mitad del ancho, para así poder tener una mayor estabilidad cuando se encuentre erguido y sin movimiento.

Lo siguiente era acoplar la pantalla LCD, para esto al igual que la sujeción de los servos, teníamos la pieza, por lo que tomando las medidas se procedió al vacío para el acoplamiento. 2 cuadrados de 6,55mm de ancho y alto.

Por último, solo quedaba realizar el vacío para acoplar la pieza que sujetaría la porta baterías, como las anteriores modificaciones, con la pieza ya diseñada, fue adaptarla a este diseño y generar un vacío de 9,7mm de ancho x 12,2mm de alto.

Con esto teníamos la base con los acoples de los componentes terminada, el siguiente paso era darle altura, lo que me genero el primer problema con el diseño que había hecho, no alcanzaba a hacer esos bordes redondeados para continuar la altura con la forma de la base. La solución que encontré fue quitar el redondeo de la base y así facilitar el diseño de la altura, una vez solucionado el problema, procedí a darle la altura. Mientras seguía con el diseño se me ocurrió el adaptarlo a la cara de una rana, por lo que comencé a darle la altura y a su vez darle inclinación hacia fuera por los lados para simular dicha forma de rana, la altura que alcanzo fue de 52mm x una anchura de 153mm, lo que suponía una diferencia de 27mm con respecto a la base 126mm.

Teniendo prácticamente la base y mitad de la cabeza terminada, solo quedaba darle visibilidad a la pantalla, con su respectiva pieza, teníamos la altura que tendría, por lo que como anteriormente hemos hecho, necesitábamos generar el vacío en la parte frontal. Este vacío tenía las medidas de 16mm de alto x 30mm de ancho en la parte más céntrica de la cara frontal. También en dicha parte frontal decidí añadir dos semiesferas de 40mm de diámetro, que simularan unos mofletes para que así transmitiera algo más al verlo. Posteriormente, en la parte trasera de la cara, se añadió un vacío de 3,3mm de diámetro para el tornillo que usaríamos para fijar tanto la parte superior de la cabeza como la inferior.

Por último, decidí realizar unos vacíos en ambas partes laterales, teniendo las medidas de 32,9mm de alto x 76mm de ancho, con esto podría manipular el encendido y apagado del prototipo y, además, reducir en la mayor medida posible el peso de la cabeza.

### 4.4.5. CaraSuperiorV3.1

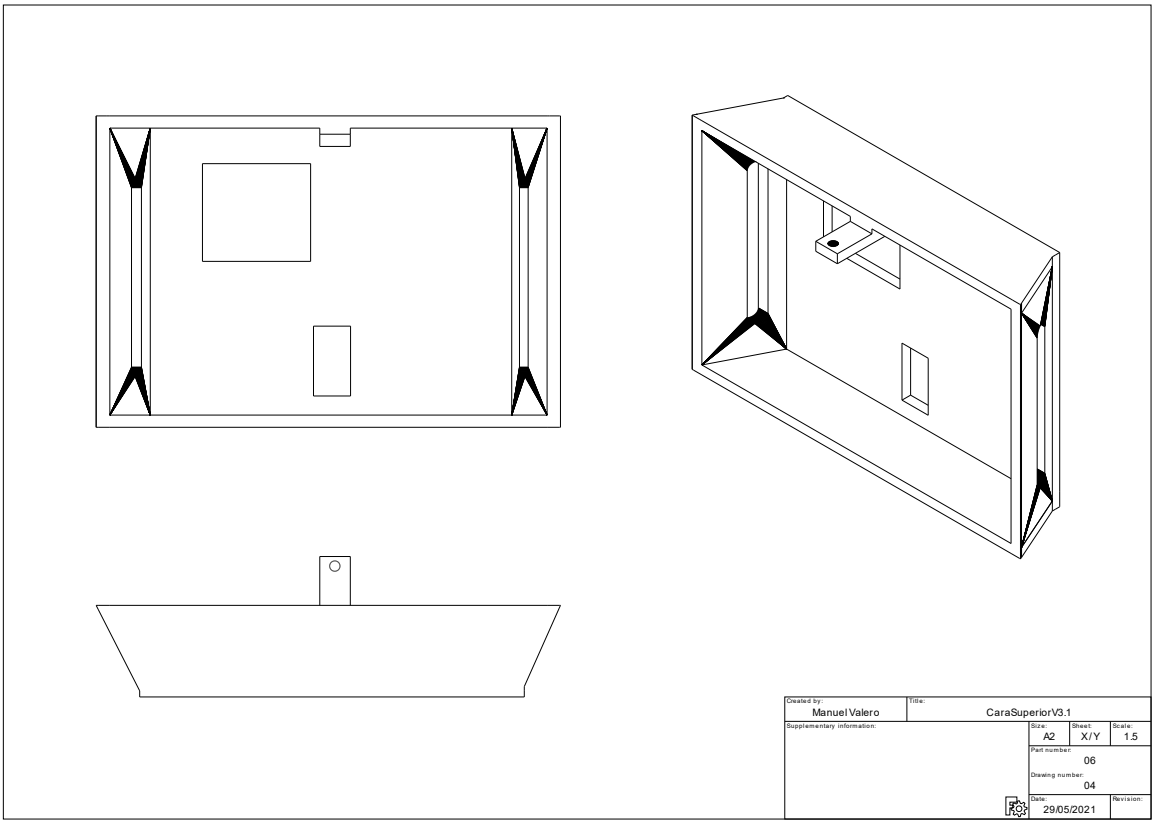



*Ilustración 43. CaraSuperiorV3.1*

Para la parte superior, se tuvieron en cuenta las medidas de la base, por lo que la base de esta parte era de las mismas medias. Para las modificaciones, empecé por un cuadrado en la parte superior, pensado para facilitar la manipulación de los componentes cuando estuviera cerrado el robot, este vacío tiene 35,4mm de ancho x 32mm de alto.

El siguiente vacío está situado en el centro más próximo a la parte frontal de la cara, ya que tiene como función la sujeción del sensor ultrasónico, así evitamos la captación errónea de datos si lo colocáramos más próximo al centro de la base. Este vacío tiene como medidas por la parte ancha de 12,1mm x 22,9mm de alto.

Las dos siguientes modificaciones son una para la fijación y seguridad del cierre entre la parte superior e inferior de la cabeza, y como en la parte inferior,

modificaciones para reducir el peso. La primera consta del acoplamiento de una pieza rectangular fusionada en la parte trasera y con un orificio para roscar un tornillo y así fijar tanto la parte inferior como superior, las medidas son: 10mm de ancho x 26mm de alto x 4mm de grosor. En cuanto a los vacíos laterales para reducir el peso, tiene unas medidas de 7.15mm de alto x 65,1 de ancho.

### *4.4.6. BaseBoardV2.1*

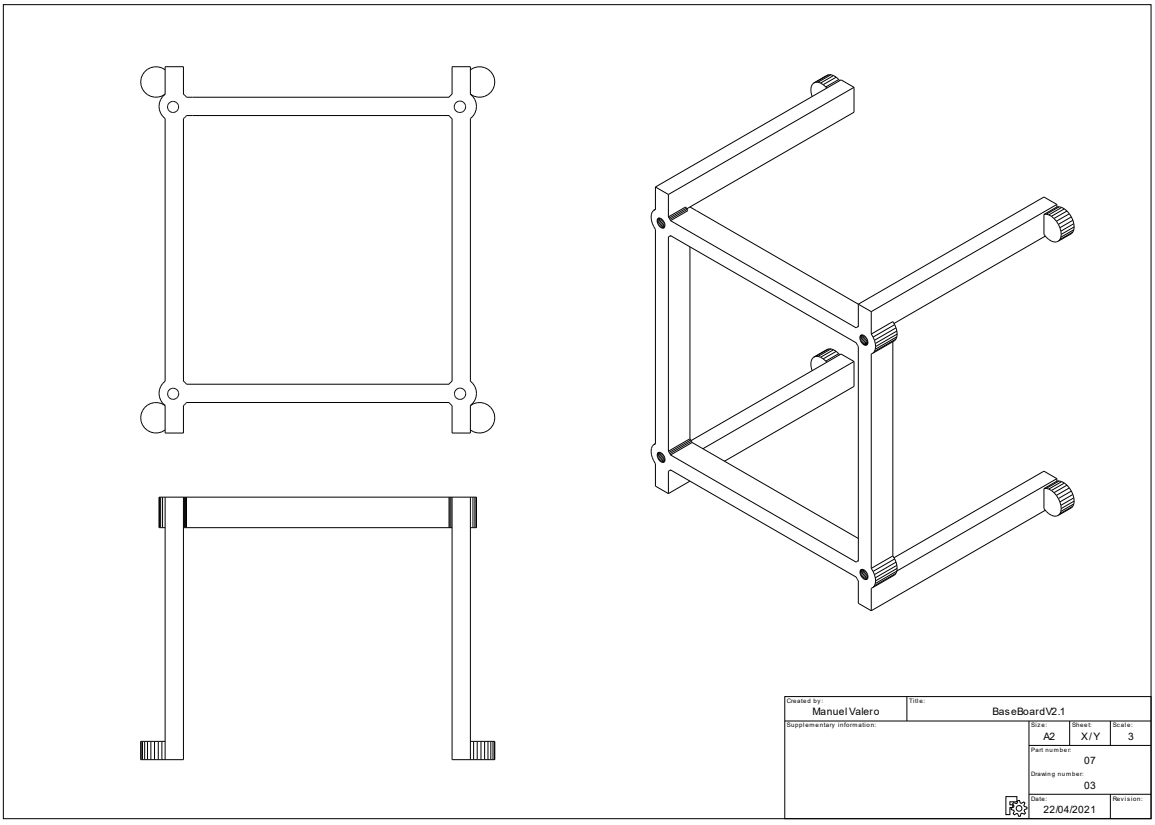



*Ilustración 44. BaseBoardV2.1*

Con el diseño de la base para la placa, primero se tomaron las medias de los agujeros para los tornillos que trae por defecto Arduino, sabiendo la colocación de dichos agujeros, comencé a darle forma de rectángulo a cada parte y fusionando las piezas para formar un cuadrado, donde en el punto que se unen entre sí, añadir un cilindro y así tener más base para el posterior vacío donde irían los tornillos que figaran la placa a esta pieza.

Para los tornillos el vacío que se genera sería de 3,3mm, las medidas de este diseño se quedarían con 60mm el ancho x 50mm de largo x 43mm de altura. Por último, añadí unos cilindros a la base de las patas para una vez este colocado, proceder a marcar y realizar el atornillado a la base de esta pieza.

### *4.4.7. BaseBateria*

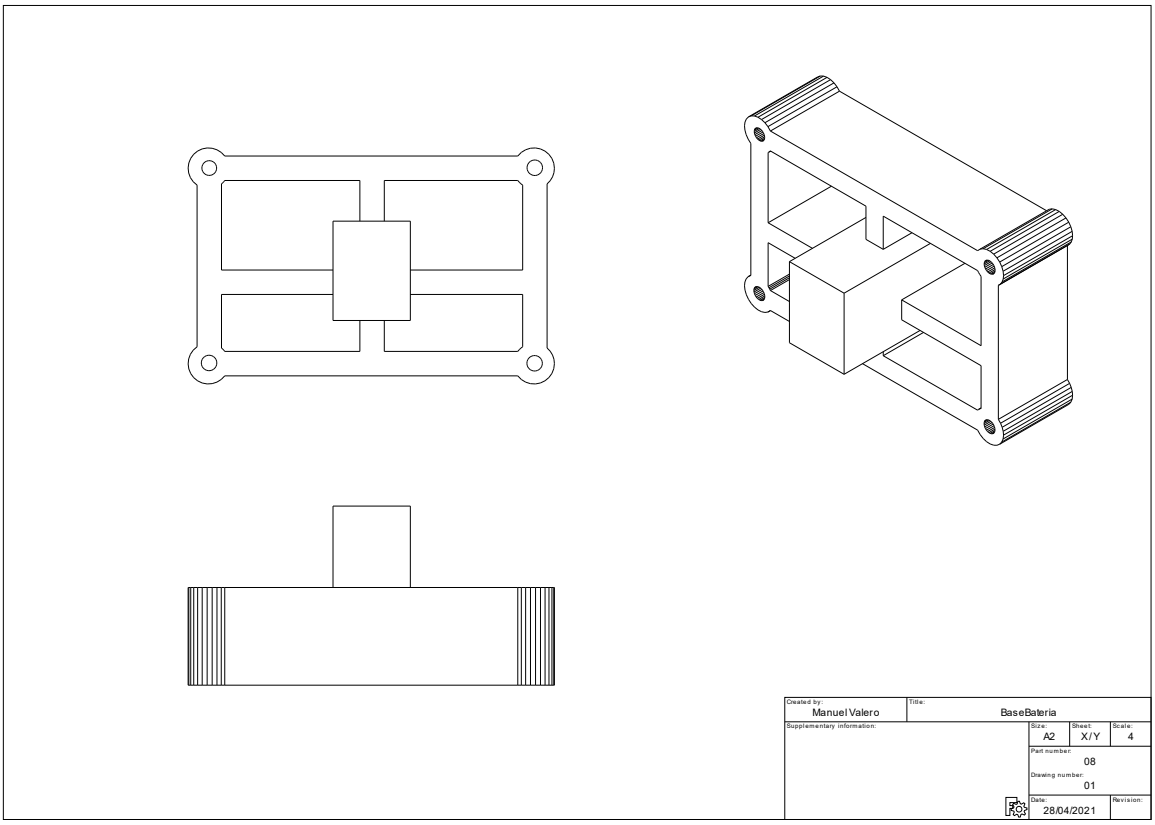



***Ilustración 45. BaseBateria***

Para la base de la batería, tuve en cuenta las medidas de los tornillos de la caja porta baterías, aparte de tener en cuenta las medidas que ya se mencionaron en la base para el vacío donde iría este diseño, que fueron de 9,7mm de ancho x 12,2mm de alto, también había que considerar que, si optábamos por las mismas medidas, se produciría demasiada fricción e incluso no poder acoplar esta pieza a su correspondiente hueco en la base. Por lo que hay una diferencia de 1mm aproximadamente entre ambas medidas, quedando tal que así: 8,5mm para la parte

ancha x 11mm para la parte alta y, añadiendo la profundidad de 22mm, para obtener así superficie para poder fusionarla con la base donde irían los tornillos fijadores.

Las siguientes medidas fueron para la sujeción de los tornillos, donde al igual que con la base de la placa, se fundieron varios rectángulos de 4mm de grosor y, en el punto de unión se le ha añadido un cilindro para el diámetro del vacío que se haría de 3,3mm para los tornillos, haciendo que la medida final de esta parte sea de 43mm para la parte ancha x 27mm de la parte alta.

Después hacía falta unir ambas partes, tanto la base para la caja porta baterías como la base donde se ajusta con la cabeza, esto se consiguió duplicando uno de los rectángulos usado para la parte ancha de la base y otro para la parte alta de la misma, desplazándolos y fusionándolos con la base central y reforzando la estructura del diseño formando una cruz central.

### 4.4.8. BaseFijacionPantalla

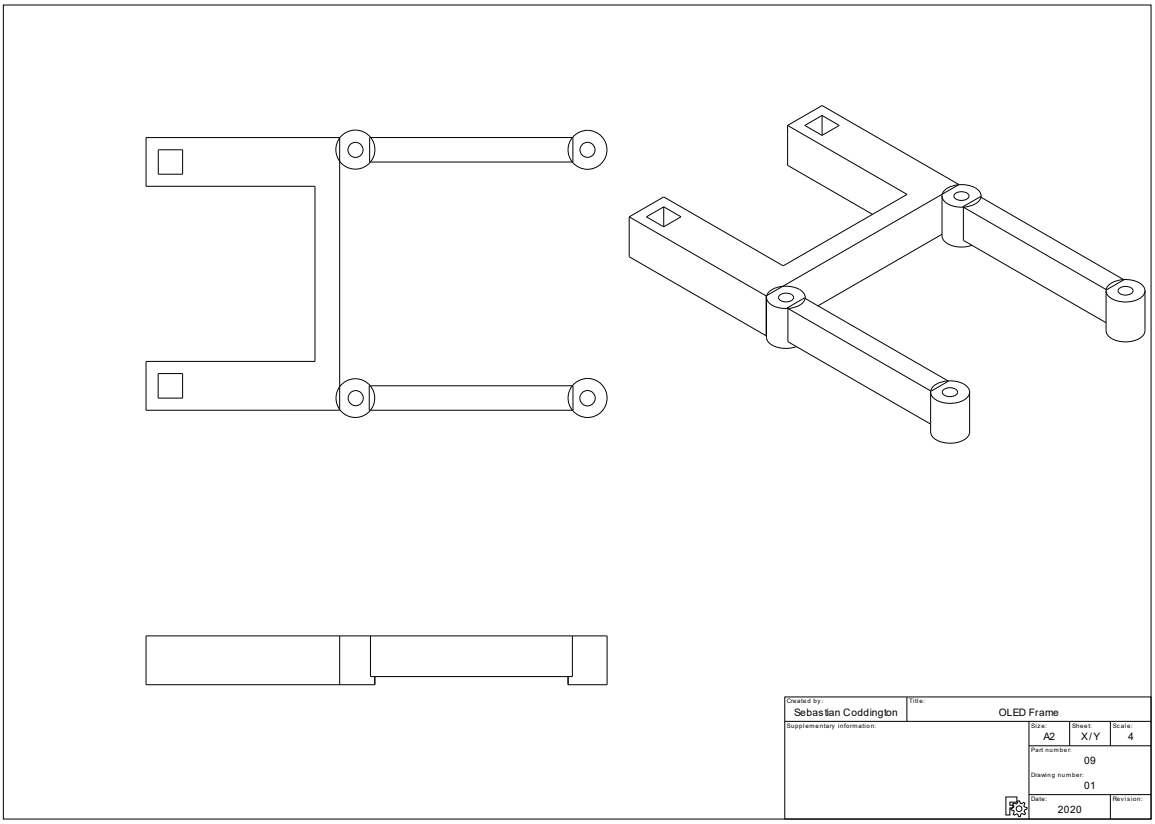



*Ilustración 46. BaseFijacionPantalla*

Base fijadora de la pantalla LCD, esta pieza la cual fija la pantalla y se acopla a la base de la cabeza del robot, ha sido diseñada por el autor del proyecto Boris, que anteriormente mencione como la guía e inspiración para mi prototipo. La idea de incluir una pantalla en la cara que pudiera mostrar diferentes estados, me gusto, por lo que la copie y adapte a mi prototipo de robot.

Las medidas de la base fijadora de la pantalla son: 33,2mm de ancho x 56,5mm de alto x 5,5mm de grosor, esta última medida, es la que hace el acople con la base de la cabeza, la cual, anteriormente se nombraron las modificaciones para esta pieza en la base. También tenemos unas medidas de 2mm de grosor para los tornillos fijadores de la pantalla y un vacío con forma de cubo que en el proyecto Boris usa para introducir unos topes y así sujetar esta pieza firmemente a la base. En mi

prototipo esta función no ha sido usada, requería la impresión de ese diseño de pasador de bloqueo y, sería otro diseño más del proyecto Boris, por lo que opte en aproximar las medidas del grosor con el vacío que se había generado en la base y así hacer una fricción que me sirviera de fijación.

### *4.4.9. SujeccionModuloApoximidad*

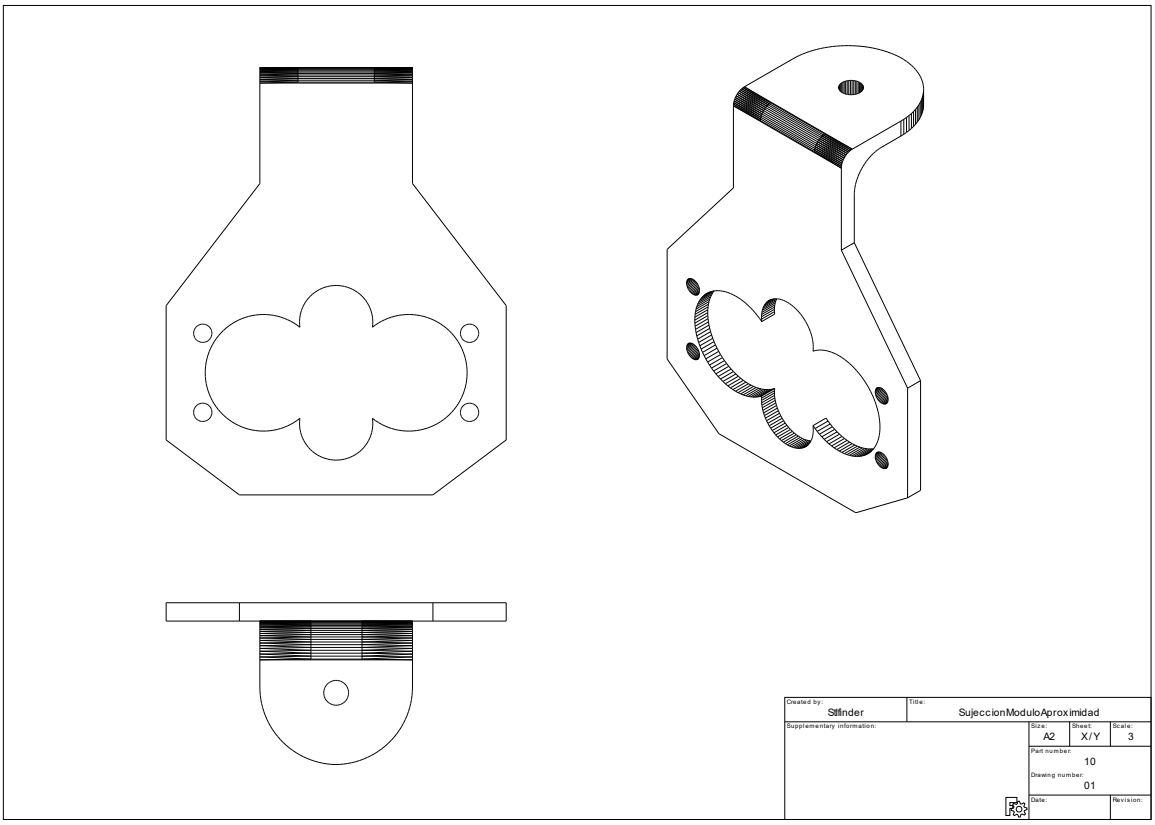



*Ilustración 47. SujeccionModuloApoximidad*

Al igual que la pieza BaseFijacionPantalla, este diseño también ha sido obtenido de otro usuario, en este caso de una web donde puedes realizar una búsqueda de diseños 3D hechos por la comunidad. Al utilizar un módulo que es de los más usados en los proyectos de Arduino, el sensor ultrasónico, era de esperar que hubiera un diseño de esta pieza que encajara con mi prototipo, teniendo en cuenta

que mi principal idea era colocar dicho sensor en el engranaje de un servo, el único requisito que debía tener era el disponer de una base para, en el caso de no tener un orificio ya generado, realizarlo nosotros mismos para el acople con el engranaje del servo.

El diseño encontrado constaba con lo necesario para poder incorporarlo sin modificaciones en el prototipo que estamos realizando. Dicha pieza tiene unas medidas de: 55,5mm de ancho x 73mm de alto x un grosor de 3mm, aparte tiene una base de unos 26,5mm x 24,5mm, donde en la parte central tenemos un vacío con forma de circulo con diámetro de 4mm, este tiene la función de ser acoplado al engranaje del servo y así proporcionar la fijación de la pieza.

Con los diseños anteriores, tenemos el esqueleto de nuestro robot bípedo. Estas piezas han pasado por diversas modificaciones y versiones conforme avanzaba el prototipo, ya que se iban generando problemas y superposiciones de algunas piezas que impedían el correcto recorrido de las mismas.

## 4.5. Implementación

En este apartado se describirá con un mayor grado de detalle el código empleado en este proyecto, además de nombrar y explicar los principales métodos de programación del prototipo, utilizando el lenguaje de programación C++ y desarrollado en el IDE Arduino.

### *4.5.1. Código del prototipo*

En primer lugar, tenemos las librerías y definición de los servos que usaremos, para las librerías tenemos 4, siendo “servo.h” para poder definir y controlar los servos que hemos usado, “wire.h” para las comunicaciones con dispositivos por el bus I2C y necesario para el funcionamiento de los módulos que hemos usado en el prototipo, “Adafruit_GFX.h” es la librería que nos permite el uso de pantalla y

"Adafruit_SH1106.h" la librería específica para nuestra pantalla. En cuanto a los servos, se han nombrado de la manera más descriptiva posible para facilitar en el desarrollo de las funciones del movimiento, el reconocimiento del servo que moveríamos.

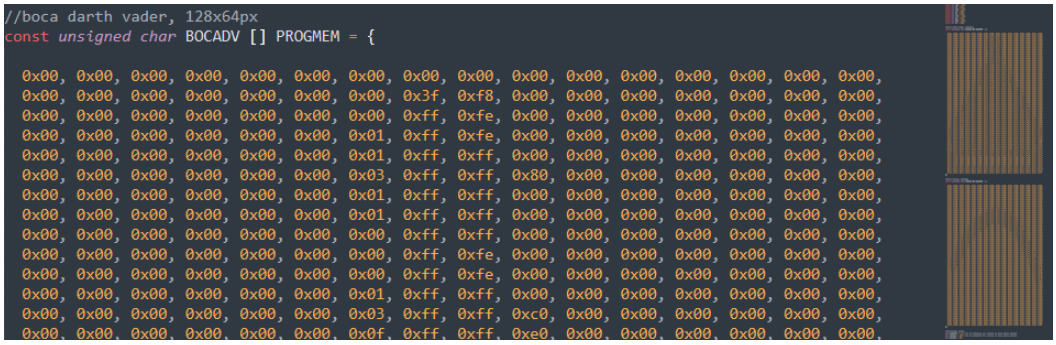
```
#include <Servo.h>            //Libreria para los servos
#include <Wire.h>             //libreria para bus I2C
#include <Adafruit_GFX.h>     //libreria para pantalla
#include <Adafruit_SH1106.h>  //libreria para controlador SSH1106

//Servos que usaremos
Servo    pieD;
Servo    rodD;
Servo    cadeD;

Servo    pieZ;
Servo    rodZ;
Servo    cadeZ;
```

*Ilustración 48. CÓDIGO, definición servos y librerías*

A continuación, tenemos las variables y pines que hemos declarado para los diferentes módulos. Donde tenemos la variable tiempo y distancia, que usaremos para obtener el cálculo necesario para el sensor ultrasónico, junto a los pines del mismo, también, tenemos declaradas otras variables necesarias tanto para el módulo de la pantalla, como para el módulo zumbador.

```
//Sensor ultrasonico
long tiempo;
int trig = 10;
int echo = 9;
float distancia;

//Necesario para la pantalla
#define OLED_RESET 4

//Objeto de la clase Adafruit_SH1106 para la pnatalla
Adafruit_SH1106 pantalla(OLED_RESET);

//Pin zumbador
int buzzer = 6;

//Notas para buzzer
const int c = 261;
const int d = 294;
const int e = 329;
const int f = 349;
const int g = 391;
```

*Ilustración 49. CÓDIGO, variables sensores y módulos*



```
//boca darth vader, 128x64px
const unsigned char BOCADV [] PROGMEM = {

  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

*Ilustración 50. CÓDIGO, matriz de bytes para el LCD*

En esta última imagen, se puede ver la matriz de bytes generada mediante la herramienta image2cpp[5] y, una abreviatura del diseño generado en esta matriz, con las distintas imágenes que se usan para un más amigable diseñó del prototipo.

Posteriormente, tenemos declaradas las posiciones que tendrán los servos, para tener el prototipo en un estado erguido y mantener el equilibrio. Además, se declara la primera función, cuya principal finalidad es mantener el equilibrio en estático y, dos variables que se usarán para generar un retardo.

[5] Image2cpp, herramienta para convertir imágenes en matrices de bytes, http://javl.github.io/image2cpp/

```
//Posiciones servos
int cadeZPos = 77; //si se disminuye los valores se mueve hacia delante
int rodZPos = 107; //si se aumentann los valores se mueve hacia delante
int pieZPos = 95;

int cadeDPos = 77; //si se aumentan los valores se mueve hacia delante
int rodDPos = 80;  //si se disminuyen los valores se mueve hacia delante
int pieDPos = 70;

//Diferentes retardos
int forDelay = 40;
int delayUno = 30;

//Posicion estable
void posicionInicial(){

//Posicion inicial de los servos
   cadeZ.write(cadeZPos);
   rodZ.write(rodZPos);
   pieZ.write(pieZPos);

   cadeD.write(cadeDPos);
   rodD.write(rodDPos);
   pieD.write(pieDPos);

   delay(1000);
}
```

*Ilustración 51. CÓDIGO, ángulos servos y función para estabilidad*

Aquí tenemos la función principal, el movimiento hacia el frente, esta función realiza el movimiento de la marcha bípeda, separado en diferentes bucles for, que generan un movimiento un acotamiento y movimiento más suave de los servos. El recorrido por toda la función empezaría con el movimiento de ambas rodillas y caderas, haciendo que cuando la pierna derecha realice el movimiento de su rodilla hacia atrás y su cadera hacia delante, la pierna izquierda lo genera a la inversa, haciendo que su rodilla izquierda se mueva hacia delante y su cadera hacia atrás, tal y como se muestra en la siguiente imagen.

*Ilustración 52. Disposición piernas primer movimiento*

El siguiente bucle hace que el pie izquierdo realice un ligero movimiento hacia dentro y el derecho hacía fuera. Después tendríamos el tercer bucle for, el cual, realiza el movimiento del primer bucle, pero a la inversa, haciendo que la rodilla derecha vaya hacia delante y su cadera hacia atrás y la rodilla izquierda hacia detrás y la cadera hacia delante. Por último, los movimientos de los pies al contrario que en el segundo bucle, moviendo el pie derecho hacia dentro y el izquierdo hacia fuera.

```
//Función para mover hacía el frente
void adelante(){

  //Mueve la rodilla der hacia atras y la cadera hacia delante
  for (int i = 0; i < 20; i++){
    cadeD.write((cadeDPos-10)+i);
    rodD.write((rodDPos-10)+i);

    //Mueve la rodilla izq hacia delante y la cadera hacia atras
    cadeZ.write((cadeZPos-10)+i);
    rodZ.write((rodZPos-10)+i);
    delay(forDelay);
  }

  delay(delayUno);

 //Pie izquierdo hacia dentro y pie derecho hacia fuera
  for (int i = 0; i <20; i++){
    pieZ.write((pieZPos-10)+i);
    pieD.write((pieDPos-10)+i);
    delay(forDelay);
  }

  delay(delayUno);
  //rodilla izquierda hacia detras y cadera hacia delante
  for (int i = 0; i < 20; i++){
    cadeZ.write((cadeZPos+10)-i);
    rodZ.write((rodZPos+10)-i);

    //rodilla derecha hacia delante y cadera hacia atras
    cadeD.write((cadeDPos+10)-i);
    rodD.write((rodDPos+10)-i);

    delay(forDelay);
  }

  delay(delayUno);

  for (int i = 0; i <20; i++){
    pieD.write((pieDPos+10)-i);   //pie derecho hacia dentro
    pieZ.write((pieZPos+10)-i);   //pie izquierdo hacia fuera

    delay(forDelay);
  }

  delay(delayUno);

}
```

***Ilustración 53. CÓDIGO, función deslazarse hacia el frente***

La siguiente parte del código, es la que genera el movimiento hacia atrás del bípedo, al igual que la marcha hacia delante, esta función también está desarrollada en 4 bucles for los cuales realizan el movimiento necesario para dar los pasos hacia atrás, manteniendo la estabilidad del bípedo. Esto lo conseguimos modificando la posición de los pies al realizar el movimiento con las caderas y rodillas, haciendo que el peso de la pierna recaiga sobre la que está más atrasada de las dos y así conseguir el movimiento hacia atrás.

```
//Pie izquierdo hacia fuera y pie derecho hacia dentro
for (int i = 0; i < 15; i++){    //probando posiciones
  pieZ.write((pieZPos)-i);       //pie izquierdo 15
  pieD.write((pieDPos+10)-i);    //pie derecho 15
  delay(forDelay);
}
for(int i = 0; i <20; i++){
  pieD.write((pieDPos-10)+i);   //pie derecho hacia fuera
  pieZ.write((pieZPos-10)+i);   //pie izquierdo hacia dentro

  delay(forDelay);
}

delay(delayUno);
```

*Ilustración 54. CÓDIGO, parte función desplazamiento hacia atrás*

Los últimos movimientos que hemos desarrollado, han sido el giro a la derecha y el giro a la izquierda. Esto lo hemos conseguido moviendo la cadera y rodilla como si iniciáramos la marcha hacia delante, anteriormente nombrada, pero solo el generado por un bucle for que, en este caso del giro a la derecha, el movimiento sería el de cadera izquierda hacia atrás y rodilla izquierda hacia delante y, cadera derecha hacia delante y rodilla derecha hacia atrás. Para el giro a la izquierda, se repetiría la misma secuencia, pero, al contrario.

```
//Función para girar a la derecha
void girarDerecha(){
  for (int i = 0; i < 20; i++){
    //Cadera derecha hacia atras y rodilla hacia delante
    cadeD.write(cadeDPos-i);
    rodD.write((rodDPos-5)-i);

    //Cadera izquierda hacia delante y rodilla hacia atras
    cadeZ.write(cadeZPos-i);
    rodZ.write((rodZPos-5)-i);

    delay(100);
  }
}

//Función para girar a la izquierda
void girarIzquierda(){
  for (int i = 0; i < 20; i++){
    //Cadera izquierda hacia atras y rodilla hacia delante
    cadeZ.write(cadeZPos+i);
    rodZ.write((rodZPos+5)+i);
    //Cadera derecha hacia delante y rodilla hacia atras
    cadeD.write(cadeDPos+i);
    rodD.write((rodDPos+10)+i);

    delay(100);
  }
}
```

*Ilustración 55. CÓDIGO, funciones que giran hacia izquierda y derecha*

Posteriormente, nos encontramos con las funciones que realizan el sonido de la melodía inicial, así como la de toparse con un obstáculo. Esto se ha conseguido con varias funciones, la más importante, es la que usamos para hacer llamada a la función tone()[6], que crea un objeto llamando al pin donde está el zumbador, la frecuencia y la duración, después, se han declarado dos funciones para la reproducción de sonido por el zumbador, que en este prototipo y por el color del diseño y junto a la imagen de la pantalla, se ha usado parte de la melodía de Start Wars [13] que, se reproduce al encender el prototipo, y por último, tenemos el sonido triste que es el que se reproduce cuando el sensor ultrasónico se encuentra con un obstáculo.

```
//Función para las notas de la melodia
void beep(int nota, int duracion){

    //Iniciar tono en el buzzer
    tone(buzzer, nota, duracion);

    //Retardo para no acoplar las notas
    delay(duracion);
}

//Parar la melodia
void pararMelodia(){ ...
}

//Notas para la melodía buzzer
void melodiaInicio(){ ...

}

void seguParteSonido(){ ...

}

//Sonido para cuando encuentra obstaculo
void sonidoTriste(){ ...
}
```

*Ilustración 56. CÓDIGO, funciones necesarias para la melodía*

[6] Tone(), función interna de Arduino.

En cuanto a la función setup(), que es la encargada de preparar el programa, se ha rellenado con los datos necesarios para poder arrancar los modulosa si como los pines en los que va asociado cada servo y algunos estados iniciales como: la posicionInicial(), cuyo contenido realiza el bloqueo de los servos en una posición en la que este estable y en equilibrio, bocaDV, es la función que muestra por la pantalla la imagen la boca de Darth Vader, esto se hace con la matriz de bytes anteriormente declarada y por último, tenemos la melodiaInicio() que es la reproducción de una parte de la secuencia de Star Wars.

```
void setup(){

    //Setup pin modes
    //Pin para el buzzer
    pinMode(buzzer, OUTPUT);    //Declaramos que el pin 6 es de salida
    //Pines sensor ultrasonico
    pinMode(trig, OUTPUT);      //Declaramos que el pin 10 es de salida
    pinMode(echo, INPUT);       //Declaramos que el pin 9 es de entrada

    Serial.begin(9600);

    Wire.begin();                               //Inicializa bus I2C
    pantalla.begin(SH1106_SWITCHCAPVCC, 0x3C);  //Inicializa pantalla con direccion 0x3C
    pantalla.clearDisplay();                    //Limpiamos pantalla

    //Asociar servos con pines
    pieZ.attach(2);
    rodZ.attach(4);
    cadeZ.attach(7);

    pieD.attach(8);
    rodD.attach(12);
    cadeD.attach(13);

    //Estabilizar los servos
    posicionInicial();

    //BocaDV
    bocaDV();
    //Melodia inicial
    melodiaInicio();

}
```

*Ilustración 57. CÓDIGO, setup(), preparación inicial de ejecución*

Por último, tenemos la función loop(), cuyo objetivo es el de realizar en bucle su contenido. Este contenido se puede separar en dos partes principales, siendo la primera la que usaremos para calcular y obtener las mediciones del sensor ultrasónico, donde se calcula el tiempo del pulso, así como la distancia, con esto obtenemos la posición del obstáculo en cm. Datos que son almacenados y mostrados por el terminal de Arduino.

```
void loop(){

    delay(50);

    //Sensor ultrasonico
    digitalWrite(trig, HIGH); //Envio de pulso ultrasónico
    delayMicroseconds(10);    //Tiempo de reproducción
    digitalWrite(trig, LOW);  //Silenciamos el pulso ultrasónico

    //Se divide entre dos porque el pulso va y viene
    tiempo = (pulseIn(echo, HIGH)/2); //Recibir el pulso de respuesta

    //0.0343 porque la velocidad del sonido viaja a 343 m/s
    distancia = float(tiempo * 0.0343);

    //Mostramos los datos por el monitor
    Serial.print("Distancia: ");
    Serial.print(distancia);
    Serial.println(" cm");
    delay(500);
```

*Ilustración 58. CÓDIGO, loop() 1, cálculos para obtener datos del ultrasónico*

La segunda parte es la que se encarga de ejecutar los movimientos del prototipo, dependiendo de si el sensor detecta un obstáculo o no, hará una determinada secuencia de ejecuciones para esquivar dicho inconveniente o no.

```
//Ejecutamos los movimientos para esquivar el obstaculo
if(distancia<=25){

  Serial.println(distancia);    //Mostrar datos sensor ultrasónico
  triste();
  sonidoTriste();
  posicionInicial();
  delay(500);

  haciaAtras();
  haciaAtras();
  haciaAtras();
  haciaAtras();
  delay(250);

  posicionInicial();
  delay(500);

  girarIzquierda();
  delay(250);
  girarIzquierda();
  delay(250);
  girarIzquierda();
  delay(250);
  girarIzquierda();
  delay(250);
  girarIzquierda();
  delay(250);
  girarIzquierda();
  delay(250);
  girarIzquierda();
  delay(250);
  girarIzquierda();
  delay(1000);

  posicionInicial();

}else{
  bocaDV();
  adelante();

}
```

*Ilustración 59. CÓDIGO, loop()2, secuencia de movimientos superar un obstáculo*

Con esto, se ha obtenido un código, en el que se han usado diferentes módulos con los que hemos conseguido darle más calidez a la primera impresión del prototipo y unos comportamientos que, permiten esquivar obstáculos en la medida de lo posible. Siendo estos movimientos programados mediante la secuencia que se observa en la ilustración 59, de forma que este código trabaja de manera fija, solo pudiendo cambiar el comportamiento del prototipo al modificar la secuencia programada.

### *4.5.2. Actualizaciones del código*

Este código permite modificaciones, ya sean, con la secuencia que realiza para esquivar un obstáculo o añadir más funcionalidades. Tenemos que tener en cuenta que, si queremos seguir usando los sensores y módulos que ya están declarados, deberemos dejar esa parte tal cual esta.

En cuanto a las funciones de movimiento, adelante(), haciaAtras(), girarIzquierda() y girarDerecha(), se pueden modificar y perfeccionar añadiéndole otros movimientos intermedios para hacer el desplazamiento más natural y parecido al del ser humano. Por otra parte, se le puede añadir más dotes al prototipo, ya sean secuencias de baile, dar una patada o agacharse, etc. Esto tendría más sentido en añadirlas cuando se obtenga una aplicación móvil o un mando para poder controlar el movimiento a placer. Estas dos opciones se han planteado como actualizaciones futuras.

En cuanto a cómo se realizaría estas modificaciones, primero deberíamos de crear su respectiva función, está por cómo está desarrollado el código, la colocaremos antes del void setup(), por ejemplo, si se desea realizar una secuencia de movimientos que realicen un baile, declararíamos:

```
void patada (){
  for (int i = 0; i < 20; i++){
    cadeD.write(cadeDPos+i);
    rodD.write((rodDPos+10)+i);
    }
  for (int i = 0; i < 20; i++){
    rodD.write((rodDPos-10)+i);
  }
}
```

Este podría ser una primera versión para la acción de dar una patada. Tenemos el primer bucle for donde le damos un límite a la acción de 20 vueltas y el movimiento hacia el frente del servo de la cadera derecha (cadeD.write(cadeDPos+i); ) y hacia atrás de la rodilla derecha (rodD.write((rodDPos+10)+i); ), una vez realiza este movimiento tendríamos esta posición:

*Ilustración 60. Patada prototipo*

Como se puede observar, el resultado sería aproximadamente el que se muestra. Como es normal, perdería el equilibrio y se caería por el peso, se tendría que realizar pruebas moviendo el tobillo izquierdo para darle más apoyo central al peso y quizás mover unos grados la cadera izquierda hacia delante para que la cabeza se mantuviera unos grados para para atrás.

Con esto quiero hacer ver, que se le puede dar más movimientos al prototipo más allá de los que se le han dado, y que el código es totalmente modificable, pero para este proyecto no tiene sentido añadir estas acciones, ya que la principal función del prototipo es la de esquivar un obstáculo.

Pero en cambio, en las futuras actualizaciones al adquirir un modo manual, se podría realizar algunas funciones como se han descrito anteriormente

## 4.6. Tecnologías utilizadas

En el desarrollo del prototipo, se han hecho uso de varias tecnologías y herramientas, las cuales se van a hacer una reseña para entender su función dentro del presente TFG: (extender más la explicación)

- FreeCAD: aplicación libre de diseño en 3D, utilizada para el diseño de las piezas usadas en el prototipo. FreeCAD es un modelador paramétrico 3D de código abierto creado principalmente para diseñar objetos de la vida real de cualquier tamaño, dándote total libertad para construir lo que quieras poniendo el límite en tus habilidades para desarrollarlo.

*Ilustración 61. Logo FreeCAD*

También te permite esbozar formas 2D restringidas por geometría y usarlas como base para construir otros objetos. Contiene muchos componentes para ajustar dimensiones o extraer detalles de diseño de modelos 3D para crear dibujos listos para producción de alta calidad.

Es totalmente accesible siendo un software multiplataforma, altamente personalizable y extensible. Siendo una aplicación de diseño adaptativa y con amplia gama de usos, incluido el diseño de productos, la ingeniería mecánica y la arquitectura. [14]

- Meshmixer: software 3D libre con características interesantes para ayudar al usuario en el modelado y corrección de errores en los diseños desarrollados.

*Ilustración 62. Logo Meshmixer*

Es uno de los software 3D libres ofrecidos por Autodesk, que presenta varias características interesantes para ayudar al usuario en el modelado y la impresión 3D. Disponible en Windows y Mac OS, se basa en el modelado de superficies para diseñar cualquier tipo de pieza a partir de otro modelo, siendo una excelente herramienta para analizar y corregir los errores que son necesarios solventar en los diseños 3D antes de su impresión, y así evitar posibles defectos de los mismos tras la impresión.

Fácil de aprender, y está dirigido tanto a estudiantes como a profesionales que se inician en el modelado 3D. Las principales características y herramientas que tenemos al usar este software son: [15]

- Mezcla de malla de arrastrar y soltar
- Escultura 3D y Estampado de Superficies
- Conversión robusta en sólido para impresión 3D
- Patrones y celosías 3D
- Hueco (¡con agujeros de escape!)
- Estructuras de soporte de ramificación para impresión 3D

- Optimización, diseño y empaquetado automático de la orientación de la cama de impresión
- Herramientas de selección avanzadas que incluyen cepillado, lazo de superficie y restricciones
- Remallado y simplificación / reducción de mallas
- Suavizado de malla y deformaciones de forma libre
- Relleno de orificios, puentes, cierre de límites y reparación automática
- Cortes planos, reflejos y valores booleanos
- Extrusiones, superficies compensadas y superficie de proyecto a objetivo
- Tubos y canales interiores
- Posicionamiento 3D preciso con pivotes
- Alineación automática de superficies
- Medidas 3D
- Análisis de estabilidad y espesor

- Ultimaker cura: usada para la impresión 3D, en la que se puede modificar parámetros y obtener una previsualización con medidas, peso y tiempo del diseño a imprimir.

*Ilustración 63. Ultimaker*

Software libre de impresión 3D cuyo funcionamiento consiste en dividir el archivo del modelo de un usuario en carias capas, dejando una gran cantidad de detalles sobre el diseño 3D (peso, tamaño, tiempo estimado impresión, etc.) Además de una gran cantidad de parámetros modificables para obtener una previsualización deseada del resultado final, esto genera un código para la impresión, tras la finalización del proceso, dicho código se envía a la impresora y se obtiene el objeto físico final. [16]

- FDM: modelado por deposición fundida, es considerada la técnica más sencilla de impresión 3D y más económica, se basa en 3 elementos principales: una placa de impresión en la que se imprime la pieza, una bobina de filamento que sirve como material de impresión y una cabeza de extrusión. En la siguiente ilustración se puede observar los 3 elementos que hacen posible la impresión de cualquier diseño, siendo:
    - 1, la boquilla depositando el material fundido.
    - 2, el material depositado formando la pieza modelada.
    - 3, la plataforma móvil que ayuda a obtener el resultado final.

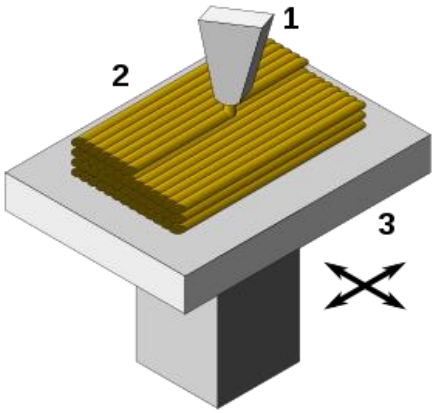



*Ilustración 64. Elementos de la impresión*

El material se deposita en capas hasta formar la pieza, mediante un filamento de plástico o metal que inicialmente se almacena en bobinas, se introduce en la boquilla que asciende su temperatura para obtener la fusión del filamento y poder generar el diseño al desplazarse por sus 3 ejes. [17]

- IDE Arduino: entorno de desarrollo usado por Arduino para insertar código en las placas. Aplicación multiplataforma, desarrollada en java y usada para escribir y cargar programas en las placas de Arduino compatibles y, placas de desarrollo de terceros.


***Ilustración 65. Logo Arduino***

El IDE de Arduino admite los lenguajes de C y C++, utilizando las reglas especiales de estructuración de códigos, este IDE proporciona una biblioteca de software del proyecto Wiring[7], que suministra muchos procedimientos de E/S.

- Lenguaje C++: lenguaje de programación orientado a objetos considerado de alto nivel. Proviene de la extensión del lenguaje C actualizándolo para poder manipular objetos, a pesar de ser un lenguaje

[7] Wiring: biblioteca software. http://manueldelgadocrespo.blogspot.com/p/biblioteca-wire.html

con muchos años, su potencial lo convierte en uno de los lenguajes de programación más usados.

*Ilustración 66. Logo C++*

Fue diseñado a mediados de los 80, su intención fue la de extender el lenguaje de C para que tuviese el mecanismo necesario para manipular objetos. Por lo tanto, C++ contiene los paradigmas de la programación estructurada y orientada a objetos. Sus principales ventajas y desventajas son:

- Alto rendimiento, siendo una de sus principales características, esto es debido a que puede hacer llamadas directas al sistema operativo, es un lenguaje compilado para cada plataforma, posee gran variedad de parámetros de optimización y se integra de forma directa con el lenguaje ensamblador.
- Lenguaje actualizado, a pesar de que tiene muchos años, el lenguaje se ha mantenido actualizado permitiendo crear, relacionar y operar con datos complejos.
- C++ se trata de un lenguaje muy amplio, lo que genera errores que complican su uso, además de un manejo de librerías que es más complicado que otros lenguajes, dejando su curva de aprendizaje muy alta. [18]

- Adafruit GFX: biblioteca de gráficos que proporciona conjunto de primitivas gráficas (puntos, líneas, círculos, etc.). Adafruit proporciona una sintaxis común y un conjunto de funciones de gráficos para todas las pantallas LCD y OLED y matrices led.

  Esto permite que los bocetos se adapten fácilmente entre los tipos de pantallas, y actualizando dicha biblioteca con características nuevas de mejoras de rendimiento y corrección de errores. Adafruit necesita de una librería adicional única para cada tipo de pantalla, siendo, Adafruit_SH1106 la específica para nuestro módulo LCD. [19]

## 4.7. Dificultades encontradas

A lo largo del desarrollo de este proyecto, han surgido numerosos imprevistos y complicaciones a los que se ha tenido que hacer frente, y por los cuales, el tiempo de ejecución ha sido más del deseado. A continuación, se exponen los contratiempos más importantes que han surgido durante la ejecución de este proyecto y las soluciones propuestas.

### *4.7.1. Coste del proyecto*

Debido a la idea principal que tenía para este prototipo y que ha sido nombrada con anterioridad, en este proyecto, se habría desarrollado un humanoide. Esta era la idea principal, pero conforme iba obteniendo información de los componentes que necesitaría y el costo del diseño, decidí cambiar y realizar el prototipo como se ha presentado en este TFG.

El principal motivo para este cambio en el diseño fue el costo de los servos, para el desarrollo de un humanoide y que pueda ser llamado así por su parecido a los humanos, mínimo se necesitaba de 11 servos, los cuales, para soportar el peso que tendría dicho humanoide, son más robustos que los mini servos metálicos usados en

el actual prototipo. Estos servos llegan a soportar pesos de 20kg, perfectos salvo por su costo, rondando los 15€ por unidad, lo que solo con estos componentes ya llegaríamos a los 165€ aproximadamente.


*Ilustración 67. Servo 5521MG*

Además, habría que añadirle que, debido a su potencia, es conveniente el uso de diseños metálicos en vez de los diseños de plástico por FDM 3D, elevando aún más el costo. Por estos motivos decidí, que el prototipo a desarrollar finalmente fuera con mini servos, pero de engranaje metálico ya que tendría que soportar un peso considerable y, por impresión en FDM 3D.

### *4.7.2. Disponibilidad de componentes*

Tal y como se ha hablado anteriormente, el prototipo de este proyecto, está formado por piezas impresas en 3D. La impresiona que se ha usado para realizar esta función ha sido la Witbox BQ, proporcionada por (SCAI) Servicios Centrales de Apoyo a la Investigación, Área de Ingeniería y Computación Científica de la Universidad de Jaén. Debido a que dicha impresora está a disposición de cualquiera que la solicite ya sea personal universitario como investigadores, profesorado o alumnado, así como

empresas externas. Ha sido aumentando el tiempo de impresión de las piezas, hasta llegar casi al mes desde su solicitud.

A parte, uno de los componentes principales no estaba en stock, la board que usaríamos para acoplar Arduino Nano, por lo que hubo que esperar hasta que la empresa aviso de su disponibilidad, lo cual no se pudo realizar pruebas de las primeras versiones del código.

### *4.7.3. Diseño y montaje del prototipo*

Puesto que no tenía conocimientos sobre diseños 3D ni había realizado anteriormente ninguna impresión en este formato, se tuvo que emplear una gran cantidad de tiempo buscando la manera en la que se realizan dichos diseños y el desarrollo de los mismos en el programa específico para ello. Esto me llevo a ver numerosas documentaciones, así como videos explicativos para obtener los conocimientos básicos que me hicieran dar los primeros pasos, y con ello, las versiones previas de los diseños finales que se han impreso.

Además, era necesario informar de los correctos parámetros de las piezas antes de imprimirlas, para que el resultado final sea el más óptimo, lo cual, también añadió más tiempo en el que conocer para que y como afectaban los parámetros antes de mandar la solicitud de la impresión donde debería especificarlos.

Por otra parte, al ser un diseño propio, no había manual ni instrucciones para realizar el montaje ni la ubicación de los diferentes módulos, así como la disposición de las piezas.

### *4.7.4. Modificación de diseños tras impresión*

Durante el desarrollo del proyecto se observaron pequeños inconvenientes en el diseño del prototipo, los principales han sido en el diseño del pieIzquierdoV2 y pieDerechoV2, que tras montar los servos en cada pie y ajustar al puenteVersionFinal,

se detectó el rozamiento excesivo incluso bloqueo, el cual, ha impidió realizar un correcto movimiento en ambos pies, por lo que se tuvo que realizar un hueco para solventar dicho problema, este hueco se realizado con una desbarbadora y un taladro.

*Ilustración 68. Banco de trabajo*

Además, posteriormente le salió otro defecto en el interior de ambos pies que ocasionaba una fricción que generaba tropiezos e inestabilidad. Después de dichas modificaciones, se quedó con un resultado final tal que así:

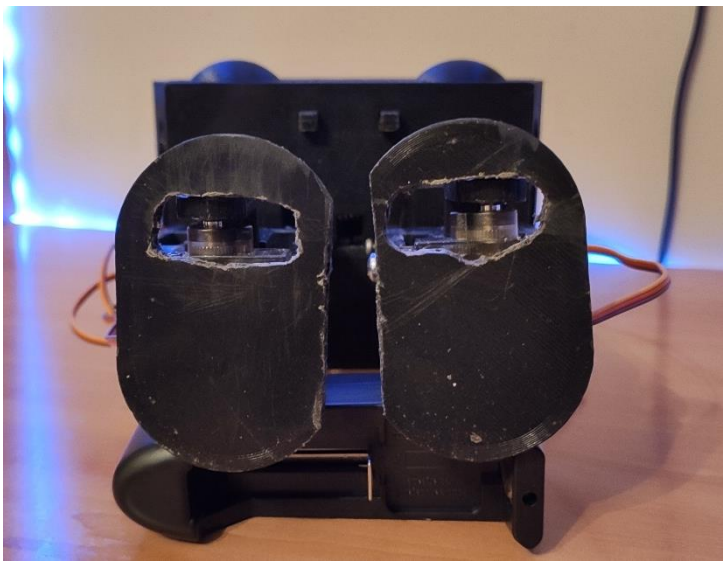


*Ilustración 69. Modificaciones en los pies*

### 4.7.5. Estabilidad por sobrepeso

La estabilidad es una de los problemas más importantes para superar los objetivos marcados, este problema de la estabilidad, ha surgido al añadir el soporte de la batería junto a la caja porta baterías y las dos baterías modelo 18650, provocando un incremento del peso el cual, al realizar los movimientos de la marcha bípeda, generaba una inestabilidad que impedía mantener el equilibrio al prototipo.


*Ilustración 70. Peso del prototipo*

Esto ha hecho que se modifique el resultado final del prototipo, dejando la batería fuera del mismo y añadiéndole más acoples a los cables, para así conseguir

mayor distancia y, poder realizar los movimientos del prototipo sin tener que mantener la batería en la mano.

Las extensiones de los cables de alimentación, se han realizado con los conectores barril, estos vinieron en un pack de 5 unidades y solo fue necesario el uso de una unidad, por lo que se cogió los otros cables, se cortaron y puentearon en los cables de alimentación de la batería para darle mayor distancia. Con estas modificaciones se ha conseguido suplir el exceso de peso y la perdida de estabilidad.

***Ilustración 71. Baterías 18650 en su porta baterías***

Por último, tras la realización de la multitud de pruebas para obtener resultados con las funciones que dan movimiento al prototipo, se llegó a generar una holgura por el exceso de movimiento en las piezas de plástico. Esto al igual que el sobrepeso, genero inestabilidad ya que los diferentes diseños no quedaban lo ajustado que debería para producir los movimientos.

En principio se usó unos tornillos rosca chapa de 3mm de grosor, lo que se ajustaba a la perfección con el diseño que se realizó. Pero como se ha comentado, tras las numerosas pruebas y los inconvenientes del peso, llego a generar la holgura que entorpecía el correcto movimiento del prototipo, por lo que se han cambiado dichos tornillos por unos de mayor grosor (4mm).

Pero el uso de estos tornillos nos obligaba a limar el orificio por donde debería de pasar este tornillo en los diseños puenteVersionFinal, al ser de mayor grosor, se creaba una fricción que en ocasiones era un bloqueo del movimiento de esta pieza, entonces bloqueaba la movilidad de los servos e impedía realizar cualquier acción por porte del prototipo.

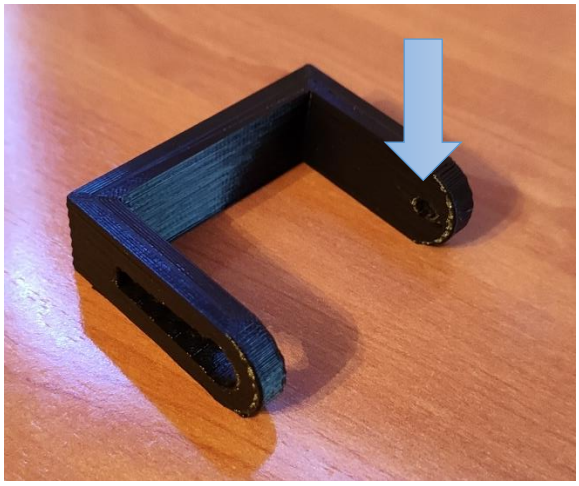


*Ilustración 72. Orificio puenteVersiónFinal*

Una vez limados los orificios por donde pasaba el tornillo, se ajustaba a la perfección a los diseños de sujeccionServoFinal, eliminando la holgura que teníamos sin perder movimiento.

### *4.7.6. Problemas de compatibilidad y componentes*

En cuanto a la compatibilidad, el problema lo generó una librería fundamental para el uso del módulo ultrasónico, NewPing, esta librería no se incluye inicialmente en el entorno de desarrollo Arduino, cuya función es la de mejorar el rendimiento y solventar algunos de los problemas que se generan en las detecciones de distancia.

Esta librería, entraba en conflicto con una librería interna de Arduino llamada tone y usada para el zumbador pasivo que hemos usado. Esto me hizo adaptar las variables y métodos que se usarían con el módulo ultrasónico para poder usarlo sin necesidad de esta librería y poder solventar dicho problema.

Para el problema del componente, se sufrió un gran retraso ya que este componente se estuvo esperando a que la web notificara su stock durante algún tiempo, y tras su posterior adquisición, se realizaron las pruebas de código las cuales, me hicieron ver que estaba defectuosa la board.

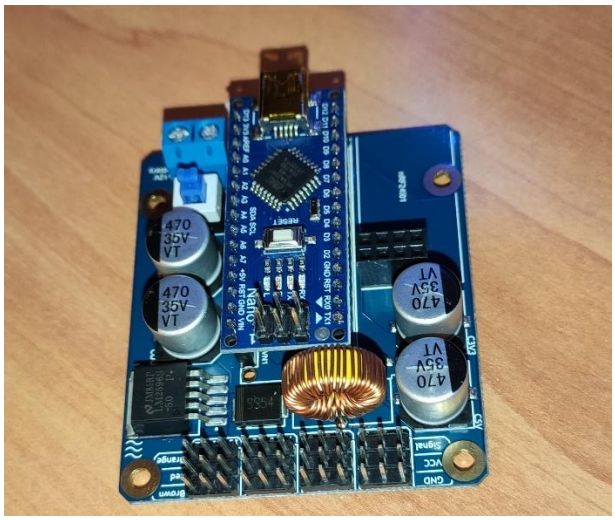



*Ilustración 73. Arduino Nano acoplada a la board*

Tras hablar con la empresa, decidí adquirir Arduino Uno para poder continuar de la manera más rápida posible con el prototipo, ya que la empresa me ofrecía una devolución tras la realización de pruebas que realizarían, y esto incrementaría aún más el tiempo que necesitaría esperar.

### *4.7.7. Exceso de rapidez en los servos*

Los servos, necesitan de la librería interna servo.h para poder ser usados, pero esta librería genera un movimiento excesivamente rápido de los servos, esto hacia

una pérdida de estabilidad muy difícil de solventar que, tras intentar hacer un movimiento lento y añadiéndole directamente los ángulos a los servos y generando el movimiento en mini funciones como se muestra esta primera versión del código:

```
void MoverPieIzqDelante(){

  pieZ.attach(sPieZ);
  rodZ.attach(sRodZ);
  cadeZ.attach(sCadeZ);

  cadeZ.write(45);
  rodZ.write(92);
  pieZ.write(90);
  delay(500);
}

void MoverPieDerDelante(){

  pieD.attach(sPieD);
  rodD.attach(sRodD);
  cadeD.attach(SCadeD);

  //Mover pierna derecha delante
  cadeD.write(115);
  rodD.write(100);
  pieD.write(70);
  delay(500);
}

void MoverPieDerInicio(){

  pieD.attach(sPieD);

  //Volver a la posicion inicial al pie

  pieD.write(70);
  delay(500);
```

***Ilustración 74. Primera versión del código***

No se pudo aumentar la estabilidad en los movimientos. Esto me hizo investigar y había una librería perfecta para este problema y muy usada por la comunidad VarSpeedServo[8] [20]. Esta librería permite el movimiento de hasta 8 servos de forma asíncrona, estableciendo la velocidad de los mismos e incluso la creación de secuencias.

[8] VarSpeedServo: librería que controla la velocidad del movimiento de los servos.

Esta opción era la más idónea que podía usar para solventar este problema, pero al ponerla en marcha, no conseguí poder ejecutar los servos a las velocidades que se suponía que permitía esta librería. Esto me hizo tras muchas pruebas, el descartar esta gran librería e intentar solventar el problema con la librería principal que incorpora Arduino.

La solución que se encontró, ha sido la que se ha mostrado en el apartado de Implementación.

### *4.7.8. Retrasos temporales por diferentes causas*

Durante la realización de este proyecto se han sufrido múltiples retrasos temporales como ya hemos nombrado, la impresión de las piezas del prototipo, que llevo a generar más retraso del que se había calculado. También el montaje del prototipo, que añadió más problemas por los diferentes defectos que fueron surgiendo. A parte, se ha sufrido retraso en la adquisición de uno de los componentes importantes y su fallo posterior, el cual, añadió más retraso.

Por último, la finalización del grado en los exámenes finales, junto a un postgrado ofrecido por la Universidad de Jaén, hicieron que en algunos meses la disponibilidad para poder avanzar en el proyecto, fueran limitadas e incluso nulas.

## 4.8. Puntos a destacar

Para finalizar este capítulo, y como se ha podido entender en el mismo, la elección de cada acción que ha repercutido en el resultado final, ha sido expuesta, haciendo más hincapié en los problemas sufridos, ya que estos han generado las modificaciones de diversos puntos de este aparado como son el diseño del prototipo final y el código software desarrollado, los cuales, en su conjunto, generan la posibilidad del movimiento del prototipo.

Por terminar este capítulo, destacar el primer apartado del mismo, donde se explica más personalmente el porqué del diseño final del prototipo. Donde se intenta mostrar el interés hacia esta rama de la informática que, gracias al desarrollo de este TFG, se ha visto incrementada.

# 5. PRUEBAS Y RESULTADOS

En este capítulo, se muestran las diferentes pruebas realizadas sobre el prototipo, así como los resultados obtenidos en dichas pruebas. Cabe destacar que estarán diferenciadas en dos apartados, los cuales serán una batería de pruebas en las que se intensifico holguras en el prototipo y, una segunda batería de pruebas donde se solventa ese problema.

Para ello, a continuación, se detallan diferentes parámetros del prototipo que han sido usados para las pruebas:

- (TiemEq), el tiempo en el que se mantiene en equilibrio a caminar.
- (NumD), número de pasos que realiza la pierna derecha.
- (NumZ), número de pasos que realiza la pierna izquierda.

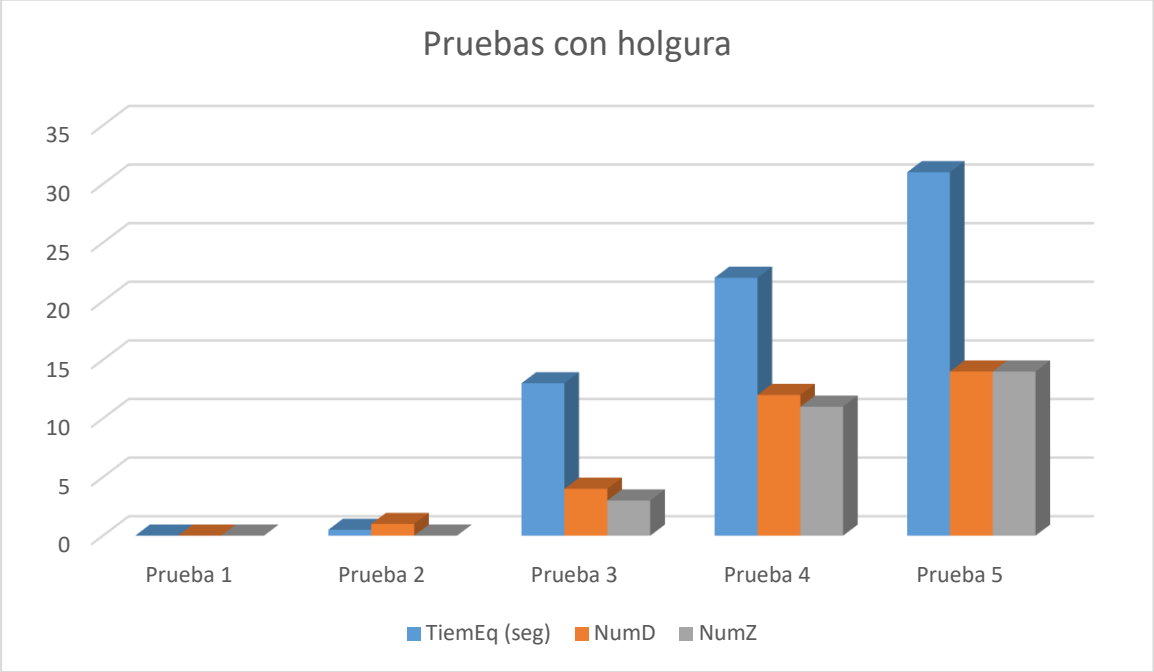
## 5.1. Pruebas con holgura

Con estos parámetros y con las siguientes tablas, se muestra algunas de las pruebas que han sido realizadas. La medición en centímetros son una aproximación, ya que el prototipo no tiene un movimiento completamente recto, desplazándose ligeramente hacia su derecha.

A continuación, se muestran las pruebas y resultados de los movimientos, tras 15 segundos aproximadamente de movimiento hacia el frente, se tapará el módulo ultrasónico para observar el comportamiento que genera al realizar la secuencia completa de movimientos para esquivar un obstáculo. Cabe destacar que estas pruebas fueron realizadas con holguras en algunos de las piezas, uno de los problemas mencionados anteriormente y, en cada prueba se modificó el código para solventar la causa de la caída en el caso de así serlo.

| | TiemEq (seg) | NumD | NumZ |
|---|---|---|---|
| **Prueba 1** | 0 | 0 | 0 |
| **Prueba 2** | 0,5 | 1 | 0 |
| **Prueba 3** | 13 | 4 | 3 |
| **Prueba 4** | 22 | 12 | 11 |
| **Prueba 5** | 31 | 14 | 14 |

*Tabla 19. Pruebas con holgura*



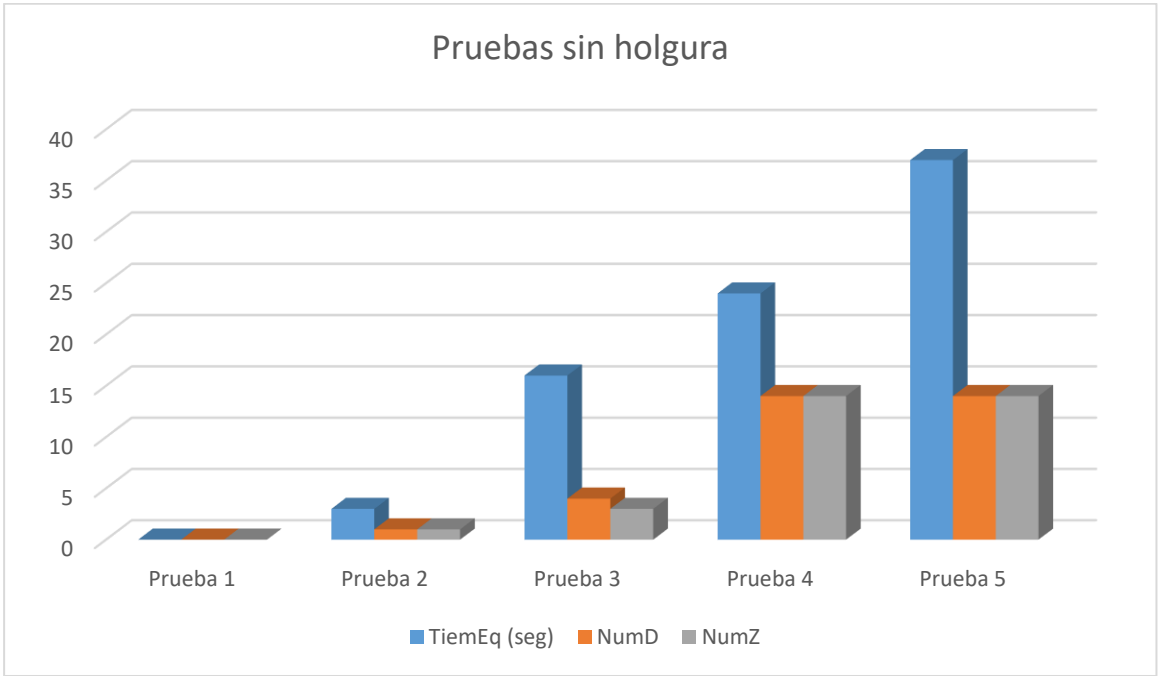

En la prueba 1 no realizo ningún avance, la prueba 2 dio un paso y termino cayéndose, en la prueba 3 camino varios pasos hacia el frente, pero con tambaleos, lo que propicio que no pudiera mantener la estabilidad. La prueba 4, realizando el movimiento hacia atrás, sufrió una pérdida de potencia en la pierna izquierda, lo que ocasiono que se cayera de espaldas, y para la última ejecución, la prueba 5, se mantuvo a pesar de los tambaleos, en equilibrio, lo que le llevo a completar la secuencia completa aun que a duras penas por el tambaleo que generaba.

## 5.2. Pruebas sin holgura

Para la segunda batería de pruebas, se solvento el problema de la holgura que provocaba que se tambaleara en sus movimientos. Como los parámetros del código estaban implementados con las pruebas anteriores, al eliminar la holgura, este código necesito de modificaciones puesto que algunas posiciones provocaban que no se mantuviera el equilibrio y se golpeara contra el suelo. Conforme se avanza en las pruebas, se va mejorando el posicionamiento y la estabilidad, consiguiendo resultados satisfactorios en la última prueba.

| | TiemEq (seg) | NumD | NumZ |
|---|---|---|---|
| **Prueba 1** | 0 | 0 | 0 |
| **Prueba 2** | 3 | 1 | 1 |
| **Prueba 3** | 16 | 4 | 3 |
| **Prueba 4** | 24 | 14 | 14 |
| **Prueba 5** | 37 | 14 | 14 |

*Tabla 20. Pruebas sin holgura*



Para la primera prueba, se ha obtenido el mismo resultado que en la batería de pruebas anteriores, a consecuencia como ya se ha descrito previamente, al tener el código anterior, no realizo ningún avance siendo su primer movimiento el de caerse, con la segunda prueba, se solvento el primer problema consiguiendo unos primeros pasos con ambas piernas, pero posteriormente se desplomaba al perder el equilibrio.

Con la prueba 3 se completó el movimiento hacia el frente, justo cuando entro la secuencia que le hacía andar hacia atrás, termino por caerse con un ángulo más forzado que proco la perdida de equilibrio, para la prueba 4, se llegó a completar los movimientos hacia el frente y para atrás, pero al llegar al giro se perdió la estabilidad, y por último, en la prueba 4 se realizó el recorrido por completo, con mejor estabilidad y equilibrio que en las anteriores baterías de pruebas, dándole más consistencia en sus desplazamientos.

## 5.3. Puntos a destacar

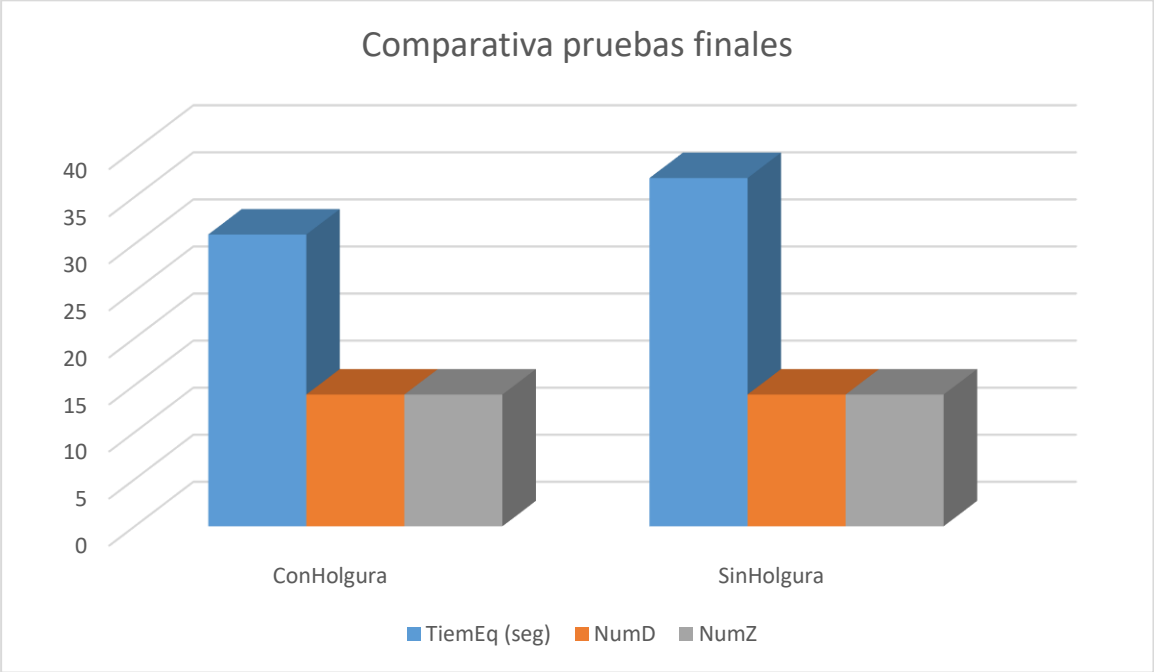
Como se ha podido observar en las pruebas, ha habido dos tipos diferentes de pruebas, siendo las pruebas con holgura las más sufridas sin lugar a duda, ya que, por ese problema, costaba mucho el poder ejecutar la secuencia completa para esquivar un objeto.

Para la segunda batería de pruebas, este problema se solvento lo cual ayudo mucho para que el prototipo pudiera realizar la secuencia completa, pero en las primeras pruebas se sufrió los desajustes ocasionados en las primeras pruebas

Con estas pruebas cabe destacar que, ambas fueron satisfactoriamente superadas a pesar de los inconvenientes con unos tiempos muy similares ya que el código solo variaba en los ángulos de los servos.

| | TiemEq (seg) | NumD | NumZ |
|---|---|---|---|
| **ConHolgura** | 31 | 14 | 14 |
| **SinHolgura** | 31 | 14 | 14 |

*Tabla 21. Comparativas de las pruebas*




El tiempo de sin holgura es mayor que el de con holgura ya que, al ver el recorrido total, se le añadió más funciones para optimizar el esquivo del obstáculo.

# 6. CONCLUSIONES Y TRABAJOS FUTUROS

Como conclusión puedo decir que durante el desarrollo de este TFG se ha sufrido una serie de hechos que han provocado que, debido a la falta de tiempo, no se haya podido completar al 100% algún requisito propio como incorpora la batería dentro del prototipo.

Pero cabe destacar, que, tras la finalización de este proyecto, se han adquirido conocimientos sobre diseño 3D, así como un amplio conocimiento sobre la robótica. Conocimientos que me van a servir para poder actualizar este proyecto y conseguir el objetivo propio de crear un humanoide por mí mismo.

El objetivo del diseño se ha cumplido gracias al empleo del programa FreeCAD. Para ello, se han diseñado varios prototipos, obteniendo finalmente un diseño de bípedo, que, junto al otro objetivo hardware y software, mediante Arduino, algunos de los módulos compatibles y la programación en C++, ha sido capaz de realizar la marcha bípeda y, la superación de obstáculos que impidan su desplazamiento, consiguiendo así superar los objetivos de este TFG.

En resumen, señalar que este prototipo alcanza los objetivos marcados para este proyecto, objetivos que dicho queda, van más allá de un TFG común en este grado. Pero como se comentó anteriormente, era un deseo personal el poder realizar este proyecto dado mi pasado y gracias los conocimientos adquiridos a lo lardo de este grado, cuya etapa de formación ha sido enriquecedora, pudiendo realizar este TFG, a pesar de la inmensa falta de documentación sobre los puntos más importantes del proyecto.

## 6.1. Futuras actualizaciones

Para finalizar el presente proyecto fin de carrera, se proponen una serie de estudios que continúan la línea de trabajo que se ha ido desarrollando en este documento. Dichos trabajos están orientados en la actualización y futuras modificaciones que se implementaran en este prototipo.

Los desarrollos futuros propuestos son:

- *Diseño optimizado.*

  Mejorar y consolidar el diseño inicial de las piezas que formas el prototipo con el fin de mejorar los aspectos que han generado problemas, así como reducir tiempo y coste de la fabricación y montaje del prototipo.

- *Actualización de las baterías.*

  Estudiar otras baterías para suplir el problema de la perdida de estabilidad por la sobrecarga.

- *Control desde mando o dispositivo móvil.*

  Diseñar un mando con el cual, poder dar órdenes al prototipo, imprimiendo las piezas, comprando los módulos, joystick y demás componentes necesarios.

  A parte, se le puede dotar de una aplicación móvil, que sea capaz de controlar movimiento del prototipo a través.

- *Movimiento sensor ultrasónico.*

  Una idea que se me ocurrió conforme realizaba el desarrollo del prototipo fue la de añadirle una funcionalidad extra al sensor ultrasónico, la cual, le diera la mejor opción posible para seguir con el movimiento de esquive.

Esto se conseguiría acoplando dicho sensor en un servo el cual le proporcionara más campo de visión realizando giros de izquierda a derecha, para poder detectar si nos encontráramos en una esquina, la parte donde no hubiera una pared, con esto se conseguiría darle más inteligencia al prototipo y reducir los movimientos que implicarían el esquivar sin usar esta funcionalidad.

## 6.2. Guía original del Trabajo Fin de Título

El objetivo principal consiste en el diseño y prototipado de un robot autónomo y sensible al contexto. Para ello será necesario:

- Identificación de escenarios de funcionamiento del prototipo.
- Identificar los requisitos hardware y software necesarios.
- Adquisición de los elementos necesarios para la construcción del prototipo.

## 6.3. Anexo

Documentación entregada en este TFG:

- Memoria de trabajo

  Desarrollo e información completa del proceso de creación del objetivo de este TFG.

- Documentación

  Autorización para su publicación con fines educativos y de investigación. Anexo necesario para obtener la aprobación y optar

a la defensa con el tribunal asignado y el código desarrollado para el prototipo.

- Vídeo demostración

  Puesta en marcha y funcionamiento del prototipo en un entorno adaptado para obtener los resultados de su finalidad.

- Imágenes

  Archivos STL de los diseños finales que se han desarrollado para su impresión y que han servido para la construcción del prototipo.

- Diseño final prototipo modificable

  Archivos FCStd para la visualización y modificación en 3D del diseño realizado en FreeCAD.

- Esquemas de los diseños

  Esquema de las piezas diseñadas para el prototipo con el programa FreeCAD en formato SVG.

## 6.4. Puntos a destacar

Para finalizar este capítulo, podemos destacar que, gracias a este proyecto, se han adquirido grandes conocimientos en cuanto a la robótica, así como al diseño 3D. A parte, se han hablado de algunas actualizaciones futuras, que le darían más funcionalidades al prototipo y, se han presentado los objetivos marcados para este trabajo fin de grado, los cuales, a lo largo de este documento se han ido exponiendo las superaciones de los mismo. También, se ha añadido un anexo con el que documentamos los complementos que serán adjuntados a la entrega final del TFG.

# 7. BIBLIOGRAFÍA

[1] D. Sandrone, Autómatas, robots y androides: parecidos pero diferentes. https://www.lavoz.com.ar/numero-cero/automatas-robots-y-androides-parecidos-pero-diferentes/

[2] J. Ruiz de Gariba y Pascual, Robótica: Estado del arte. https://www.academia.edu/913608/Rob%C3%B3tica_Estado_del_arte

[3] S. Parra, Este fue el primer robot de la historia y podía decir hasta 700 palabras en voz alta y fumaba cigarrillos.https://www.xatakaciencia.com/robotica/este-fue-primer-robot-historia-podia-decir-700-palabras-voz-alta-fumaba-cigarrillos#:~:text=ELEKTRO%2C%20el%20primer%20androide,un%20fon%C3%B3grafo%20de%2078%20rpm).

[4] R. Álvarez, El impresionante robot con inteligencia artificial. https://www.xataka.com.mx/robotica-e-ia/impresionante-robot-inteligencia-artificial-erica-sera-primero-mundo-protagonizar-pelicula-ciencia-ficcion

[5] ElEconomista, Paises con más robots trabajando por humano. https://www.eleconomista.es/economia/noticias/11145029/04/21/Estos-son-los-paises-con-mas-robots-trabajando-por-humano-Espana-entre-ellos.html

Alfredo Moreno Muñoz, Sheila Córcoles Córcoles, Aprende Arduino en un fin de semana, Time of Software.

[6] Esneca, Clasificación de los robots. https://www.esneca.com/blog/clasificacion-de-los-robots-segun-su-funcion/

[7] U. Autónoma de Querétaro (México), Diseño, modelado y simulación de un robot bípedo: marcha bípeda. http://www.mecamex.net/revistas/LMEM/revistas/LMM-V03-N01-02.pdf

[8] C. C. Atribution-ShareAlike. ¿Qué es Arduino? https://www.arduino.cc/en/Guide/Introduction

[9] Y. Fernández, Qué es Arduino, cómo funciona y qué puedes hacer con uno,https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

[10] Maida, Esteban, Paciencia, Julian, Metodología y diferenciación entre tradicional y ágil, https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

[11] Systems group, La ingeniería de software, ¿qué es?, https://systemsgroup.es/tecnologias-de-la-informacion/la-ingenieria-de-software-que-es-y-que-utilidad-tiene/32363/

[12] M. José, Ujaen, Ingeniería de requisitos.

[13] Nicksort, melodía Star Wars, https://create.arduino.cc/projecthub/HiHiHiHiiHiiIiH/star-wars-on-a-buzzer-0814f2

[14] Frecadweb, ¿qué es FreeCAD?, https://www.freecadweb.org/

[15] Alicia M, MeshMixer, el software gratuito de modelado 3D, https://www.3dnatives.com/es/meshmixer-software-modelado-261020202/#!

[16] Ultimaker, ¿qué es ultimaker cura?, https://ultimaker.com/es/software/ultimaker-cura

[17] Stratasys, ¿qué es la tecnología de impresión 3D FDM?, https://www.stratasys.com/es/fdm-technology

[18] Ángel R, qué es C++: características y aplicaciones, https://openwebinars.net/blog/que-es-cpp/

[19] Dherrada, ¿qué es Adafruit GFX?, https://github.com/adafruit/Adafruit-GFX-Library

[20] VarSpeedServo, librería para el movimiento de los servos, https://github.com/netlabtoolkit/VarSpeedServo

[21] S. Coddington, Boris, https://www.instructables.com/BORIS-the-Biped-for-Beginners-and-Beyond/

[22] Technovation, bípedo, https://www.instructables.com/Arduino-Controlled-Robotic-Biped/

[23] Bonnetx, bípedo 4 servos, https://www.instructables.com/Walking-biped-4-servos-per-leg/