Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Ingeniería Informática

Proyecto Final

Robot cuadrúpedo para entretenimiento y aprendizaje de niños y jóvenes

Autor del proyecto: Sergio Lasheras Cariñena

Directora del proyecto: María Francisca Pérez Pérez

Lugar, 24 de junio de 2022

universidad SANJORGE
GRUPO SANVALERO

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma:  Fecha: 24 de junio de 2022

Dedicatoria y Agradecimiento

Pareja, familia y amigos.

María Francisca (Paqui) Pérez por haber querido llevar este proyecto incluso estando en una situación compleja.

Profesores y personal de la universidad en general.

Discord, Github, Stackoverflow, Boston dynamics, YouTube, Arduino Forum.

Tabla de contenido

Resumen 9

Abstract 9

1. Introducción 11

2. Antecedentes 13

3. Objetivos 15

4. Metodología 17

4.1 Planificación inicial 19

4.1.1. Primera entrega (19/11/2021) 19
4.1.2. Segunda entrega (17/12/2021) 20
4.1.3. Tercera entrega (31/12/2021) 20
4.1.4. Cuarta entrega (28/01/2022) 20
4.1.5. Quinta entrega (25/02/2022) 20

5. Diseño y desarrollo del prototipo y los kits 22

5.1 Fabricación Spot Micro 22

5.1.1. Estructura 23
5.1.2. Electrónica y circuitos 25
5.1.3. Comunicaciones 30

5.2 Programación de la placa Arduino MEGA y los sensores. 32

5.2.1 Declaración de variables 34
5.2.2 Funciones 35

5.3 Diseño y desarrollo de la aplicación 39

5.3.1 Pantalla principal 40
5.3.2 Instrucciones 40
5.3.3 Control remoto 40
5.3.4 Secuencias 41

5.4 Kits de desarrollo 42

5.4.1 Kit de desarrollo básico 42
5.4.2 Kit de desarrollo medio 43
5.4.3 Kit de desarrollo avanzado 43

6 Estudio económico 45

7 Resultados 49

7.1 Desviaciones en la planificación inicial 49

7.1.1 Primera entrega (19/11/2021) 49
7.1.2 Segunda entrega (17/12/2021) 49
7.1.3 Tercera entrega (23/02/2022) 49
7.1.4 Cuarta entrega (08/06/2022) 49

7.2 Pruebas, problemas y posibles mejoras durante el desarrollo 50

7.2.1 Mecánica 50
7.2.2 Electrónica 52
7.2.3 Código 55
7.2.4 Aplicación 57

7.3 Contribuciones de este proyecto respecto a otros similares. 58

7.4 Variaciones respecto a la propuesta inicial. 58

8 Conclusiones 61

9 Tablas y figuras 63

10 Glosario de siglas 65
11 Bibliografía 67
12 Anexos 69

Resumen

El objetivo de este proyecto es el Diseño y fabricación de un robot cuadrúpedo para ser usado en entretenimiento y aprendizaje de niños y jóvenes.

Para alcanzar el objetivo de este proyecto se va a juntar el proyecto existente "spot micro" y los "kits de Arduino" en uno solo y diseñar un kit de spot micro adaptado para ser usado en entretenimiento y aprendizaje de niños y jóvenes. El kit se adaptará en función de la de la cantidad de piezas disponibles y dificultad de montaje para que independientemente del usuario puedan disfrutar de la experiencia por igual.

Dado que es un proyecto que van a usar niños y contiene componentes peligrosos como baterías o la propia electrónica, se diseñará un sistema de detección de fallos que contiene amperímetros, voltímetros, sensores de proximidad e indicadores tanto en pantalla como en sonido del estado del robot en todo momento.

El resultado de este proyecto consta de una aplicación móvil con las instrucciones de montaje y con capacidades de control remoto, así como acceso a una pantalla de programación sencilla en la que por medio comandos la gente más joven podrá hacer una secuencia de comportamiento y la gente más adulta tenga acceso al código original tanto del robot como de la aplicación (MIT App Inventor) para añadir, quitar o modificar cualquier componente o comportamiento del robot.

Términos: Robot, Spot micro, Arduino, Bluetooth, Android

Abstract

The objective to this project is to design and manufacture a quadruped robot to be used in entertainment and learning for children and young people.

To achieve the objective of this project, the "spot micro" project and the "Arduino kits" are going to be merged into one concept and design a micro spot kit adapted for different ages and levels of knowledge depending on the number of parts available and the difficulty of assembly.

Since it is a project that will be used by children and contains dangerous components such as batteries or the electronics itself, a fault detection system will be designed that contains ammeters, voltmeters, proximity sensors and indicators both on the screen and sound of the status of the robot at every moment.

It will include an app with assembly instructions and remote-control capabilities, as well as access to a simple programming screen in which, by means of boxes or commands, younger people can make a sequence of behavior and older people have access to the original code of both the robot and the app (google app Inventor) to add, remove or modify any component or behavior of the robot.

Terms: Robot, Spot micro, Arduino, Bluetooth, Android

1. Introducción

En este mundo en el que vivimos, cada vez con más progresos tecnológicos, un mundo cada vez más digital, es útil y necesario que los jóvenes empiecen cada vez más pequeños a acercarse a la tecnología, que convivan con ella para poder aprender a utilizarla de manera muy temprana.

Hasta ahora esto se limitaba a estudiar cómo usar un ordenador, manejarse por internet, ofimática, usar lenguajes de programación visuales como los legos mindstorms o juguetes programables por cajitas para poder aportar ese pensamiento lógico, pero solo se les enseñaba "tecnología" por la parte del software.

Existe una problemática generalizada a la hora de entender cómo funcionan muchos dispositivos cotidianos, desde electrodomésticos sencillos como batidoras, herramientas como taladros… y uno de los objetivos de repartir estos kits de robótica es dar ese conocimiento desde pequeños para poder reconocer diferentes tipos de fallos en dispositivos incluso ser capaces de arreglarlos o repararlos lo necesario hasta llevarlos a lugares especializados.

Otro de los objetivos de estos kits son el poder explicar a los jóvenes (y no tan jóvenes) robótica, informática… mucho más compleja usando como base estos kits y construyendo el nuevo y más avanzado conocimiento modificándolos. Por último, el tercer objetivo principal es el entretenimiento durante el aprendizaje.

Dado que este proyecto toma muchos campos del conocimiento diferentes tales como telecomunicaciones, programación, electricidad, electrónica digital, mecánica, diseño de producto, ingeniería del software e interacción humano-maquina, el proyecto se ha dividido secciones de similar ámbito.

Primero se realiza un análisis de los antecedentes, de donde se parte, que existe y lo que se quiere conseguir.

Como punto tres se indica una lista de los objetivos del proyecto, donde los principales son el diseño de los diferentes kits, programación en una plataforma con recursos limitados y la creación de la aplicación con el tutorial, control remoto y secuencias. El resto de los objetivos indicados tienen como objetivo conseguir los principales de la mejor forma posible.

Posteriormente, un análisis de la metodología utilizada, con sus motivos, y una planificación inicial diseñada para una situación en la que se le podía dedicar 4 horas diarias al proyecto. El resultado final de los tiempos del proyecto se indicará más adelante en la sección de resultados.

El punto cinco es el más amplio de todos. En él se incluye el diseño y desarrollo principal del proyecto separado en cuatro amplios bloques.

- Fabricación de spot micro: En esta sección se explica cómo está construido el robot en el que se incluyen el diseño externo, los sistemas eléctricos y electrónicos que contiene y los protocolos y comunicaciones que utiliza para funcionar de forma correcta.

- Programación de spot micro: Se explica la plataforma Arduino y se realiza un análisis de los motivos por los que el código del robot está programado de esa manera, indicando tres subsecciones principales las cuales son la declaración de variables (y configuración

inicial de las librerías utilizadas) y las funciones principales (setup() y loop()) y secundarias.

- Diseño de la aplicación: Se analiza cómo funciona la página utilizada para la realización de la aplicación (MIT App Inventor 2) y cómo se ha construido la app para que funcione de la manera que se requiere.

- Diseño de los kits: Se diseñan tres kits, el básico, medio y avanzado indicando en cada uno de ellos lo que contienen, pensando en las dificultades, conocimientos y seguridad que necesitan tener los rangos de edades correspondientes (7-12, 13-17, +18).

- Problemas y mejoras: Aquí se explican los problemas que puedan surgir durante el desarrollo, se analizan las posibles causas y se proponen mejoras para solventarlos o que funcionen mejor.

El siguiente apartado es el estudio económico, en el cual se ha recogido un presupuesto de piezas necesarias para la construcción y un análisis de posibles precios de cara al consumidor final teniendo en cuenta piezas, mano de obra y la recuperación de la inversión inicial.

El último de los puntos importantes es el capítulo de resultados, en el que se analiza el desempeño del proyecto, si los objetivos han sido completados, las desviaciones respecto a los tiempos iniciales y que tiene esta versión de especial comparada con otras propuestas similares.

Los últimos capítulos incluyen unas conclusiones del proyecto, un glosario de términos, la bibliografía y anexos.

2. Antecedentes

Hace unos años, a la empresa Boston Dynamics (Dynamics, BostonDynamics, s.f.) que se dedican a hacer robots de altas prestaciones con características de animales se hizo muy famosa por la difusión de videos de estos robots bailando (Dynamics, Youtube, s.f.). Los dos más conocidos de esta empresa actualmente son el spot mini (figura 1) (con forma de perro y en el que se basa este proyecto por los motivos que se explicaran más adelante) y un robot antropomórfico (figura 2), ambos con amplio rango de movilidad.

Figura 1: Spot Mini de Boston Dynamics

Figura 2: Atlas de Boston Dynamics

También existe un grupo de personas que formaron una comunidad alrededor del proyecto opensource llamado OpenCat (figura 3) y basado en prototipos creados por Boston Dynamics. Según su creador original en 2016, OpenCat es un robot cuadrúpedo basado en Arduino y Raspberry Pi con forma de mascota (gato). El creador se llama Petoi y es experto en realizar mascotas robóticas programables. (PetoiCamp, s.f.)

Figura 3: OpenCat de Petoi

Posteriormente Boston Dynamics presentó el llamado Spot Mini, un robot cuadrúpedo desarrollado para asistir en tareas de reconocimiento y trabajo en áreas poco accesibles y de riesgo. Sus usos, por ejemplo, pueden variar de forma muy amplia entre tareas de seguridad, rescate, detección de fugas de gas, tareas de análisis de estructuras mapeo en 3D en arquitectura, transporte de material peligroso… y se le pueden añadir multitud de accesorios como un brazo con pinzas, sensores LIDAR, puntos de acceso móviles de internet y telefonía… Incluso puede ser pilotado en primera persona con gafas de realidad virtual. (PlainConcepts, s.f.)

Como unión entre OpenCat y Spot Mini, nace "spot micro", más abajo en la figura 4. Una versión mejorada de OpenCat con el diseño de Spot y con mayor tamaño capaz de almacenar más

sistemas informáticos, electrónicos y sensores en su interior, a la vez que hace más fácil su manipulación. Por este motivo, la comunidad hizo OpenCat más grande, más potente, y con más inteligencia.

Figura 4: Spot micro

Tanto OpenCat, como su versión más avanzada Spot Micro (de la que se crea una rama de desarrollo en este proyecto) no se centra en una materia concreta, sino que bebe un poco de todas. No alcanza la excelencia en ninguna de ellas sino se ve cómo se relacionan y cómo funcionan todas ellas juntas. Por ello, este proyecto parte de la investigación de todos esos campos.

Durante años han existido multitud de kits de lego, de Arduino, de Raspberry y otras plataformas y webs como Kiwico (Kiwiko, s.f.) con las que muchas personas de todas las edades han podido aprender física, química, informática, electricidad, electrónica, programación… de una manera práctica explorando los prototipos y proyectos que ofrecen.

Dado que la plataforma Arduino es la base de este proyecto. La decisión de utilizar Arduino parte de ser una plataforma opensource, bien establecida en el mercado, barata, muy sencilla de hacer funcionar (aunque complicada de dominar) y con amplias posibilidades. Lo que se propone podría asemejarse a los kits que existen para desarrollar un brazo robótico (BRACCIO, s.f.), un vehículo de 2 ruedas autobalanceable tipo segway (Keyestudio, Keyestudio Self balancing Car, s.f.) y un coche que sigue una línea pintada en el suelo (Keyestudio, Keyestudio mini tank v3.0, s.f.). Estos kits incluyen unas instrucciones junto a todos los componentes, tanto eléctricos como mecánicos necesarios para construir estos vehículos o artilugios. Juntando la esencia de estos kits y el proyecto Spot Micro, salen kits de aprendizaje para niños y jóvenes de diferentes edades en las que se incluye las piezas del robot (mayor o menos cantidad según la dificultad), la aplicación con las instrucciones de montaje y control remoto (básico y por comandos) y el acceso a esa aplicación, código y recursos para que todo el mundo que quiera pueda modificarla a su gusto.

La decisión entre convertir Spot Micro (concretamente el modelo 3D de la figura 5 (Kubina, s.f.), porque a la hora de imprimir las piezas, no necesita imprimir andamios con lo que el ahorro de material y la sencillez de impresión son puntos a favor) y no otros proyectos similares (BRACCIO, s.f.) (Keyestudio, Keyestudio Self balancing Car, s.f.) (Keyestudio, Keyestudio mini tank v3.0, s.f.) viene de que tras un análisis exhaustivo del mercado de los kits, no hay ninguno en el que haya que intente simular ser un animal cuadrúpedo, mostrando así la alta complejidad de movimientos que poseen este tipo de robots frente a los que usan ruedas, orugas u otros métodos de movimiento.

Figura 5: Spot micro de Michael Kubina

3. Objetivos

Los objetivos principales de este PFG son:

- Diseño de los diferentes kits
- Programación en una plataforma con recursos limitados
- Creación de la aplicación con el tutorial, control remoto y secuencias.

Para alcanzar los objetivos principales, los objetivos secundarios son:

- Diseño de hardware. Se incluyen sensores para gestionar el correcto funcionamiento del robot y detectar fallos en tiempo real
    - Voltímetro: Medición del voltaje que sale de la batería.
    - Amperímetro: Medición de la corriente que circula por el circuito.
    - Sensor de ultrasonidos: Utilizado para medir distancias similar al radar o sónar.
    - Pantalla OLED: visualización del estado del robot.
    - Buzzer: Pequeño altavoz que pita cuando suceden eventos concretos.

- Diseño de un código con el que poder usar estos componentes electrónicos para permitir que se mueva, teniendo en cuenta las limitaciones de memoria y potencia del chip ATmega328P (microcontrolador del Arduino UNO). (Arduino, Arduino, s.f.)
    - Uso de dispositivos simultáneos en una CPU Single-Thread.
    - Máquina de estados según input de aplicación y sensores.

- Diseño de una aplicación con 3 pantallas básicas:
    - Tutorial de montaje, con diferentes versiones para cada grupo de edad.
    - Control remoto básico del robot por bluetooth.
    - Secuenciador de acciones del robot por bluetooth.

4. Metodología

Para un proyecto a gran escala es necesario tener desde un principio una metodología con la cual vas a basar el proceso de desarrollo.

Existen dos tipos de metodologías, la tradicional y la ágil. La tradicional, clásica, consiste en una única iteración en la que deja poco espacio para revisiones y análisis de requisitos o funcionalidades durante el desarrollo. Las ágiles sin embargo plantean la posibilidad de meter varias iteraciones con las que poder revisar varios aspectos del proyecto y adaptar las siguientes iteraciones según el estado de las anteriores.

En la figura 6 se puede ver una pequeña grafica del proceso de desarrollo en una metodología tradicional dónde sólo hay iteración frente a las metodologías ágiles en las que hay varias iteraciones.

Figura 6: Metodologías Ágiles frente a tradicionales

Dada la naturaleza del proyecto (aprendizaje e investigación con continuos cambios durante el desarrollo) las metodologías agiles fueron consideradas mejor opción.

Partiendo de una lista de metodologías agiles con RUP, SCRUM y XP, se presenta la tabla 1 con un estudio que diferencia las metodologías evaluadas para realizar este proyecto:

	RUP	SCRUM	XP
CICLO	Cada ciclo tiene 4 fases, pero algunos flujos pueden realizarse simultáneamente.	Se mide en sprints (iteraciones) de entre dos y cuatro semanas.	Solo un ciclo de vida muy corto para proyectos muy rápidos.
PLAN	Se planifica todo desde un inicio hasta la fecha final, cuenta con hitos intermedios y posee varios ciclos o iteraciones.	Cada plan de la siguiente iteración se planifica al final de la interacción actual dado que depende de lo conseguido. Cada sprint conlleva una reunión y análisis del proyecto.	Programación rápida y extrema para una mayor velocidad en el lanzamiento de los proyectos.

ALCANCE	Proyectos en los que se tiene muy claro el alcance de lo que se requiere hacer. Pueden ser revisados durante el proyecto, aunque no suele ser muy común y es un procedimiento estrictamente controlado.	Se tiene claro a donde se tiene que llegar, y cada sprint es un paso más cerca para llegar, pero por el camino pueden surgir nuevos requisitos o problemas muy fáciles de corregir.	Muy usada en desarrollo de software y de las más exitosas en la actualidad.
TIPO DE PROYECTO	Proyectos grandes y a largo plazo en el que hay fechas fijadas importantes que no pueden variar (o pueden hacerlo muy poco).	Grandes proyecto con visión de futuro para el que no se conocen todos los detalles desde el primer momento.	Proyectos cortos y rápidos en equipos de tamaño reducido.

Tabla 1: Tabla comparativa entre RUP, SCRUM y XP (SumariMiguel)

Como se puede ver en la tabla, XP se encuentra descartada porque el proyecto es a largo plazo (un año) y no solo incluye software, tambien hay hardware y análisis de productos.

Entre RUP y SCRUM era difícil elegir. Para RUP, el tener una fecha fijada (entrega del PFG) y conociendo los requisitos desde el principio podría encajar bien, pero sopesando eso con la capacidad que te brinda SCRUM para poder analizar y gestionar todos los problemas del desarrollo cada iteración, y siendo un proyecto con cierta base en investigación y aprendizaje personal fue considerada como la mejor opción. Así mismo, pese a que el proyecto tiene una fecha de entrega concreta, el margen de tiempo para acabarlo era amplio.

El total del proyecto ha sido dividido en diversas tareas independientes, siendo algunas de ellas necesarias para poder completar las siguientes. Debido a esto, las tareas iniciales fueron marcadas como alta prioridad, mientras que las que dependían de las anteriores se marcaron como prioridad media o baja.

A cada tarea fue asignado un espacio de tiempo de 1, 2 o 3 semanas, y el conjunto de una o varias tareas indicaban las entregas.

El trabajo de cada día consistía en analizar los resultados y el trabajo del día anterior y proseguir con los cambios necesarios para completar esa tarea de forma satisfactoria. Cada semana revisaba el trabajo de la semana anterior y modificaba los plazos a razón de si había cumplido las tareas o había tenido algún problema durante el desarrollo provocando algún retraso.

Las tareas iniciales fueron separadas en una lista según temática, donde la asignación de tiempo corresponde a lo indicado en la tabla 2:

ID	Tarea	Horas	Semanas	Prioridad
1	Scrum	20	1	Top
2	Montaje básico	20	1	Top
3	Montaje avanzado	20	1	Top
5	Control remoto básico (inicio app) + robot	40	2	Top
8	Presupuesto, Scrum inicial, primeros problemas y antecedentes	40	2	Mid
4	Montaje autónomo	20	1	Mid

6	App avanzada (Lector de PDF y secuencias) + robot	40	2	Mid
7	Crear los Tutoriales y PDF	20	1	Low
9	Finalizar memoria	80	4	Low
		300	15	

Tabla 2: Tiempo asignado a cada conjunto de tareas

Se incluye un tablero de tareas organizado entre nuevas, en curso, completadas y en espera como se puede ver en la figura 7, para poder tener una gestión de lo realizado y lo que falta por realizar en un vistazo rápido.

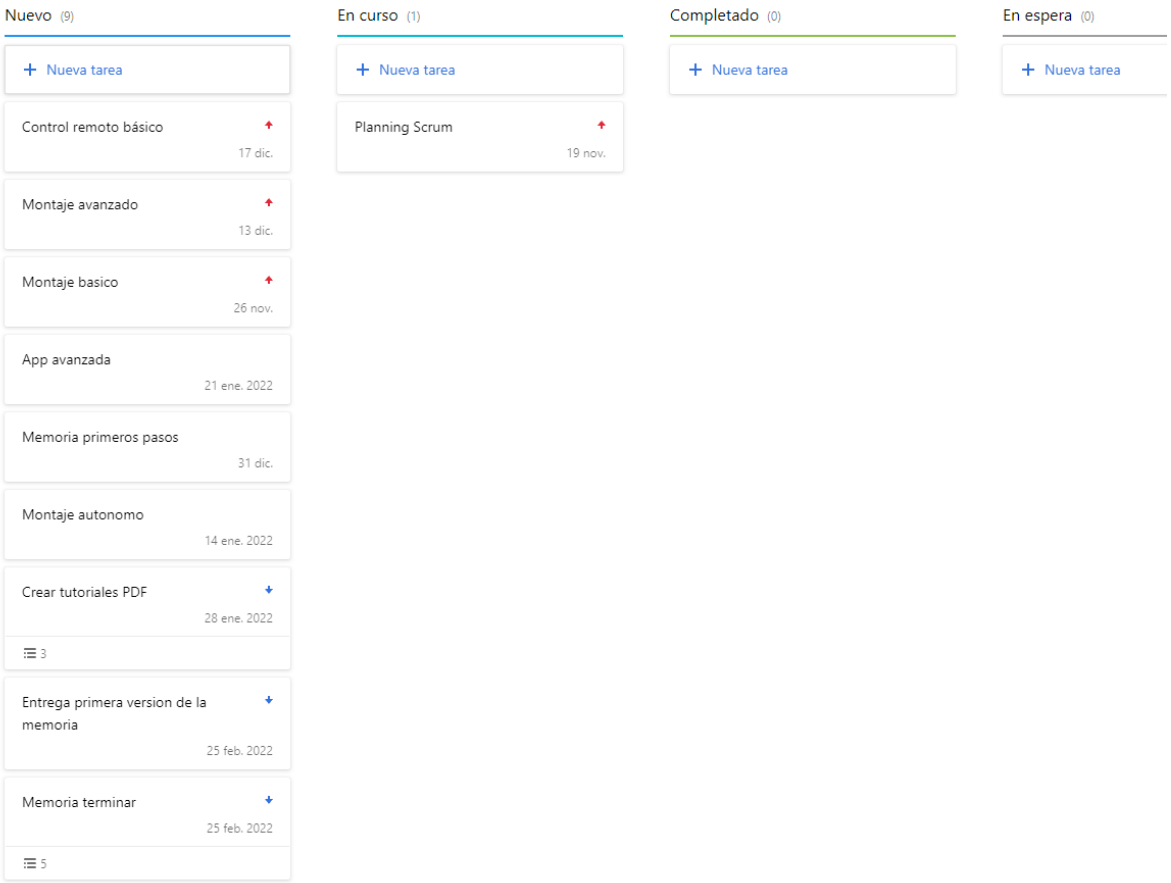

Figura 7: Tablero de tareas pendientes, en curso y terminadas

4.1 Planificación inicial

4.1.1. Primera entrega (19/11/2021)

- Planificación

4.1.2. Segunda entrega (17/12/2021)

- Montaje básico. Consiste en el montaje del cuerpo principal con los sistemas necesarios para poder moverse tales como el Arduino, la controladora, los motores y la alimentación (calibración incluida).
- Montaje avanzado. Se incluyen todos los sensores, luces, pantalla, buzzer y el módulo bluetooth.
- Control remoto básico. Se implementa una pequeña aplicación con los botones necesarios para comprobar el correcto funcionamiento de la conexión bluetooth.

4.1.3. Tercera entrega (31/12/2021)

- Memoria
    - Presupuesto
    - Scrum (inicio)
    - Primeros problemas
    - Antecedentes
    - Introducción
    - Diseño y desarrollo del prototipo

4.1.4. Cuarta entrega (28/01/2022)

- Robot autónomo. Incluye la batería para poder manejar el robot sin estar enchufado a nada externo.
- Aplicación completa. Con todos los menús para instrucciones, pantalla de secuencias y de control remoto.
- Instrucciones. Se diseñan unos pequeños PDFs con el montaje de cada uno de los kits.

4.1.5. Quinta entrega (25/02/2022)

- Memoria completa

La distribución del tiempo a lo largo de los primeros meses se ve representada en las figuras 8 y 9, teniendo en cuenta que algunas tareas dependen de otras.

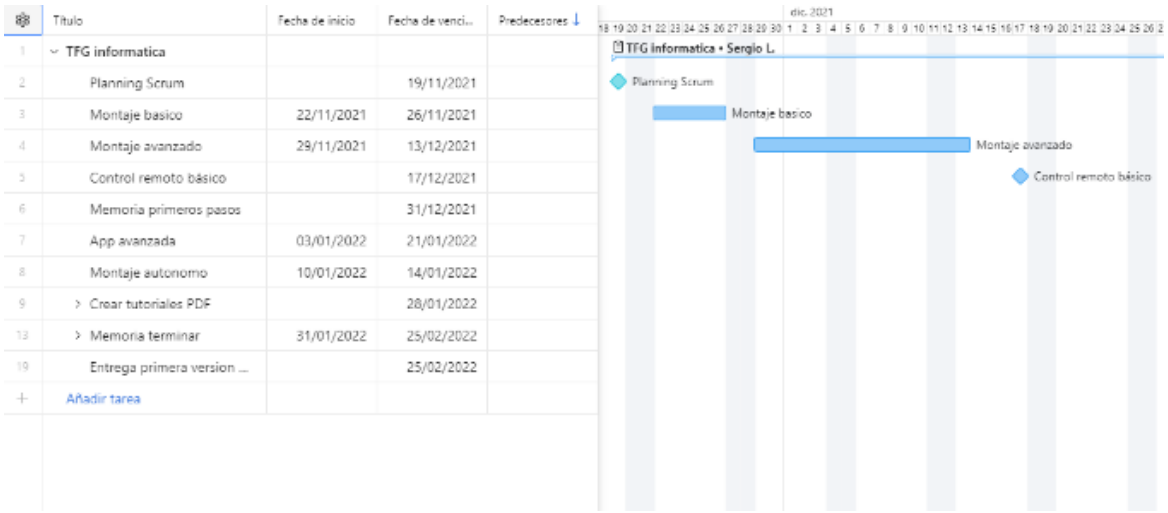

Figura 8: Distribución de tareas a lo largo del tiempo 1

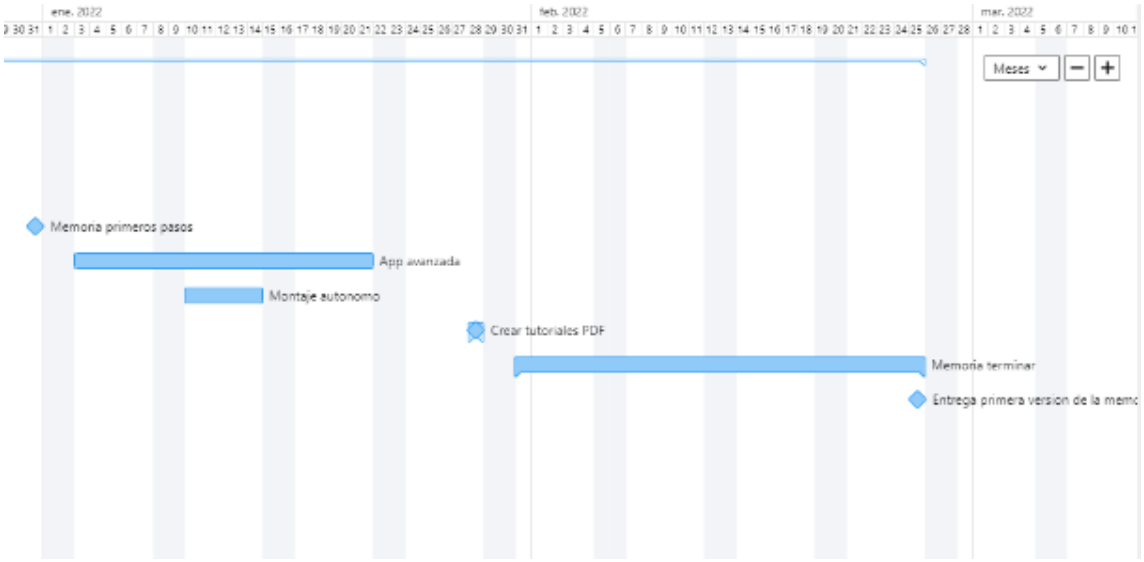

Figura 9: Distribución de tareas a lo largo del tiempo 2

Como se puede ver, el primer bloque consiste en el montaje básico, montaje avanzado, y el control remoto básico, el segundo bloque redactar algunos aspectos de la memoria, el tercer bloque la app avanzada y el montaje autónomo junto con los tutoriales en PDF y el ultimo bloque terminar la memoria y entregarla para correcciones. Las fechas de cada entrega y bloque corresponden a las anteriormente indicadas.

5. Diseño y desarrollo del prototipo y los kits

5.1 Fabricación Spot Micro

Esta es la sección más amplia del documento en el que se explica cómo está construido el robot junto con la aplicación y los problemas que se han encontrado durante el desarrollo con sus posibles soluciones y mejoras. Se presentan las diferentes secciones separadas por áreas de conocimiento en vez del proceso de desarrollo indicado en la planificación dado que es más sencillo de comprender cada área en conjunto que tenerlas separadas. En la figura 10 se puede ver el modelo 3D del robot que tendrá una vez acabado. Para realizar esta tarea se preparó todo el equipo necesario como se puede ver en la figura 11.

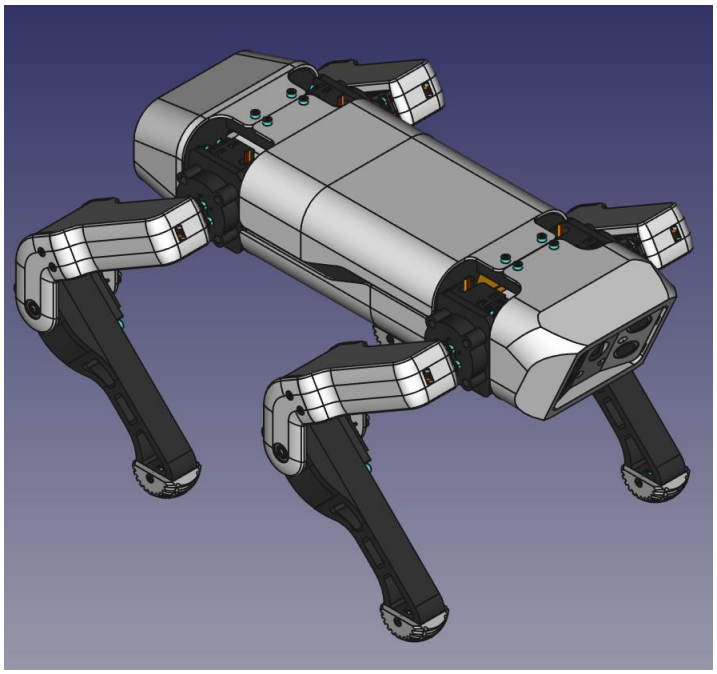

Figura 10: Modelo 3D de spot micro

Figura 11: Entorno de desarrollo del proyecto.

5.1.1. Estructura

Todo comienza con un modelo 3D del robot. La comunidad Spot Micro ha conseguido adaptar el modelo de Spot Mini original en un robot más reducido de tamaño y capaz de encajar diversos componentes electrónicos y mecánicos de tamaños estándar. En concreto, el modelo que se eligió fue el de (Kubina, s.f.) por los motivos indicados al final del capítulo 2, Antecedentes.

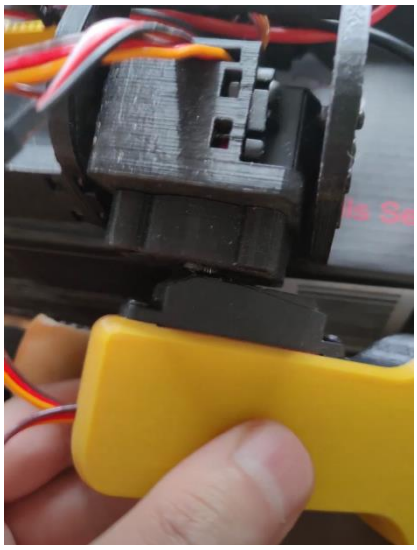

Figura 12: Eje del hombro/cadera

La estructura principal consiste en cuatro patas, simétricas dos a dos. Poseen almohadillas para no resbalarse en el suelo. La parte baja de las patas tiene un hueco para insertar un servo motor estándar. Dos piezas, una de ellas con soporte para engranaje plástico y otra para rodamiento unen la parte baja de la pata con la parte alta, con hueco para otro servo motor que actuará como un eje del hombro.

El cuerpo principal del robot está formado por la parte central, donde va toda la electrónica una parte frontal y otra trasera donde van varios sensores, ranuras de carga e interruptores y cuatro articulaciones. Estas cuatro articulaciones encajan un servo motor para el segundo eje del hombro, que junto a los motores de la parte alta de las patas hacen que puedan rotar en dos ejes diferentes.

Pese a que este modelo inicial es integro conseguido en GitHub, se realizaron diversas modificaciones al cuerpo para hacerlo más largo y así disponer de espacio extra a la hora de encajar los diferentes sensores en su interior. De la misma manera, fue necesaria la posterior modificación de varias juntas para encajar unos engranajes diferentes a los diseñados originalmente. Los motivos de este segundo cambio de engranajes se explica más en profundidad en la sección de problemas durante el desarrollo.

Una vez todas las piezas estaban impresas, se comenzó el proceso de montaje.

Figura 13: Pata trasera

Primero vienen las patas. Cada una cuenta con siete piezas de plástico impresas en 3D, varios tornillos y tuercas, dos servomotores DS3218 PRO, un rodamiento 625RS y un disco de metal T25 como eje del motor. Se puede ver una de las patas en la figura 12.

Cada uno de estos motores utiliza tres cables. El primero de ellos, naranja, lleva una señal PWM que indica el ángulo del motor.

El cable rojo y marron son +6`7V y GND.

El rodamiento permite un movimiento más libre de la articulación con el menor rozamiento posible y aporta resistencia y estabilidad a este giro. En la figura 13 se puede ver una imagen de la pata trasera.

Fue necesaria la modificación del soporte del disco de metal dado que el modelo original estaba preparado para funcionar con unos

soportes de plástico en estrella. Al ser de plástico los dientes del engranaje se quemaban y perdían agarre, por lo que se cambió a unos discos de metal que no se dañaban tan fácilmente con la fricción.

El cuerpo del robot se compone de tres secciones y se presentan cada una de ellas a continuación: la cabeza, el tórax y el abdomen.

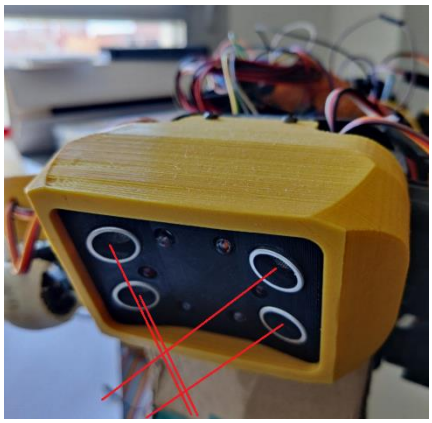

Figura 14: Frontal con sensores de ultrasonido y leds

La cabeza consta de 3 piezas impresas en 3D, que sujetan seis LEDs (Light Emitting Diode) dos de ellos rojos que se encienden cuando el robot detecta una pared o superficie frente a él y dos sensores HC-SR04 de distancia enfrentados en diagonal gracias a los cuales puede reconocer si la pared la tiene a la derecha o a la izquierda, como se puede ver en la figura 14.

Estos sensores HC-SR04 funcionan enviando ultrasonidos que rebotan en la pared que tenga delante y son recibidos por el mismo sensor, que mide el tiempo que tarda el pulso en llegar y calcula el tiempo dada la velocidad del sonido. A nivel de implementación, para que estos dos sensores no se pisen entre ellos (ambos usan la misma frecuencia de sonido) hay que dejar un tiempo prudencial entre ambos pulsos.

Está orientado hacia abajo para poder alumbrar la superficie por la que va a pasar y en el caso de incorporar una cámara poder sacar imagen RGB con condiciones de iluminación decentes.

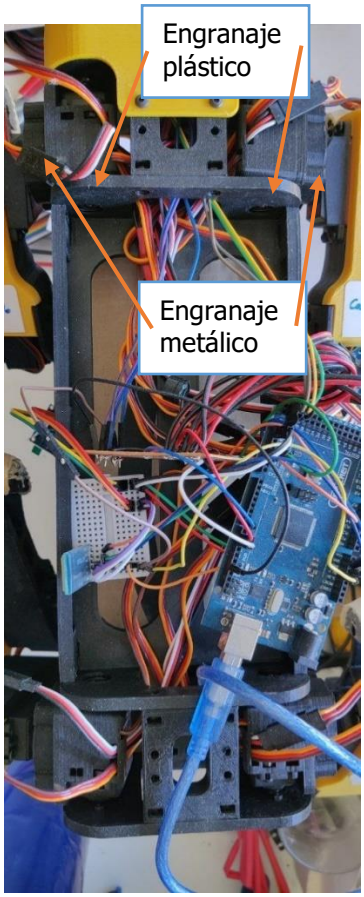

Figura 15: Vista superior

El tórax es la parte más compleja de la estructura. En él está contenida toda la electrónica al igual que las articulaciones donde las patas se unen al cuerpo.

El diseño del tórax no es exactamente igual al modelo inicial que se decidió usar para esta implementación, ha sido modificado para alargarlo y hacer la cavidad interna más grande para poder alojar más electrónica y circuitos.

Contiene cuatro articulaciones iguales que se dividen en tres piezas 3D, junto a un servomotor, un rodamiento, un engranaje de metal y uno de plástico cada articulación como se ve indicado en la figura 15. La estructura que sujeta todo suman 11 piezas más de plástico 3D.

El motivo por el que todavía se mantienen los engranajes de plástico en un eje de los "hombros" es que ese eje no está sometido al mismo peso ni esfuerzo que el otro eje del hombro y el codo, por lo que el fallo es mucho menos probable.

5.1.2. Electrónica y circuitos

El circuito ha sido diseñado para hacer que sus componentes funcionen en las condiciones más optimas, teniendo en cuenta los voltajes y corriente de cada uno de sus componentes. Posee tambien varios elementos de seguridad para detectar fallos del sistema que podrían desencadenar en el robot ardiendo y quemarse.

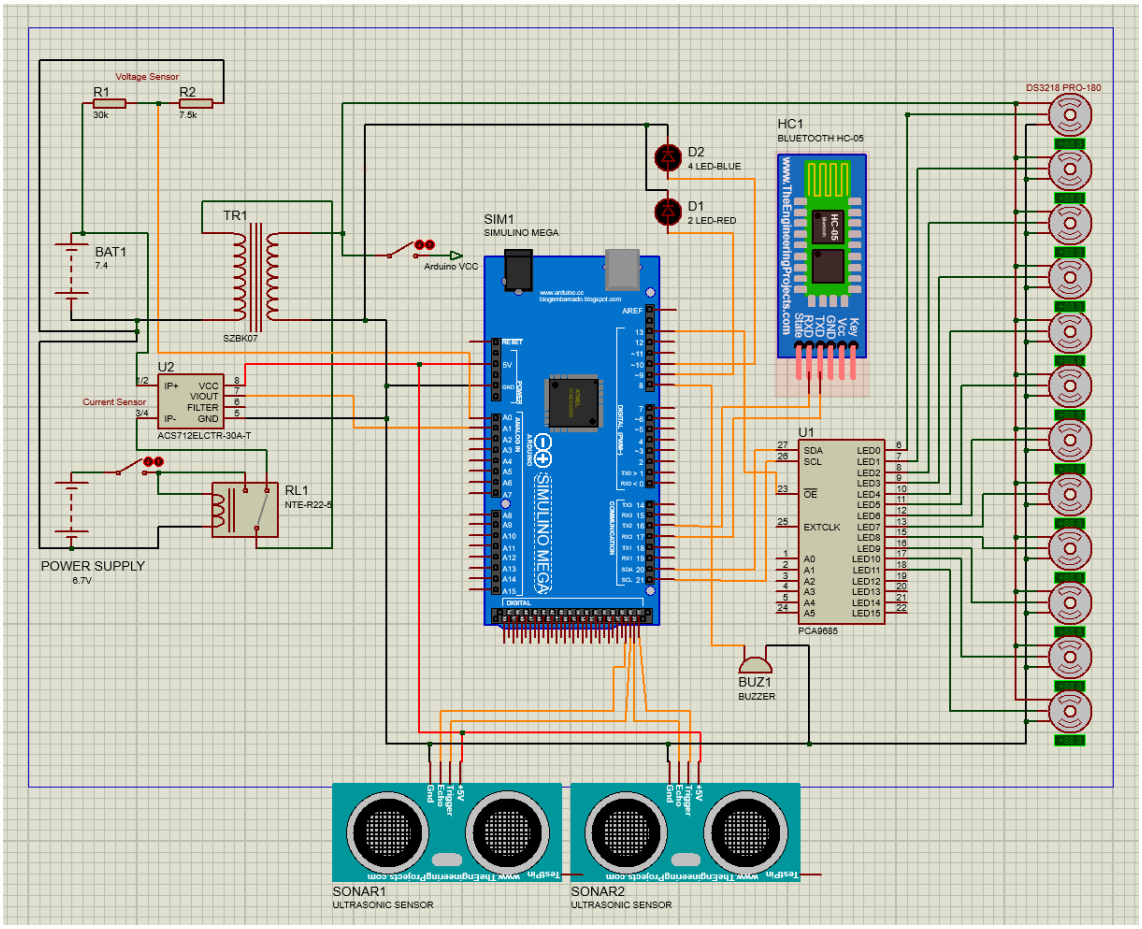

Figura 16: Esquema eléctrico

Mediante el software de simulación de circuitos "Proteus" se realizó un esquema de la electrónica (figura 16) del robot, donde se pueden ver los diferentes circuitos donde los cables verdes conducen 6'8 Voltios, los cables naranjas llevan 5 Voltios o señales digitales y los cables negros son el GND.

El circuito consiste en 4 sistemas llamados sistema de alimentación, sistema de control, sistema de visualización y sistema mecánico.

La elección de estos componentes eléctricos y electrónicos se basan principalmente en poseer las especificaciones técnicas necesarias para el correcto funcionamiento del sistema. Cuando hay varios dispositivos disponibles capaces de hacer lo mismo, se elige el más sencillo y cómodo de utilizar, barato y fiable. Tambien es un buen punto a favor que sea fácil de conseguir.

El **sistema de alimentación** está representado en la parte izquierda de la figura 10 y se compone de:

- Una batería de litio 2S con un voltaje nominal de 7'4 Voltios y 5200mAh, la cual alimentará el circuito cuando no haya corriente externa (BAT1).

- Una fuente de alimentación de laboratorio mediante la que se puede regular el voltaje a las necesidades del usuario. Las pruebas han sido realizadas con 6'7 Voltios, pero actualmente se sitúa en 9 Voltios debido a un fallo en el diseño del que no permitía hacer actuar el relé de alimentación.

Figura 17: Divisor de voltaje

- Un sensor de voltaje (figura 17), también llamado voltímetro DC0-25V (contiene un divisor de voltaje básico). Una batería de litio muestra su nivel de carga según el voltaje que es capaz de dar. Conociendo el voltaje es posible implementar un indicador de nivel de carga que además permita desconectar el sistema cuando la batería este muy baja evitando que se dañe conociendo los datos indicados en la figura 18.

State Of Charge vs. Lipoly Pack Voltage

% Capacity	1S Cell	2S Pack	3S Pack	4S Pack	5S Pack	6S Pack
100	4.20	8.40	12.60	16.80	21.00	25.20
95	4.15	8.30	12.45	16.60	20.75	24.90
90	4.11	8.22	12.33	16.45	20.56	24.67
85	4.08	8.16	12.25	16.33	20.41	24.49
80	4.02	8.05	12.07	16.09	20.11	24.14
75	3.98	7.97	11.95	15.93	19.92	23.90
70	3.95	7.91	11.86	15.81	19.77	23.72
65	3.91	7.83	11.74	15.66	19.57	23.48
60	3.87	7.75	11.62	15.50	19.37	23.25
55	3.85	7.71	11.56	15.42	19.27	23.13
50	3.84	7.67	11.51	15.34	19.18	23.01
45	3.82	7.63	11.45	15.26	19.08	22.89
40	3.80	7.59	11.39	15.18	18.98	22.77
35	3.79	7.57	11.36	15.14	18.93	22.72
30	3.77	7.53	11.30	15.06	18.83	22.60
25	3.75	7.49	11.24	14.99	18.73	22.48
20	3.73	7.45	11.18	14.91	18.63	22.36
15	3.71	7.41	11.12	14.83	18.54	22.24
10	3.69	7.37	11.06	14.75	18.44	22.12
5	3.61	7.22	10.83	14.43	18.04	21.65
0	3.27	6.55	9.82	13.09	16.37	19.64

Stay in the white region for maximum pack longevity

Figura 18: Tabla relacional entre voltaje de la batería y su nivel de carga (Ampow, s.f.)

- Un sensor de amperaje o amperímetro ACS712 (Figura 119). Los motores consumen aproximadamente una corriente de 2A cada uno en pico (cuando más carga soportan). Hay 12 motores en el sistema de los cuales se calcula que solo 4 de ellos pueden llegar a estar en pico al mismo tiempo, donde el resto estarán consumiendo, pero por debajo de los 0'5A cada uno (despreciable).

Figura 19: Amperímetro ACS712

El resto de los componentes del circuito consumen una corriente despreciable, lo que hace que el sistema de alimentación necesite estar preparado para aguantar 15A aproximadamente. Mediante el amperímetro se puede controlar en todo momento el consumo de corriente del circuito y detectar cortocircuitos o consumos excesivos y bloquear el sistema como medida de seguridad.

- Un relé de coche RLP12V (figura 20). Se diseñó el circuito para que pudiese tener simultáneamente conectada la batería y la alimentación externa. Para ello es usado un relé de intermitentes de un coche capaz de trabajar hasta con 12 voltios y 40 amperios, donde al enchufar la alimentación externa se activa y cierra el circuito de alimentación para esa alimentación externa abriendo el circuito de la batería. Cuando el cable externo se desconecta se cierra el circuito para la batería de nuevo y puede seguir funcionando sin cables.

Figura 20: Relé

- Transformador "Buck Converter" SZBK07 (figura 21). Los motores funcionan a un voltaje de 6'8V como máximo por lo que era necesario reducir el voltaje de la batería y la fuente externa a ese número. El Arduino deja alimentarlo entre 6V y 20V, lleva su propio regulador por lo que no había problema con él. La elección de este transformador es que permite 20A (frente a otros modelos que apenas llegan a los 8A) necesarios para que todos los motores puedan funcionar de la forma correcta. La posición de este transformador en el circuito de alimentación fue cambiando durante la implementación debido a problemas encontrados relacionados con la activación del relé por parte de la alimentación externa (*véase la sección de problemas*).

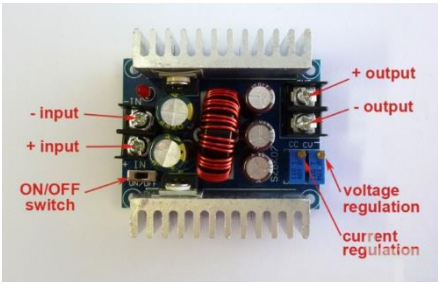

Figura 21: Transformador SZBK07

De este sistema de alimentación se consiguen sacar 6'7 voltios estables para alimentar el Arduino y los motores, pero tambien son necesarios 5V para alimentar el resto de los componentes electrónicos que requiere el sistema. Estos 5V son transformados por el propio Arduino, cuya corriente no es muy alta, pero si suficiente para alimentarlos.

El **sistema mecánico** se compone de los servomotores y su placa controladora:

- Controladora PWM PCA9685 (figura 22). Utilizar unos pocos motores con un Arduino es tan sencillo como conectarlos directamente a uno de los pocos puertos que tiene PWM (pulse width modulation). Cuando se tienen 12 motores, el Arduino no tiene suficientes salidas PWM y se necesita una placa extra que controle todos los PWM como la PCA9685 que se está usando (16 salidas). Existen pocas placas en el mercado tan fiables y fáciles de conseguir como ésta, por lo que se decidió usarla. Esta placa originalmente se usaba para controlar el brillo

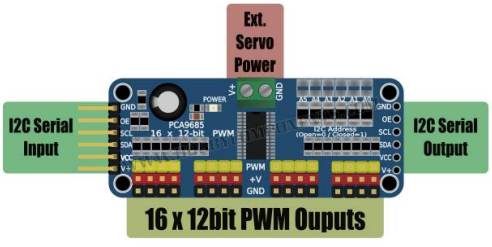

Figura 22: Controladora PWM PCA9685

de LEDs, pero con el tiempo han ido saliendo librerías para cualquier dispositivo que necesitase ser controlado por una señal PWM.

Esta placa solo podía alimentar a los motores con 5V, por lo que hubo que modificarla para funcionar con los 6'7V que se requiere (véase la sección de problemas).

Esta placa se conecta al Arduino mediante una comunicación de bus I2C (SDA-SCL). (RootSaid, s.f.)

- Servo motores DS3218 PRO (figura 23). 12 de ellos, rotación de 0 a 270 grados con características (figura 24):

Brand	Dsservo	
Product Name	DS3218	DS3218 PRO
Torque	18kg. cm@5V	21kg. cm@5V
	21. 5kg. cm@6. 8V	23. 5kg. cm@6. 8V
Speed	0. 16S/60° @5V	0. 12S/60° @5V
	0. 14S/60° @6. 8V	0. 09S/60° @6. 8V
	recommend	strongly recommend

Figura 24: Especificaciones servomotores

Figura 23: Servomotor DS3218 PRO

Su máximo voltaje es 6'8V pero son usados a 6'7V para evitar fallos a largo plazo y ser más seguros. Tienen una velocidad y fuerza dependiente del voltaje, lo que trae muchos problemas a la hora de regularlos como se requiere (*véase la sección de problemas*). Las especificaciones del motor son:

  - Frecuencia de funcionamiento: 50-333Hz
  - Recorrido de funcionamiento: 270 °(PWM 500-2500µs)
    (Mechatronics, Tutorial uso de servomotores con arduino, s.f.)

El **sistema de visualización** contiene una pantalla OLED para datos, leds como indicadores visuales y un zumbador para indicaciones sonoras.

Figura 25: Buzzer

- Un zumbador o altavoz (figura 25) que pita cada vez que sucede algo en el sistema. Similar a los pitidos que realizaban los ordenadores antiguamente cuando arrancaban.

Figura 26: Pantalla OLED SSD1306

- Una pantalla OLED SSD1306 (figura 26) conectada por I2C (al igual que la PCA9685) en la cual se puede ver información útil como la batería restante, la acción que está realizando (modo debug), indicador de ángulo de choque frontal, posibles fallos en el circuito (posible futura implementación) ...

- 6 LEDs, dos de ellos rojos y cuatro azules. Los azules están permanentemente encendidos, alumbrando el frente y suelo mientras que los rojos se encienden cuando detecta una pared o superficie delante suya.

Por último, el **sistema de control** se compone de la controladora principal Arduino Mega, un módulo bluetooth, dos sensores de distancia por ultrasonidos y la propia PCA9685 (vista anteriormente).

- Arduino Mega (figura 27): En un inicio estaba planeado el uso de un Arduino Uno, la opción más básica y conocida de las controladoras open source Arduino. Durante el desarrollo se encontraron problemas de memoria disponible para la gran cantidad de

librerías , variables y código en general fue cambiada a un Arduino Mega, con mucha más potencia y conectividad (figura 28).

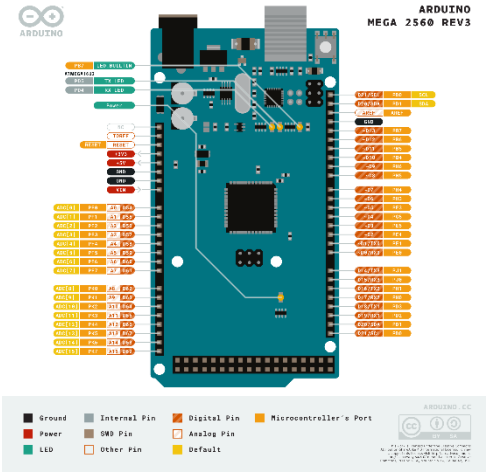

Figura 27: Esquema de conexiones Arduino Mega

MICROCONTROLLER	ATmega2560
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	54 (of which 15 provide PWM output)
ANALOG INPUT PINS	16
DC CURRENT PER I/O PIN	20 mA
DC CURRENT FOR 3.3V PIN	50 mA
FLASH MEMORY	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
CLOCK SPEED	16 MHz
LED_BUILTIN	13
LENGTH	101.52 mm
WIDTH	53.3 mm
WEIGHT	37 g

Figura 28: Especificaciones técnicas Arduino Mega

Las anteriores fotos contienen un diagrama de las salidas que posee la placa y las especificaciones técnicas de ella.
Pese a haber placas con mejores características como las ESP32, jetson nano de Nvidia y las muy conocidas raspberry pi, la elección de esta placa viene dada porque es barata, fiable y sencilla de usar y suficiente para las tareas que requiere el producto final.

- Modulo bluetooth HC-05 (figura 29). Conectividad por puerto serie (TX-RX). Este puerto serie utiliza la misma comunicación que los USB 2.0 de los ordenadores, por lo que puede ser usado tanto en el ordenador como en placas como Arduino, raspberry…
  Existen dos modelos, el HC-05 y HC-06. La diferencia entre ambos es que el HC-05 puede funcionar tanto como dispositivo maestro como esclavo, mientras que el HC-06 solo puede actuar como esclavo. Para este proyecto es necesario que esté configurado como esclavo para que el móvil pueda conectarse a él. (Mechatronics, Configuracion del modulo bluetooth HC-05 usando comandos AT, s.f.)

Figura 29: Bluetooth HC-05

- Sensores de distancia por ultrasonidos HC-SR04 (figura 30), dos de ellos capaces de medir distancias entre 2 y 450 cm. Consisten en un altavoz el cual emite un ultrasonido a 40KHz y un micrófono que recibe el rebote del sonido, con el que calcula la distancia dependiente del tiempo entre la emisión y recepción.

Figura 30: Ultrasonidos HC-SR04

5.1.3. Comunicaciones

Todos estos sistemas están conectados entre ellos por cuatro tipos de conexiones y señales: señales digitales, PWM, I2C, Serial y Bluetooth.

- Las señales digitales llevan un nivel de señal 0 o 5V en Arduino (0 o 3'3V en otras placas). Este tipo de señales únicamente tienen dos estados, encendido o apagado, y mediante el cambio de estos dos estados se puede transmitir información.
  Las ventajas de estas señales sobre las analógicas es que puede ser amplificada y reconstruida sin pérdida de información infinitas veces, son fáciles de corregir errores en la transmisión, su procesamiento es muy sencillo y se ven menos afectadas que las analógicas ante ruido eléctrico externo.

- Las señales PWM (figura 31) son la solución que se da ante la necesidad de mandar una señal analógica desde un chip que no puede regular el voltaje.

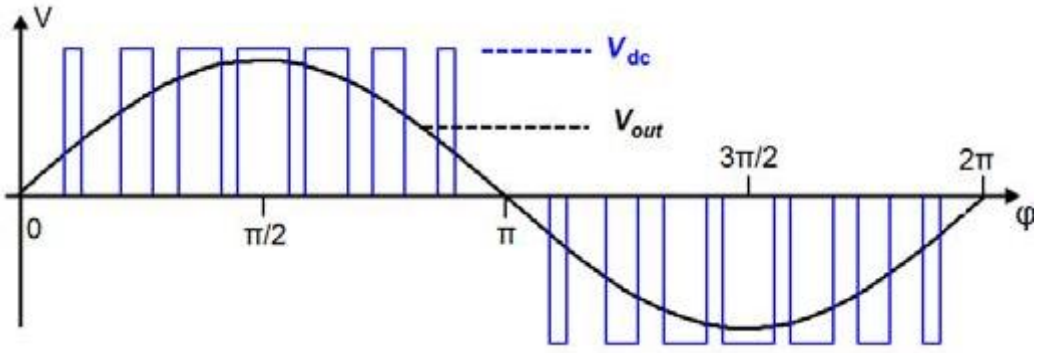

Figura 31: Funcionamiento de una señal PWM

PWM (pulse width modulation) consiste en una señal digital enviada a pulsos de diferentes tiempos, con 256 bits de precisión. Este tipo de señales pueden ser usados tanto para cambiar voltajes como salida "analógica" entre 0 y 5V o para enviar pulsos estables TTL con pico de 5V. La señal o dato que hay que enviarle al servo es una señal de PWM donde el tiempo en alto es equivalente al ángulo o posición del servo (figura 32). Estos valores pueden variar y van desde 0.5 a 1 milisegundo para la posición 0º (0'5ms en este caso) y 2 a 2.5 milisegundos para la posición de 180 – 270º (2'5ms en este caso), el periodo de la señal debe ser entre 15 y 25 milisegundos. La posterior ilustración es del equivalente a un servo motor con rango 180º pero en nuestro caso son 0º, 135º y 270º

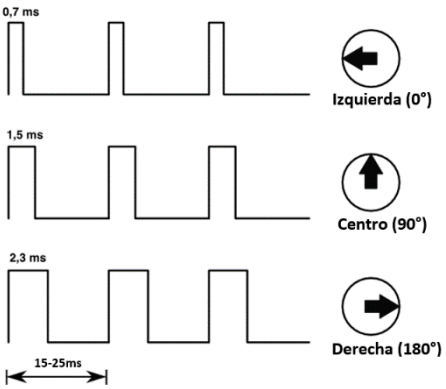

Figura 32: Relación entre una señal PWM y giro del motor

- El bus I2C (Inter-Integrated Circuit) fue desarrollado por Philips en los años 80 para sus dispositivos electrónicos. Posteriormente fue adaptado por otros fabricantes hasta convertirse en un estándar.
  El bus requiere únicamente dos cables (figura 33), uno para las señal del reloj y sincronización (CLK) y el otro para el envío de Datos (SDA). En este bus, cada dispositivo tiene una dirección que se emplea para reconocer y acceder a ese dispositivo de forma individual, la cual puede ser fijada por hardware (las placas vienen con una dirección

prefijada pero varios jumpers para poder modificarla o juntar varias placas iguales con direcciones diferentes) o por software. Las direcciones deben ser únicas y poseen una arquitectura maestro-esclavo, donde el maestro inicia la comunicación y permite enviar o recibir datos de los esclavos. Los esclavos tampoco pueden hablar entre ellos. Se permiten varios maestros conectados a la red, pero solo uno puede estar definido como maestro al mismo tiempo.
Es un bus síncrono (CLK) lo que evita que cada dispositivo necesite un reloj y acordar velocidades de transmisión.
Su funcionamiento consiste en una señal de 7 bits de dirección, un bit de envío-recepción, un bit de validación, uno o más bytes de datos y un último bit de validación. (Llamas, El bus I2C en arduino, s.f.)

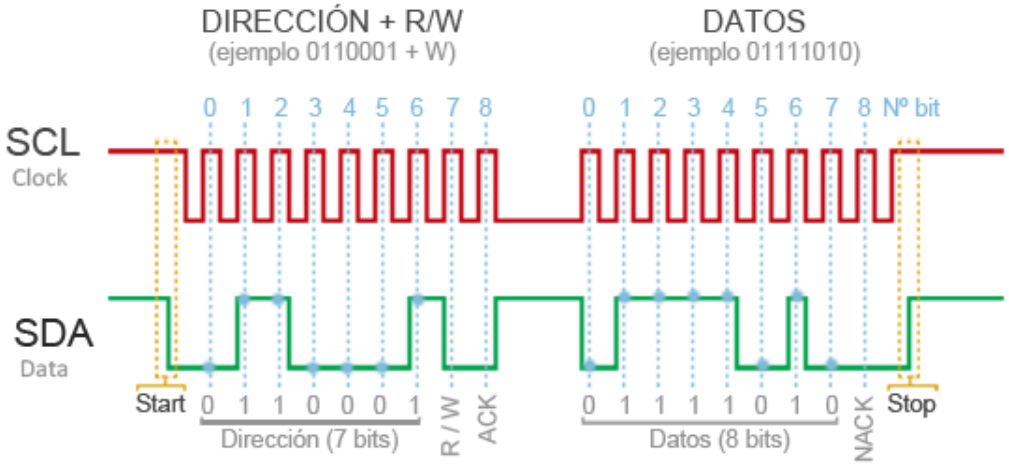

Figura 33: Protocolo I2C

Con 7 bits es posible direccionar 112 dispositivos en un mismo bus, donde las últimas 16 direcciones para llegar a las 128 se reservan para usos especiales. Se usan 18 bits por cada 8 bits de datos, lo que hace que la velocidad de bus sea reducida (100Khz).
Para Arduino existen multitud de librerías, aunque la más utilizada es "Wire". En el Arduino Mega los pines habilitados para esta comunicación son el 20 para SDA y 21 para SCL.

- Conexión en Serie (Serial). Los puertos serie son la forma de comunicación principal entre el Arduino y el ordenador. Igualmente, el puerto serie es el uso actual para ratones, teclados, pendrives… Existen varias versiones de la conectividad USB (Universal serial bus) como el USB 1.1, 2.0, 3.0, 3.1… dependiendo de la cantidad de datos que puede transmitir, corriente que puede mover, cables internos…
  El Arduino utiliza conectividad USB 2.0, que consiste en 4 cables. El primero lleva una señal positiva estable de 5V, con el que alimenta el dispositivo. El segundo cable lleva una señal de GND o común. Los otros dos son TX (transmisión) y RX (recepción). En la figura 34 se puede ver la diferencia entre comunicación en serie y en paralelo (Llamas, Comunicacion de arduino con puerto serie, s.f.)

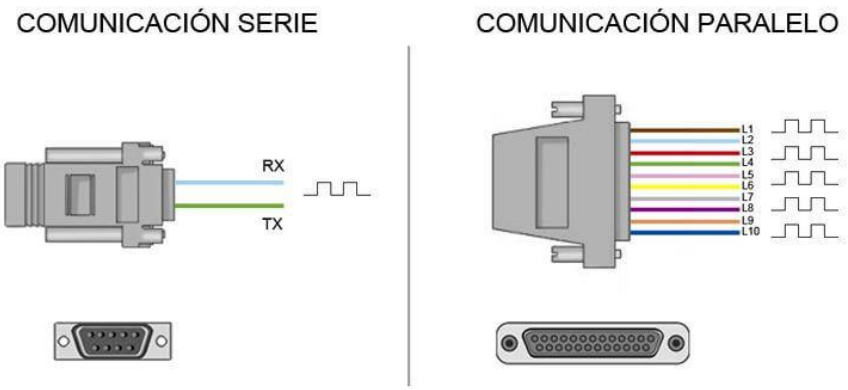

Figura 34: Conexión en serie contra paralelo

Mientras que el Arduino uno (y la mayoría de Arduinos) solo contienen una unidad UART (universally asynchronous reciver/transmitter) cuya función es convertir datos a una secuencia de bits y transmitirlos o recibirlos a una velocidad prefijada. En el caso del Arduino Mega, éste posee tres puertos serie para conectividad de módulos externos donde uno de ellos coincide físicamente (los pines están conectados) con el cable con el que se conecta al ordenador. Podemos abrir en el ordenador monitores de puerto serie y enviar o recibir información.
Mediante esta conexión se hace funcionar el módulo bluetooth HC-05, ya que el chip está preparado para funcionar tanto en formato "pendrive" con la conexión USB clásica como los pines por separado para usarlo.

- Bluetooth. Un chip bluetooth de clase 2 conecta el robot con la aplicación móvil. La comunicación se realiza siendo el móvil maestro y el HC-05 conectándose a él. Los datos transmitidos son unos comandos enviados desde una terminal bluetooth serie o la aplicación y son recibidos e interpretados por el robot como si de un puerto serie cableado se tratase. Esta comunicación sucede a una velocidad de 9600 baudios, aunque depende del módulo que se esté usando y cómo esté configurado.

5.2 Programación de la placa Arduino MEGA y los sensores.

Uno de los objetivos iniciales del robot era su capacidad de andar. Para poder conseguirlo, se buscó una gran cantidad de imágenes y videos de gatos, perros y otros animales andando para observar su movimiento y simular la posición de las patas de forma similar. Una de las imágenes revisadas (figura 35) demuestra que existen 8 pasos que se repiten continuamente, los cuales se implementaron en el código dando problemas de precisión y sincronía ya que estos movimientos por parte del gato los hace a diferentes velocidades, cosa imposible de hacer con los servomotores que se usan actualmente.

La implementación de estos 8 puntos fue el inicio del código que se presenta, sobre lo que se construyó el resto del código mientras se intentaban solucionar estos problemas de precisión y sincronía, aunque lamentablemente sigue sin funcionar bien este movimiento en concreto. Se decidió, por un código más limpio, eliminar las líneas que correspondían a ese movimiento pero se añaden como anexo en un proyecto de pruebas y se han mantenido las matrices de posiciones clave para ese movimiento.

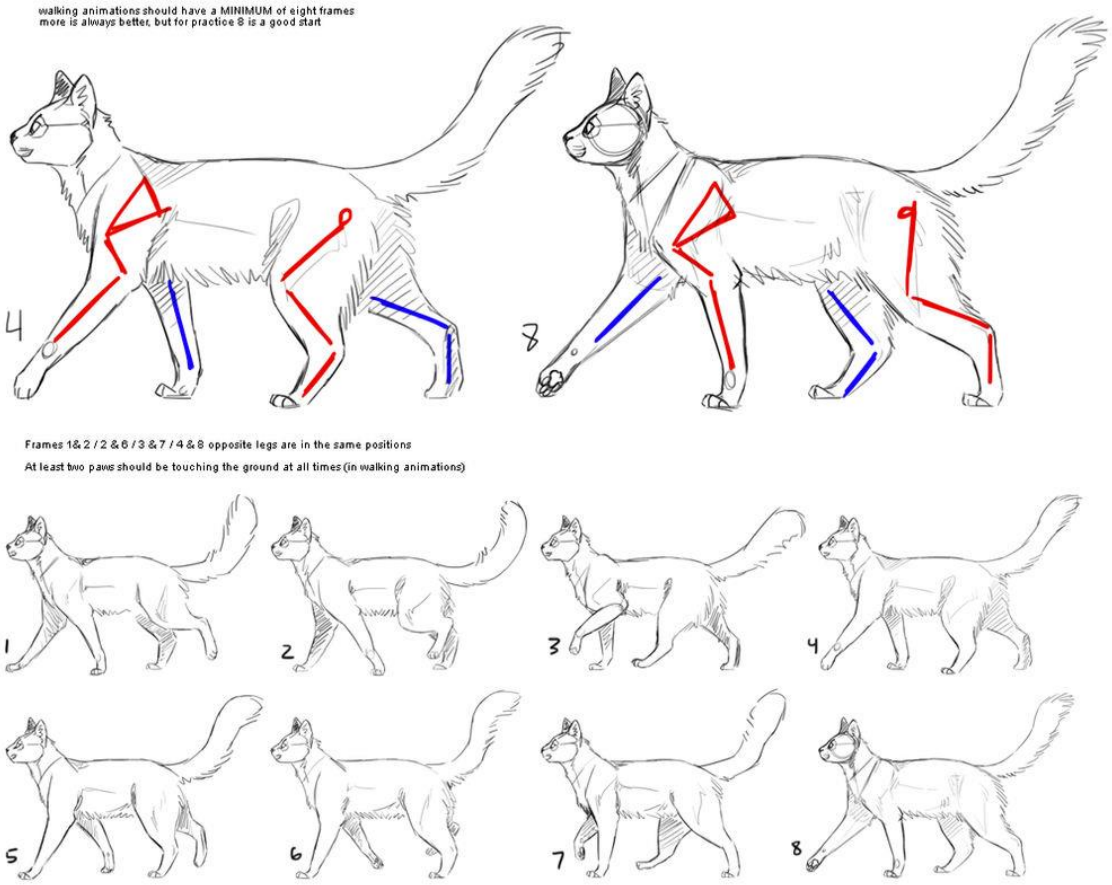

Figura 35: Pasos de un gato

(SowoD, s.f.)

La plataforma Arduino provee varios IDEs gratuitos. Entre las opciones se incluye un IDE online (Arduino, Arduino Code Online, s.f.) que guarda os proyectos en la nube y solo requiere descargar un software que conecta el USB con el proyecto en la nube. También se incluye una versión del IDE compatible con la tienda de Windows y por último la opción que se ha usado para este proyecto, un software instalable en cualquier versión de Windows desde XP (figura 36).

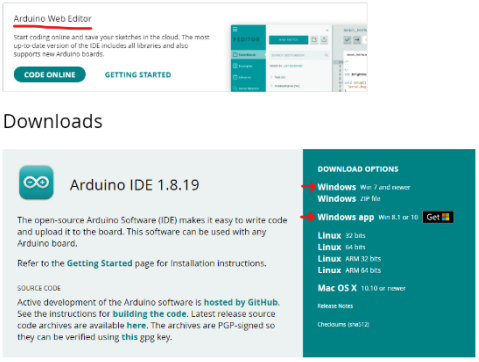

Figura 36: Opciones de descarga de Arduino

El lenguaje de programación de Arduino es C++, aunque es posible programarlo en otros lenguajes. No es un C++ puro, sino que es una adaptación que proveniente de avr-libc que provee de una librería de C de alta calidad para usar con GCC en los microcontroladores AVR de Atmel y muchas funciones específicas para los MCU AVR de Atmel.

Este proyecto se ha realizado con una placa Geekcreit Mega, versión clónica y china de la placa Arduino Mega por lo que fue necesaria la instalación de unos drivers genéricos compatibles con estas placas clónicas llamados "CH340".

Un codigo de Arduino contiene dos funciones principales, el setup() y el loop().

"void setup()" es una funcion que se ejecuta una unica vez en el encendido de la placa y su uso principal es, como su nombre lo indica, para inicializar variables, el modo de los pines (entrada o salida por ejemplo), configurar librerias en uso…

"void loop()" es la funcion del arduino que es ejecutada en bucle continuamente despues del setup().

El código está organizado de tal manera que las secciones y cada módulo vienen perfectamente identificadas para su fácil comprensión, modificación y arreglo. La primera parte consta de la declaración de variables e importación de librerías.

5.2.1 Declaración de variables

- General: En esta zona se declaran las variables que controla el refresco de la pantalla oled, el indicador de la acción, variables de control de los comandos de bluetooth y variables de gestión de las secuencias. Aquí van tambien las variables de test, debug y estado. Por último, se crea la variable que contendrá lo recibido por el bluetooth.

- Front LEDs: Se asignan los pines a los que están conectados los Leds a una variable.

- Battery, voltage and current: Se crean las variables relacionadas con la batería, voltaje y corriente y se asignan los pines correspondientes a las entradas analógicas del voltímetro y amperímetro

- Buzzer: Se asigna el pin correspondiente al zumbador.

- Ultrasonic sensor params: Se importa la librería HC-SR04 y se crean dos instancias asignándoles los pines correspondientes. Tambien se crea una variable de distancia por cada sensor.

- OLED Screen parameters: Se importa la librería Wire (I2C), se importan las librerías de bajos recursos de la pantalla OLED, se asigna la dirección I2C correspondiente a la pantalla y se crea la variable "oled".

- Servo key points: En esta sección van los arrays con las posiciones que deben tomar los motores, generados por un simulador de cinemática inversa y ajustados en el robot. En esta sección se implementan todos los grupos de movimientos relacionados con el movimiento. Actualmente solo están los grupos de andar hacia adelante con problemas de sincronización, la posición de idle (única en uso actualmente) y las posiciones que usaría una máquina de estados para pasar de estático a andar.

- PWM board and servo parameters: Se importa la librería de Adafruit PWM servo driver (Adafruit es una empresa que fabrica placas y módulos para Arduino y sus respectivas librerías compatibles con otros módulos clónicos) y se crea y asigna la variable con la que puedo interactuar con la placa. Hay que realizar una conversión del ángulo al pulso que se enviará, con lo que se definen estos parámetros para su posterior mapeado.

5.2.2 Funciones

- setup: Esta función se ejecuta una única vez cuando se enciende el microcontrolador. Aquí se inicia la comunicación serie a una velocidad de 9600 baudios tanto para el USB al ordenador como para el bluetooth e imprime por el monitor serie del pc una frase de encendido.
  Se inicia la comunicación I2C junto a la velocidad del bus y se prepara para enviar datos a la pantalla OLED. Los siguientes comandos corresponden a escribir por pantalla una serie de frases que incluyen mi nombre, "starting…" y por último cambia "starting" por "ready".
  Se configuran los pines del zumbador y leds como salidas, se inicia la comunicación con la controladora PWM pasándole valores de configuración y mueve sus motores a la posición de idle.

- fmap: Esta función se utiliza para calcular el voltaje de la batería. El sensor está preparado para trabajar de 0 a 25V, que mediante un divisor de tensión (dos resistencias) al Arduino le llega un valor entre 0 y 5V, correspondiente a un valor entre 0 y 1023. La función mapea estos valores para devolver el valor real. Se le añade un offset de -0.6 por la precisión del hardware.

- getCurrent: Recibe la información del voltaje por el pin correspondiente y con ello calcula la media de corriente en "SAMPLESNUMBER" veces. Se le aplica una sensibilidad dependiente del amperaje máximo que es capaz de leer en sensor.

- moveServoToAngle: Función principal del movimiento de los motores. A esta función se le manda el número de motor y el ángulo al que se quiere mover y junto con algunos parámetros previamente configurados, mapea el ángulo al pulso PWM correspondiente y envia esos datos a la controladora PWM para que lo genere y se lo mande al servomotor.

- Idle, stepWalk, Walk and IdleToWalk, WalkToIdle, startGreet, greet, endGreet, layDown, dance: Funciones que contienen a las acciones que puede realizar el robot. Contienen el ajuste de los motores a los ángulos correspondientes a cada acción.

- loop: Función principal, se repite en bucle durante todo el tiempo en el que está encendido el microprocesador. Contiene varias zonas bien definidas. La figura 37 muestra una guía básica para leer esta función de un vistazo rápido, seguido de la explicación de cada una de sus secciones que podrían considerarse "independientes"

```
Lee el pin del voltimetro
Mapea el voltaje

Si "bateria disponible"
    Si voltaje de bateria por encima de 7'37V:
        Da el porcentaje
    Si esta por debajo:
        Pitidos de aviso de bateria baja

Calcula la coriente
Si la corriente es mayor de 15
    Desactiva los motores (no implementado)
    Pitidos de aviso de demasiada corriente
-------------------------------------------------
Suma el tiempo de cada loop

Si el tiempo sumado (temporizador) es mayor de 5 segundos
    El temporizador se pone a 0
    Limpia la pantalla
    Calcula distancias de los dos sensores
    Indica el porcentaje de bateria
    Indica la accion actual
    Muestra (si es requerido) que la bateria esta baja
        Pita si está a menos de 10cm de un objeto frontal y cambian los leds
    Muestra distancia a objetos
    Indica angulo de choque
    Apunta si esta en modo testing o debug not walking
-------------------------------------------------
Si se dispone de una conexion serie2 (bluetooth)
    Se almacena en una variable los datos recibidos
    Se pone la variable de nuevo comando en true
-------------------------------------------------
Maquina de estados (switch-case)
    Si estado es 1, idle
    Si estado es 2, saludo. Solo se mueve si no testing. nuevo comando a false. Estado a 1
    Si estado es 3, duerme. Solo se mueve si no testing. nuevo comando a false. Estado a 1
    Si estado es 4, baila. Solo se mueve si no testing. nuevo comando a false. Estado a 1
-------------------------------------------------
Si los datos del bluetooth empiezan por E, estado a 2, saludo
Si los datos del bluetooth empiezan por W, estado a 3, duerme
Si los datos del bluetooth empiezan por Q, estado a 4, baila
Si los datos del bluetooth empiezan por S, secuencia
    Se completa la cadena con varios "0"
    Cada caracter se introduce en un array hasta llenarse (maximo 5) se salta la primera S
    Recorre el array ejecutando las acciones correspondientes.
    Si no es ninguna letra anterior se ignora
    Si lee un 0 o se acaba el array, Estado a 1, nuevo comando a false.
```

Figura 37: Seudocódigo de la función loop

- Primero está implementado todo lo necesario para gestionar el voltímetro y amperímetro. Para el voltímetro, se lee la entrada analógica del sensor y se mapea el valor para conocer el dato.
  Mediante una variable de configuración (batteryAvailable) se puede activar y desactivar la funcionalidad de control de batería baja desconexion de los motores para evitar dañarla. En esta funcionalidad se inclue la propia desconexion de los motores (no implementado en este prototipo) y una indicación auditiva.
  De la misma manera, se controla la corriente que circula por el sistema y en el caso de haber más de 15 amperios desactivar de la misma manera los motores

y realizar una indicación auditiva diferente al a del voltaje. Se imprime por pantalla del PC el porcentaje de batería para tareas de debug.

- Se controla la variable de refresco de la pantalla OLED para que solo actualice cada 5 segundos (5000 ms). Como calcular distancias con los sensores de ultrasonido es costoso en tiempo, se ha incluido en esta sección para que solo lo compruebe cada vez que se actualiza la OLED. Se escribe el nivel de batería del sistema (en el caso de no comprobar la batería, pone 50%). Se escribe también la acción actual del robot o en el caso de tener batería baja, un aviso de ello.
  En el caso de que una de las dos distancias dada por los sensores de ultrasonidos sea menor a 10 cm, emite una alerta sonora e ilumina los LEDs rojos como indicador visual. De la misma manera, escribe las distancias en la pantalla OLED e indica el ángulo de choque comparando ambas distancias.
  Si el robot está en modo "testing" o "debugNotWalking" tambien lo indica en la OLED.

- Se gestionan los datos recibidos por bluetooth. Si se ha recibido algo nuevo por el puerto serie2 conectado al módulo bluetooth, se almacena en una variable y se pone la variable "newCommand" en true para decirle al sistema que tiene un comando nuevo disponible para procesar.

- Máquina de estados básica (figura 338). Aquí es donde se añadirán el resto de los estados que permitan andar y girar al robot conforme se implementen en un futuro.

  - Estado 1 significa que el robot se encuentra en estado de idle.
  - Estado 2 indica que el robot está saludando.
  - Estado 3 marca la acción de dormir
  - Estado 4 hace bailar al robot

- Filtrado de comandos bluetooth. En esta sección se analiza el comando que nos ha llegado por bluetooth y cambia el estado del robot al correspondiente para que en la siguiente iteración del código ejecute la acción correspondiente.
  En el caso de que el comando contenga una S (de sequence) en la primera posición, esto indica que lo recibido es una secuencia compleja con la que rellenará un array por cada una de sus letras y las ejecutará en ese orden. Por ejemplo. si el robot recibe "E", saludará, si recibe "W", se tumbará, pero si recibe "SWQW" entrará en modo secuencia, almacenará en un array "S", "W", "Q", "W" y 0s (añade 0s hasta llegar a 7 posiciones) y recorrerá desde la posición uno (para saltarse la primera S) del array hasta el tamaño del array realizando la acción de "W", "Q" y "W" en ese orden.
  En el caso de ser un secuencia que no quepa en el array de 7 posiciones (incluyendo la S), solo se ejecutarán las 6 primeras posiciones. Pese a que se indica que el máximo son 5 (más la S y el final de string suman 7 posiciones), la función "toCharArray()" permite rellenar el array hasta que se complete sin que de fallos por tener caracteres extra el string a convertir. La encargada de añadir la S al inicio de la secuencia es la app (figura 39), el encargado de rellenar el string para alcanzar más de 5 posiciones es el robot (figura 40).

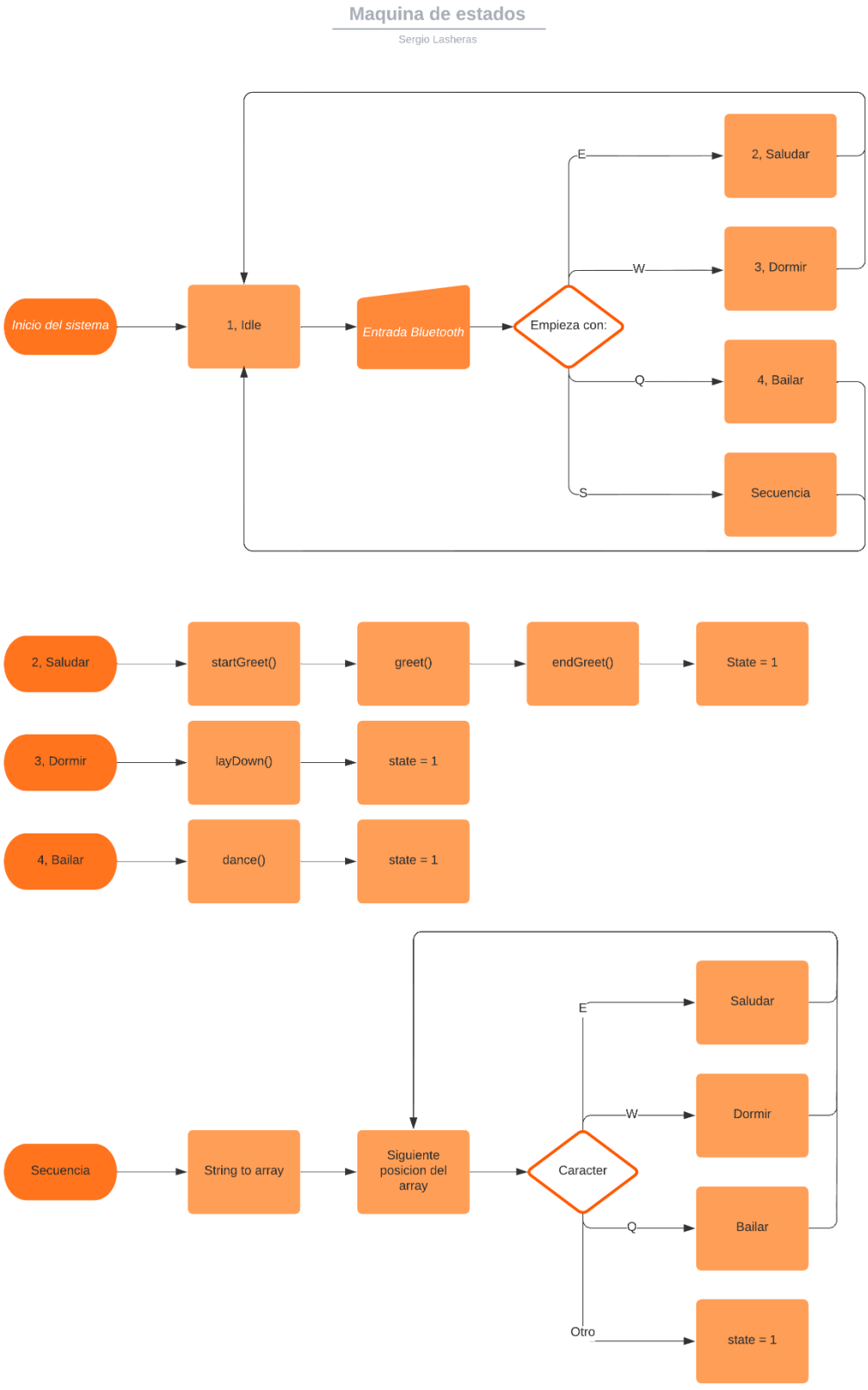

Figura 38: Máquina de estados básica

Figura 39: Función de la app que añade la S a la secuencia

```
String sequence = BTData;
sequence = sequence + "000000";

sequence.toCharArray(splitSequence, sizeof(splitSequence));
```

Figura 40: Función que añade 0s y convierte "sequence" en el array "splitSequence" en el robot

5.3 Diseño y desarrollo de la aplicación

La forma de manejar el robot es mediante el control remoto inalámbrico. Para realizar esta comunicación existen varias técnicas desde señales de radio hasta microondas (wifi, 4G, bluetooth…). En este caso, se realiza mediante una conexión bluetooth, por lo que se necesita algún controlador capaz de actuar como dispositivo maestro de la conexión y que se le pueda programar de alguna manera los comandos que circulan entre el controlador y el robot.

Todo el mundo tiene un móvil con bluetooth en el que se pueden descargar una aplicación, por lo que se decidió usar este método de control. Esta aplicación ha sido realizada utilizando la página web del MIT "app inventor" (MIT, s.f.), que te proporciona un lienzo en el que hacer una aplicación básica y sencilla para Android.

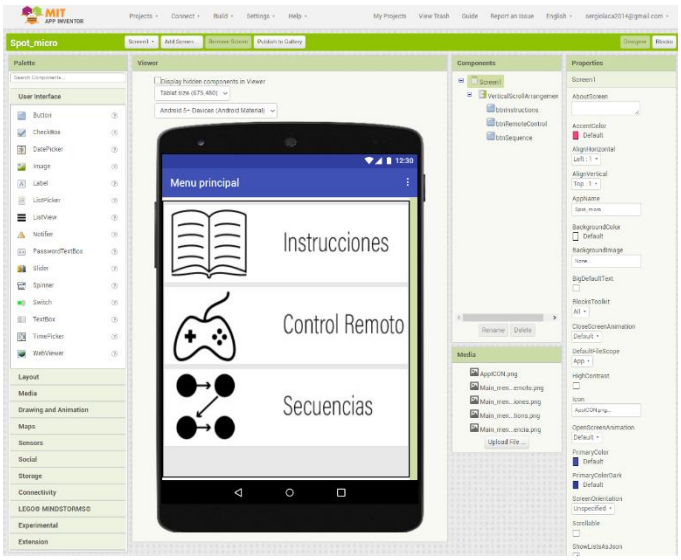

Figura 41: Interfaz principal de diseño de MIT App inventor

En esta web (figura 41) tiene a la izquierda los diferentes elementos que puede contener la escena (Interfaz de usuario, organización, multimedia, dibujo y animación, mapas, sensores, funciones sociales, almacenamiento, comunicación). Las secciones más interesantes para remarcar son la de conectividad, que te permite añadir funcionalidad bluetooth sin tener que realizar ajustes extra aparte de la propia configuración de la comunicación, módulos de lego mindstorms para realizar aplicaciones capaces de manejar kits de robótica de lego NXT y Ev3 y la posibilidad de importar módulos y extensiones personalizadas. A la derecha de la pantalla existen los componentes que hay en la escena en formato de árbol, una sección en la que están todos los recursos multimedia que hay en la pantalla de la aplicación y las propiedades

de cada uno de esos componentes de la escena. En la parte de arriba te permite cambiar entre escenas o pantallas, cambiar entre el modo diseño o modo "bloques" para programar y las opciones para conectarse con el móvil y testear la aplicación (a destacar).

5.3.1 Pantalla principal

Esta aplicación consta de una pantalla principal donde un selector te permite elegir si quieres ver las instrucciones, controlarlo de forma remota con flechas o botones de acción y la posibilidad de escribir una secuencia y mandarla para que la realice en ese orden.

Esta pantalla contiene un código muy sencillo, únicamente son 3 botones que al pulsarlos cambia de escena a la correspondiente. Esta pantalla se puede ver en la figura 35. El código para ello es muy sencillo, se puede ver en la figura 42

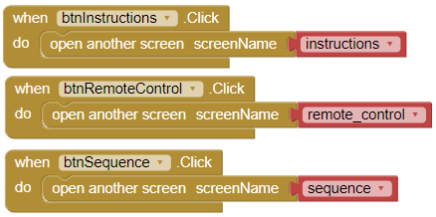

Figura 42: Código de cambio de ventana

5.3.2 Instrucciones

La pantalla de instrucciones es muy similar, son tres botones, cada uno con un rango de edad recomendada conectados a un objeto de tipo "activity starter" que mediante la accion "android.intent.action.VIEW" permiten visitar una URL almacenada en el objeto. En este caso, los tres botones llevan a unos PDF (placeholder) almacenados en Google Drive para aprovecharnos del visor de PDF que integra este servicio de almacenamiento en la nube (figura 43).

Figura 43: Pantalla de selección de instrucciones

5.3.3 Control remoto

La pantalla de control remoto (figura 44) contiene unos botones cuya función es conectar o desconectar el robot del dispositivo, tres botones de comportamiento básicos y las flechas de movimiento.

Al entrar en esta pantalla se crea una lista de dispositivos bluetooth disponibles para su posterior seleccionado y conexión.

Figura 44: Pantalla de control remoto

- El botón "conectar" despliega un selector que contiene todas las direcciones y nombres de los dispositivos bluetooth disponibles para conectarse. Si el robot está encendido, aparecerá el dispositivo "Perrito_Roboto" y al seleccionarlo se iniciará la comunicación. Del mismo modo, se bloqueará el botón y se iluminará en rojo el botón de desconectar, al igual que saldrá un mensaje indicando que se ha conectado satisfactoriamente.

- El botón "desconectar" quitará la conexión del robot, pone el botón de conectar en verde, bloquea el botón de desconectar y habilita el botón de conectar. Tambien muestra un mensaje indicando que se ha desconectado.

- El botón "Saludar" envía una señal serie "E" por la conexión bluetooth activa.
- El botón "Dormir" envía una señal serie "W" por la conexión bluetooth activa.
- El botón "Bailar" envía una señal serie "Q" por la conexión bluetooth activa.
- Los botones de dirección envían "U" para adelante, "D" para atrás, "L" para izquierda y "R" para derecha por la conexión bluetooth activa.

5.3.4 Secuencias

La pantalla de secuencias contiene los mismos botones de conexión y desconexión, pero a diferencia de la anterior tiene un campo de texto para escribir una secuencia separada por comas (UDQW…) y el botón de enviarla. Está incluida una lista de letras que usar para que los usuarios tengan siempre presente que corresponde cada letra (figura 45). Al tocar el campo de texto, aparecerá el teclado de Google y se podrá escribir.

Se ha decidido hacer de esta manera y no poniendo botones porque para los usuarios menos avanzados pueden adquirir soltura al navegar por un teclado, y los más avanzados puede llegar a ser más rápido que teniendo que buscar el botón que pulsar en la pantalla.

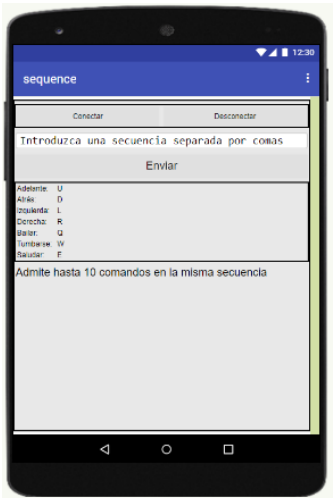

Figura 45: Pantalla de secuencias

A nivel de codigo, cuando se pulsa el boton enviar, la app concatena una "S" con lo escrito en el campo de texto pasado a mayusculas (figura 46). El arduino reconocerá que el string que recibe empieza por S, separará las letras y lo almacenará en un array (figura 40)

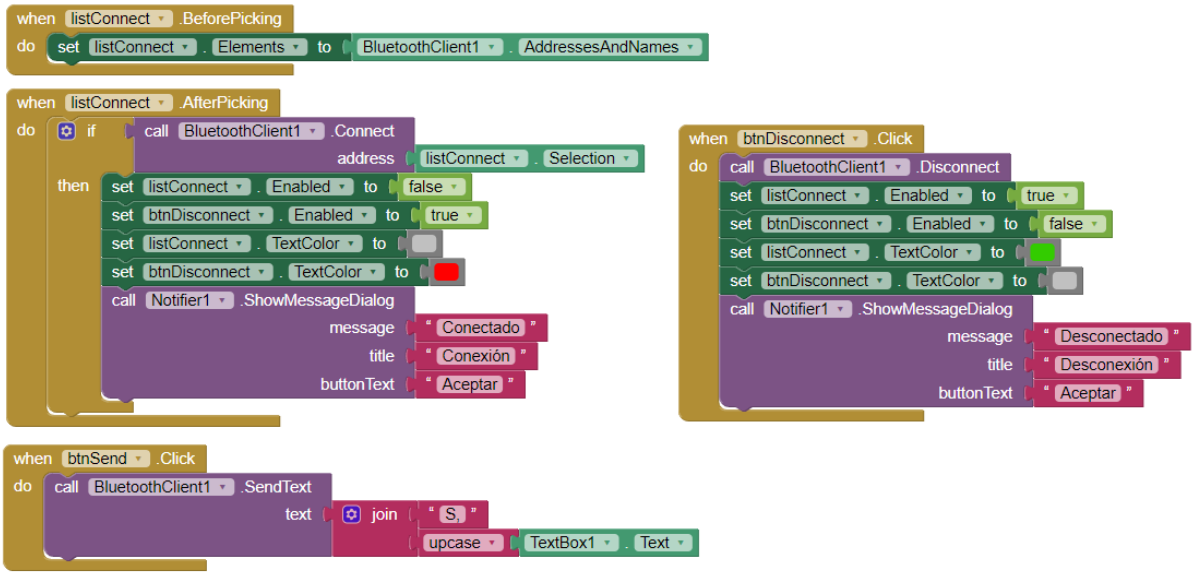

Figura 46: Código para enviar las secuencias por bluetooth

5.4 Kits de desarrollo

Pese a la decisión de crear 3 kits de desarrollo adaptados para diferentes edades y niveles de conocimiento dependiendo de la cantidad de piezas disponibles y dificultad de montaje, toda la documentación, código del robot, código de la aplicación, referencias y módulos son completamente modificables y compatibles para futuros cambios o mejoras por parte de los usuarios.

Para realizar los kits de desarrollo se ha modificado e interpretado un proyecto opensource a nivel global llamado "spot micro" en el que cientos de personas participan, cada uno con sus versiones de las que se puede investigar para aprender y mejorar en gran medida las funcionalidades del robot.

5.4.1 Kit de desarrollo básico

El kit de iniciación en la robótica está pensado para edades de 7 a 12 años aproximadamente, con un conocimiento prácticamente nulo sobre mecánica, robótica, electrónica e informática. Es el kit más básico y contiene todo lo necesario para el montaje. Las partes que se incluyen en el kit son:

- Patas (x4)
- Cuerpo (despiezado + tornillos)
- Cabeza (despiezado + tornillos)
- Abdomen (despiezado + tornillos)
- Alimentación
- Control
- App
- Batería
- Cargador de batería

Todos estos componentes y piezas son completamente independientes, encapsulados y sin riesgo para la salud. Utilizan conexiones y enchufes grandes, lo suficiente para que sean fáciles de manipular. De las patas sale un conector (todos son numerados) que contiene 4 líneas. La primera contiene la señal positiva de +5V, la segunda y la tercera las señales PWM de cada uno de los dos motores y la cuarta línea como GND.

La cabeza contiene los dos sensores de distancia (ultrasonidos) de la cual sale un único conector de con +5V, Echo 1, Trigger 1, Echo 2, Trigger 2 y GND. Tambien contiene 6 leds de dos colores, del que sale un conector de 3 pines con un cable de +5V para los leds azules, uno de señal para los leds rojos y el GND de ambos.

La alimentación es un bloque con varios conectores en él. Uno de ellos corresponde a la alimentación externa, donde la entrada puede ser entre 15V y 7'4V (se recomienda usar una fuente de 9V o 12V y 15A). Contiene un conector de batería para alimentarlo con una batería de litio (incluida en el paquete) con enchufe tipo "TRX". Contiene tambien dos cables de salida (+6'8V y GND) que van al módulo de control.

El módulo de control es otro bloque al que se puede conectar el cable que sale del bloque de alimentación. Dado que es un proyecto opensource, los conectores son de tipo "molex" (estándar de cables de alimentación interna de un ordenador) que son capaces de aguantar la corriente

necesaria para el funcionamiento del robot. Posee cuatro conectores de cuatro líneas cada uno (donde se enchufan las 4 patas). Contiene el Arduino, el HC-05 bluetooth, controladora PWM. Tambien cuenta con un cable con 4 líneas para conectarlo a la pantalla OLED que incluye la carcasa del cuerpo.

La batería es similar a las usadas durante los prototipos y el cargador vendrá con un cable alargador para poder cargar la batería sin tener que sacarla del robot.

5.4.2 Kit de desarrollo medio

El objetivo del kit medio es tener una experiencia cercana a construir el robot desde 0 pero listo para montarlo. Pensado de 13 a 17 años, todos los componentes vienen calibrados de fábrica y todas ellas utilizan conexiones perfectamente marcadas para poder identificarlas en las instrucciones.

Para esta alternativa se diseñará una placa base (PCB) personalizada similar a las que crean para impresoras 3D (figura 47) con los conectores necesarios para encajar todas las otras piezas.

Cada uno de los otros módulos externos no serán los originales. Serán modificados y adaptados y vendrán también con conectores estándar para facilitar el montaje. Pese a ser modificados, en caso de requerir un cambio o actualización de los componentes se incluirá una tabla de componentes y números de serie para poder buscar recambios de forma sencilla y poder cambiarlos usando adaptadores oficiales, no oficiales o incluso soldando el cable directamente. Las señales que circulan por estos cables ha sido explicada en secciones anteriores.

Figura 47: PCB de conexiones de impresora 3D

Se incluyen también todos los recursos que han formado parte de este proyecto, entre los que se incluye este documento, el código del robot, el proyecto de la aplicación y las hojas de datos. Todos estos documentos podrán encontrarse en una página de GitHub creada cuando se termine el proyecto y son públicos, de libre uso. De la misma manera, las hojas de datos se pueden encontrar en internet mediante el buscador "AllDataSheet" (AllDataSheet, s.f.). En los kits se incluyen unos códigos QR que enlazan a estos documentos para poder localizarlos más fácilmente.

Está prevista una actualización de la aplicación para incluir los enlaces a estos documentos de igual manera que se pueden acceder a las instrucciones.

5.4.3 Kit de desarrollo avanzado

Para gente que no le tiene miedo a experimentar, en este kit pensado para más de 18 años (hace falta usar herramientas complejas, no incluidas) y el contenido son todas las piezas por separado sin calibrar y el código del Arduino lo tendrán que cargar los usuarios en el dispositivo, no vendrá precargado como en los kits anteriores.

Se requiere uso de alicates, pinzas, soldador/estaño, destornilladores y el acceso a un ordenador y las instrucciones de este kit incluirán documentación extensa sobre lo que está ocurriendo y las posibilidades de modificación.

Este kit está pensado para aquellas personas que no quieran buscar los componentes por su cuenta, que quieran tener un paquete con todo incluido como si de un puzle se tratase, todo lo necesario para construirlo y sin tener que preocuparse por si les falta algo.

Este kit representa el culmen de la investigación realizada, representada a lo largo del capítulo 5 (Diseño y desarrollo del prototipo y los kits).

6 Estudio económico

Se ha realizado un presupuesto extenso con los enlaces de compra, unidades y precio de cada uno de los componentes que se necesitó adquirir para la construcción del prototipo funcional los cuales se pueden ver en la tabla 3. No se incluyen en esta tabla componentes o piezas que se ha conseguido rescatar de otros proyectos personales (como el propio Arduino Mega).

Pieza	pack piezas	Cantidad	Precio unitario	Precio total	Enlace	Comentarios
Servomotor DS3218 PRO	4	3	39,62	118,86	https://es.aliexpress.com/item/4000052892768.html	
Servomotor MG995	1	4	3,22	12,88	https://es.aliexpress.com/item/1005002336388915.html	
Servomotor MG996R	4	1	19,99	19,99	https://www.amazon.es/gp/product/B07DQFXDC9	
Metal Servo Arm 25T Disco Metal	6	2	10,9	21,80	https://www.amazon.es/gp/product/B07G5BTMJB	solución fallo deslizamiento eje motor
Servomotor MG996R	5	1	23,99	23,99	https://www.amazon.es/gp/product/B07H89JH74	Devuelto
				0,00		
Filamento 3D negro	1	1	21,99	21,99	https://www.amazon.es/gp/product/B07FQDKR28	
Filamento 3D amarillo	1	1	21,99	21,99	https://www.amazon.es/gp/product/B07FQL6L3W	
				0,00		
Rodamientos 625RS	20	1	10,09	10,09	https://www.amazon.es/gp/product/B07TSP8SBP	
				0,00		
Tornillos M2, M3, M4 y M5	1	2	21,00	42,00	https://www.amazon.es/gp/product/B08BXSZJ64	

				0,00		
Controladora servos PCA9685	1	1	6,99	6,99	https://www.amazon.es/gp/product/B07BS8B637	
				0,00		
Arduino leonardo R3	1	1	12,99	12,99	www.amazon.es/gp/product/B0786LJQ8K	Cambiado por Arduino Mega
				0,00		
AZDelivery HC-05 HC-06 Bluetooth	1	1	9,49	9,49	https://www.amazon.es/gp/product/B0722MD4FY	
Movilidades 5 unids Micro USB PCB	5	1	5,49	5,49	https://www.amazon.es/gp/product/B08BZWS997	
ICQUANZX Relé 5PCS KY-019 5V	5	1	7,99	7,99	https://www.amazon.es/gp/product/B07BVXT1ZK	
30Pcs Cable de Extension Servo	30	1	12,99	12,99	https://www.amazon.es/gp/product/B082WYFWSY	
DC0-25V y ACS712 corriente	4	1	11,99	11,99	https://www.amazon.es/gp/product/B08BZKPSFY	
ELEGOO Sensor Ultrasonidos HC-SR04	5	1	9,99	9,99	https://www.amazon.es/gp/product/B08BZKPSFY	

Zumbador de añarma de BIOS	10	1	6,99	6,99	https://www.amazon.es/gp/product/B07VH798TB	
AZDelivery OLED display I2C SSD1306	1	1	6,79	6,79	https://www.amazon.es/gp/product/B01L9GC470	
				0,00		
240 Piezas Jumper Cables	1	1	12,99	12,99	https://www.amazon.es/gp/product/B08HQ7K6M7	
Cable calibre 14	2	1	12,99	12,99	https://www.amazon.es/gp/product/B074QR9DT9	
Total				411,27		

Tabla 3: Inventario y presupuesto de piezas

Tampoco se incluye el tiempo de desarrollo e investigación como dinero invertido en el proyecto dado que se presupone que son las horas requeridas para completarlo. Se puede estimar que de las 285 horas destinadas al trabajo autónomo que indica la guía docente, al menos 225 horas a lo largo de todo el año han sido de investigación. Si se tienen en cuenta que este proyecto ha llevado más de las 300 horas inicialmente planificadas, la relación entre investigación-implementación se sitúa aproximadamente un 80% del tiempo invertido ha sido en investigación.

Si nos ceñimos a los datos de la guía docente, las 285 horas por 7'66€ de salario mínimo (que se redondeará a 8€) dan un total de 2280€ aproximadamente. A esos 2280€ hay que sumarle esos 411€ de componentes adquiridos para este prototipo y otros 100€ de componentes que ya tenía, lo que suma un total de 2780€ invertidos en la creación de este proyecto.

Teniendo en cuenta que solo el 20% del tiempo ha sido utilizado en implementación, esto hacen un total de 60 horas de prueba y error para el primer prototipo, que disminuirá enormemente cuando pase a la fase de producción en serie de los kits.

Para simplificar los cálculos, un tercio de los kits que se vendan serán básicos, otro tercio medios y lo mismo para los avanzados.

Un kit básico cuesta más tiempo de preparar que el medio, y el medio cuesta más de preparar que un avanzado, el cual solo es meter los componentes en una caja.

Un kit básico contiene muchos componentes y elementos personalizados, el medio contiene placas PCB personalizadas y el avanzado tiene los componentes sin modificar, por lo que esto tambien influye en el precio final.

Sabiendo que los componentes al por menor han costado 511€ y que varios incluían más unidades de las necesarias, se asumirá que al por mayor y en grandes cantidades se pueden conseguir por 200€ en total, por lo que partimos de esa cantidad para los cálculos.

Se presenta una tabla 4 con los cálculos y estimaciones por el que se podrían poner a la venta los diferentes kits tomando en consideración que cada uno de ellos cuesta un tiempo diferente de preparar y contienen componentes personalizados.

Nombre del kit	Tiempo de preparación	Componentes personalizados	Cuota de mercado relativa	Esfuerzo y modificaciones invertidas en €	Precio total del kit	PVP
Básico	2 horas	70€	33%	85€	285€	300
Medio	1 hora	40€	33%	48€	248€	275
Avanzado	15 minutos	0€	33%	2€	202€	230

Tabla 4: Estimación de coste por kit

Estos datos totales son los precios de coste de cada kit, que podrían ser aumentados a los 300€, 275€ y 230€ respectivamente para suplir los gastos iniciales de investigación y prototipos.

Si se vendiesen 9 al día con la cuota relativa del 33% indicada anteriormente, tendríamos unas ganancias diarias de 25*3 + 27*3 + 23*3 = 225€ al día lo que haría que en dos semanas se recuperase la inversión inicial para desarrollar este producto.

7 Resultados

7.1 Desviaciones en la planificación inicial

7.1.1 Primera entrega (19/11/2021)

La primera entrega se realizó sin complicaciones, se presentó por correo electrónico la planificación del proyecto, plazos iniciales y la metodología ágil decidida.

Necesitaba unos cambios para representar de forma correcta la organización del proyecto, por lo que al día siguiente presenté los gráficos y explicaciones expuestas al inicio de esta sección

7.1.2 Segunda entrega (17/12/2021)

Aquí empezaron los problemas. Para esta entrega tenía planificado presentar el robot montado (sin baterías) y el control remoto básico, pero finalmente solo se presentó el montaje básico con fallos en el diseño, las formas que veía posibles para repararlo y los diversos sistemas extra tales como sensor de distancia, voltaje, amperaje, bluetooth y alimentación implementados y testados de forma independiente para su rápida integración dentro del robot funcionando en coordinación.

Se prepararon unas modificaciones en el diseño del robot para añadirle 5cm de longitud al torso para tener más espacio de trabajo y estabilidad a costa de un mayor peso.

7.1.3 Tercera entrega (23/02/2022)

Se entrega una memoria actualizada con el presupuesto realizado, resumen, abstract, objetivos y antecedentes.

Respecto al prototipo, se comunican fallos serios de funcionamiento mecánico y se presenta la solución a esos fallos ya implementada y comprobada que los solucionaban.

Aquí hay un vacío de varios meses en los que el proyecto ha avanzado más lento de lo esperado por motivos personales y de tiempo disponible.

7.1.4 Cuarta entrega (08/06/2022)

Memoria terminada.

Prototipo finalizado y funcionando de cara a la presentación, aunque con los problemas indicados anteriormente tales como movimiento (adelante, atrás y girar) y la necesidad de usar dos fuentes de energía independientes.

El proyecto originalmente estaba pensado para ser terminado en febrero, concretamente el 25 de este mes, habiendo sido retrasado hasta junio, pocos días antes del 24. Esto deja una diferencia de tiempo de 6 meses. Estos 6 meses de diferencia se han compartido con otras asignaturas y proyectos por lo que no se puede considerar que en términos de esfuerzo se haya necesitado 6 meses íntegros. La diferencia en este caso se calcula que se ha necesitado un mes y medio aproximadamente en jornadas de 4 horas al día para poder completar el proyecto en el estado de la entrega.

7.2 Pruebas, problemas y posibles mejoras durante el desarrollo

En esta sección se incluirán diferentes problemas que han sucedido durante el desarrollo de todas las indoles, separados por categoría en mecánica, electrónica, código y aplicación. Se incluirán los motivos por los cuales han surgido esos problemas, cómo han sido solucionado o posibles soluciones que no pueden ser implementadas por falta de tiempo y presupuesto y posibles propuestas de mejoras de funcionamiento.

7.2.1 Mecánica

Este proyecto contiene muchos campos de las ingenierías, no solo programación, por lo que la parte de mecánica ha sido un lienzo en blanco mediante el que, pese a que antes de construirlo se han diseñado soluciones que a priori se pensaba que podían funcionar, se han encontrado diversos errores. Esta parte del proyecto contiene mucha "prueba y error" por ese motivo.

- Elección de los motores (figura 48): A diferencia de lo que pueda parecer, no todos los Servomotores son iguales. Poseen muchas diferencias tales como ángulo máximo de giro, torque o fuerza de giro, precisión, tamaño y materiales de construcción.
  Inicialmente pensé en usar unos Servomotores MG996R, con un torque de 11KG por centímetro y un sistema de parada interno que marca los grados máximos fácil de romperse. Tambien Se valoró usar otra opción de servos con engranajes internos de plástico, pero usar engranajes de plástico para soportar mucha carga no es buena idea porque los dientes se queman, por lo que opté por unos servos con mayor torque y engranajes internos metálicos. El tamaño de los servos son prácticamente iguales por lo que no he tenido ningún problema de diseño al cambiar entre ambas opciones.

  Hay un problema con los servomotores y es que son muy útiles por su sencillez de manipulación y precio para vehículos radiocontrol, en los que el giro de las ruedas no necesita ser muy preciso. Para el robot, sin embargo, estos motores son incapaces de dar feedback de su estado actual y poder usarlo para calcular el siguiente giro al igual que es imposible regular la velocidad de estos. La velocidad depende del voltaje y la fuerza que estén soportando, lo que no se puede controlar y esto provoca problemas graves de sincronía entre todas las patas de cara a acciones más complejas. Por ello y por falta de tiempo, por ahora este robot solo se moverá realizando acciones poco complejas en el sitio.

  

  Figura 48: Junta entre la pata y el cuerpo del robot

  - Una posible mejora de la motorización podría ser utilizar otro tipo de motores. Las alternativas son motores paso a paso, usados en maquinaria CNC como impresoras 3D dado que uno de sus puntos fuertes es la precisión, pero su torque es muy bajo comparado con otros tipos de motores.
    Otra alternativa son motores serie, en los que la fuerza es similar a los servomotores, pero su manipulación no depende de una

señal PWM sino de una comunicación en serie, teniendo la capacidad además de poder conectar los motores entre ellos para poder manipularlos en cadena. El problema principal de los motores serie es que son más complejos de programar y mucho más caros.
Una tercera alternativa, no tanto de motores como de diseño, es realizar una estructura que por medio de sus propios ejes o engranajes permitan el movimiento deseado utilizando un motor en vez de 2, con lo que son menos motores para sincronizar y por lo tanto mayor precisión general.

- Engranajes motor-estructura. Los servomotores vienen de serie con unos engranajes de plástico para tareas que soporten poco peso. El problema cuando lo sometes a más fuerza es, como se puede ver en la figura 49, que se parten y el engranaje interior que encaja con el eje dorado del motor se quema y resbala. El diseño que utilicé venia preparado para usar soportes del tipo estrella, como el que se ve en la foto, por lo que tuve que rediseñar algunas de las piezas para poder encajar unos soportes de aluminio (gris en la figura 49) y evitar que se deslizara esa unión. El diseño original se puede ver en la figura 51, el cambio se puede ver reflejado en la figura 52.

Figura 49: Engranajes de plástico roto y metal

Pese a ello, aunque estos discos de aluminio funcionan mucho mejor (de calidad baja debido al presupuesto limitado) terminan resbalando tambien.
Este es el principal motivo por el que actualmente el robot funciona en modo debug, donde tareas simples como tumbarse, sentarse y bailar en el mismo sitio puede realizarlas, pero no están habilitadas las funciones de andar. Este problema va ligado con los de sincronía del punto anterior.

Figura 51: Pieza con soporte en estrella

Figura 50: Diseño final de los engranajes de los ejes

- La forma de solucionar este problema a largo plazo va por usar materiales de mayor calidad, que permita una fijación más perfecta con el eje del motor y, en el caso de los hombros, utilizar un soporte para que esa articulación esté sujeta por dos puntos y no solo uno de ellos, como se puede ver en la figura 50.

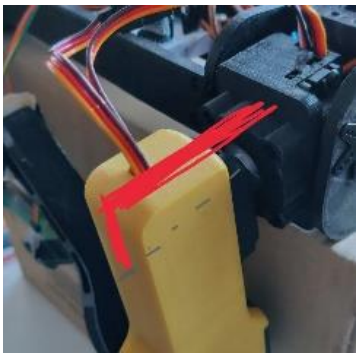

Figura 52: Propuesta de cambio del eje del hombro

- Las patas del robot son de plástico PLA (utilizado en impresión 3D). Este tipo de plástico no tiene agarre contra otras superficies, desliza contra el suelo. Para solucionar este problema, las partes que entran en contacto con el suelo se intentaron imprimir en un plástico flexible que a priori parecía que iba a tener mayor agarre con el suelo, cosa que no ocurrió.
  Tras una larga investigación de materiales que se podían utilizar para solucionar este problema, se pensó materiales tipo "goma" y en concreto el látex. El látex es un material similar de trabajar a la cola blanca, aunque con propiedades más aptas para esta tarea, por lo que se echaron varias capas en una superficie plana y sacar varias piezas que se pudieron encajar en las patas (figura 53). Posteriormente eché más capas de látex a la pata para hacer una estructura uniforme y evitar que se pudiese dañar la superficie con la fricción contra el suelo.
  Lamentablemente, en un inicio el robot podía andar porque pese a su mala sincronía, sus patas deslizaban en el suelo y la precisión no era tan necesaria. Al hacer que las patas agarren mejor era necesaria un funcionamiento más perfecto para no quemar los engranajes al hacer fuerza los motores.

Figura 53: Almohadilla de látex

  - Una posible mejora en este aspecto es hacer "capuchas" de caucho u otros tipos de goma más profesionales y de mayor calidad. Esta solución no deja de ser un parche casero con materiales a mi disposición.

7.2.2 Electrónica

- El proyecto originalmente fue pensado para realizar con una placa Arduino UNO y se empezó el desarrollo con ese microcontrolador. Con el avance del proyecto, empezó a dar problemas esta placa por cantidad de memoria interna insuficiente (el código no cabía) y eran necesario disponer más salidas para poder conectar todos los módulos (figura 54).
  Por ello, se realizó un cambio a la placa Arduino Mega, una versión vitaminada de la placa UNO, con más memoria, más pines disponibles pero misma velocidad y voltaje de funcionamiento.

	Arduino Uno	Arduino Mega 2560	Arduino Micro
Price Points	$19.99-$23.00	$36.61 - $39.00	$19.80 - $24.38
Dimension	2.7 in x 2.1 in	4 in x 2.1 in	0.7 in x 1.9 in
Processor	Atmega328P	ATmega2560	ATmega32U4
Clock Speed	16MHz	16MHz	16MHz
Flash Memory (kB)	32	256	32
EEPROM (kB)	1	4	1
SRAM (kB)	2	8	2.5
Voltage Level	5V	5V	5V
Digital I/O Pins	14	54	20
Digital I/O with PWM Pins	6	15	7
Analog Pins	6	16	12
USB Connectivity	Standard A/B USB	Standard A/B USB	Micro-USB
Shield Compatibility	Yes	Yes	No
Ethernet/Wi-FI/Bluetooth	No (a Shield/module can enable it)	No (a Shield/module can enable it)	No

Figura 54: Comparativa de diferentes procesadores Arduino (Gudino, s.f.)

- El sistema de alimentación fue complejo de diseñar por la gran cantidad de corriente que necesitaba el circuito (15A). Esto trajo problemas a la hora de elegir el transformador, el tipo de cables y el relé capaz de soportar los requisitos del circuito. En un inicio pensé en usar varios reguladores de tensión en paralelo para sumar los 15A necesarios, pero ajustarlos todos exactamente al mismo voltaje y dividir cargas iba a ser muy complicado, además que tener varias placas ocupaba más espacio que utilizar un solo modulo un poco más grande.
  Los cables fue una decisión más sencilla. Existen tablas comparativas entre diámetros (AWG) y el máximo voltaje y corriente que pueden soportar. Se eligió un cable 14AWG para el cable externo y cable azul y marron clásico de cableado de las casas un poco más finos para mover la electricidad en el interior del sistema.
  El relé de alimentación fue una decisión más difícil. En un inicio se pensó en usar un relé pequeño a 5V capaz de llevar 10A y 250VAC como máximo, pero la alimentación es a 6’7V por lo que iba a quemar la bobina. Se optó por utilizar un relé de intermitentes coche, a 12V de activación nominal y 12V 40A de circulación máxima, suficiente para lo que se necesitaba (figura 55).
  El problema ahora es que 12V está muy por encima de los 6’7V, por lo que ese voltaje no era capaz de activarlo. Viendo la hoja de datos, la tensión nominal pero la de funcionamiento es de 7V a 18V y esto es una buena noticia. El transformador a esos 6’7V estaba puesto antes del relé, pero pudiendo regular la alimentación externa al voltaje que queramos y la batería siempre da más de 7V (7’4V en su mínima carga segura), ambos voltajes podían estar por encima de los 7V necesarios para activar el relé. La solución fue tan sencilla como mover el transformador al final del sistema de alimentación.

ESPECIFICACIONES

1. TENSION NOMINAL: 12 V.
2. TENSION DE FUNCIONAMIENTO: 7 - 18 V.
3. TEMPERATURA FUNCIONAMIENTO: -30° A +80°C.
4. TEMPERATURA ALMACENAJE: -40°C A +85°C.
5. TENSION DE CIERRE: 7 V. <U< 10 V.
6. TENSION DE APERTURA (U): 1 V. <U<3.5 V.
7. INTENSIDAD MAXIMA: 40 A.

Figura 55: Especificaciones Relé RLP12V

Además, esto otorga otra ventaja (no planeada). El tiempo de actuación del relé en el que no hay contacto en ninguna de las dos conexiones es corto, pero no despreciable, suficiente para reiniciar el sistema. Gracias a que el transformador tiene condensadores, son capaces de almacenar esa carga suficiente para evitar el corte de corriente al sistema durante los milisegundos que dura la conmutación del relé.

  - Una posible mejora en este apartado es diseñar una fuente de alimentación de una pieza, con su propia PCB (placa de componentes) con todas las conexiones y sensores en ella. Con esto se evita usar tanto cable y se podría ahorrar espacio, al igual que los diferentes sensores estarían diseñados para funcionar con las características que requerimos y no necesitarían un ajuste posterior. A la hora de hacer un prototipo como es este proyecto no es una opción viable por el precio, pero en producción en masa saldría más barato que tener los componentes por separado.

- La controladora PCA9685 tiene conexión I2C (+5V, SDA, SCL y GND) para comunicarse con el Arduino, pero tambien contiene una entrada directa a la alimentación de los motores. Los motores al contener bobinas pueden meter interferencias al circuito, por lo que es necesario incluir un condensador para evitar estas interferencias y un diodo para evitar corriente inducida (al parar un motor, éste emite un pico de corriente inversa). Este diodo está preparado para funcionar con 5V, con lo que los 6'7V que se están usando lo quemarían, al igual que las líneas internas de la PCB no están preparadas para ello. La solución es realizar un puente entre el conector de la placa y los pines del motor como se puede ver en la figura 56.

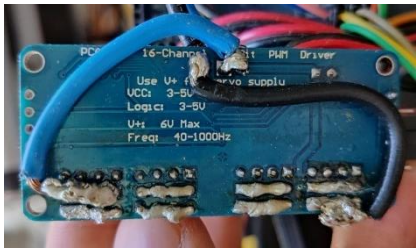

Figura 56: Trasera de PCA9685 con alimentación puenteada

  El prototipo que aquí se presenta no lleva el diodo para parar esa corriente inducida en sentido opuesto ni el condensador para las interferencias porque ha sido imposible de conseguir una que llegase a tiempo (la placa PCA tenía uno de cada, pero fue necesario puentearlos), por lo que el funcionamiento actual implica tener el Arduino alimentado por una power bank a 5V en un sistema independiente (pero con ambos GND de ambos circuitos conectados juntos).
  Los motivos por los que se sospecha de este fallo y su posible solución es porque esa corriente se induce cuando los motores paran en seco, justo en el momento que se reinicia el Arduino y en la pantalla OLED vuelve a salir "STARTING".
  Se puede descartar que sea un problema del sistema de alimentación porque ocurre lo mismo con la fuente de alimentación de laboratorio con la que se están realizando las pruebas.

- El Arduino tiene varias entradas y salidas, pero algunas como los 5V o GND solo hay uno o dos. Esto hace que sea necesario una regleta o artilugio donde pueda conectar una de estas salidas a muchas diferentes (de un cable del Arduino a un cable para cada sensor). Esto se ha suplido con unas placas base especiales que son una cuadricula de agujeros, pero no tienen líneas internas, se realizan con soldadura. Son muy usadas en prototipos porque son baratas y sencillas de utilizar (figura 57).

  - Diseñar una PCB específica para los sistemas que se usan es una solución a la hora de fabricarlas en masa, pero no para prototipos, al igual que utilizar una PCB especifica que contenga toda la alimentación.

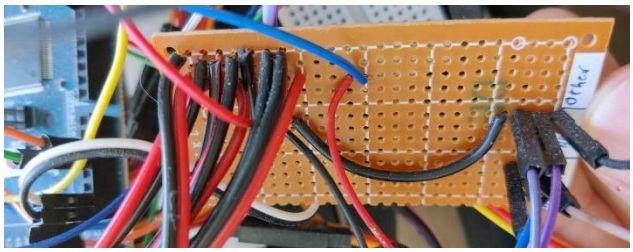

Figura 57: Placa base para prototipos con cableado

7.2.3 Código

- Originalmente el proyecto se planteó para usar cinemática inversa. Se puede configurar en el código la longitud de cada segmento de las patas, posición y orientación de los ejes y recorrido y ángulo de giro mínimo y máximo, pasarle al robot la posición de las patas y que el propio robot sea el que calcule como tiene que mover los motores.
  En robótica avanzada, se utiliza este tipo de algoritmos para lograr moverse con precisión, solo que en el caso de este robot no es una buena solución por diferentes motivos.
  El primero de ellos es el procesador, es muy lento por lo que en este caso es más optimo programar puntos iniciales, intermedios y finales para que los motores los recorran (estos puntos iniciales intermedios y finales han sido previamente calculados con un simulador de cinemática inversa y ajustados en el propio robot).
  De la misma manera, los motores son toscos, tienen poca precisión y la posición inicial de cada uno es diferente (no tienen feedback para ajustarse y sincronizarse entre ellos), por lo que habría que diseñar un algoritmo de cinemática inversa diferente para cada pata, lo que aumentaría el coste de memoria y procesamiento en gran medida.
  Se realizaron pruebas utilizando cinemática inversa, pero se descartaron por el consumo de recursos del Arduino y su nula precisión en esta situación.

- Dada la calidad de los componentes que se están usando, fue necesario un offset de +0.6 V en el código del Arduino para representar el voltaje correcto.

- De cara a la presentación del prototipo, el amperímetro y el voltímetro están programados pero el voltímetro depende de una variable "batteryAvailable" y el amperímetro tiene su código comentado (figura 58). La decisión se ha tomado desde la perspectiva de que todo está en desarrollo y no estarán enchufados los cables que lleven la señal de voltaje y corriente. Al no estar conectados detectaran 0V y 0A y mientras que no tendría problema en funcionar con esa entrada de corriente de 0A, los 0V forzarían al robot un estado de batería baja y no pararía de pitar.

```
/*
  float current = getCurrent(SAMPLESNUMBER);
  if (current >= 15){
  DEACTIVATE PCA CONTROLLER (PIN NOT IMPLEMENTED YET)
  for (int i = 0; i < 3; i++) {
    tone(buzzer, 1000); // Send 1KHz sound signal...
    delay(1000);        // ...for 1 secW
    noTone(buzzer);     // Stop sound...
    delay(10);
  }
  }
*/

if (batteryAvailable) {
  if (voltageValue > 7.37) {
    batteryPercent = ((voltageValue - 7.37) * 100) / 1.03;
  }
  else {
    for (int i = 0; i < 3; i++) {
      tone(buzzer, 1000); // Send 1KHz sound signal...
      delay(200);         // ...for 1 secW
      noTone(buzzer);     // Stop sound...
      delay(10);
    }
  }
}
```

Figura 58: Código del voltímetro y amperímetro

  - En la versión final del producto se implementaría una variable similar a la de debug, pero afectando a todo lo relacionado con el movimiento que solo estaría en "true" cuando el nivel de batería fuese superior al 20%. Por debajo de ese porcentaje bloquearía todo movimiento de los motores para liberar su carga.

  - En un futuro sería posible añadir un relé afectando únicamente a la alimentación de los motores para poder desconectarlos por físicamente ya que la próxima implementación y mejora del robot se añadirá el bloqueo por medio del pin "Output Enable". En el caso de tener mayor presupuesto disponible se implementará el relé directamente sin pasar por el "Output Enable" primero.

- La controladora PCA9685 tiene un pin llamado OE – "Output Enable" que por defecto está en "LOW". Se puede utilizar para desactivar rápidamente todas las salidas. Cuando este pin es "LOW", todos los pines están habilitados. Cuando el pin es "HIGH", las salidas están deshabilitadas. Como en las pruebas no se está controlada la corriente por parte del robot porque la fuente de alimentación externa me da los datos suficientes, este pin no está conectado al Arduino de cara al prototipo funcional. La funcionalidad está implementada pero la línea que comprueba la corriente en el código está comentada en esta fase del desarrollo.

- El Arduino posee un chip atmega2560, con un único núcleo por lo que no es posible hacer más de dos procesos simultáneos. Al ser lento (16MHz), tampoco se puede cargar de muchos procesos basura para intentar simular un procesamiento multi hilo. Con esta premisa, el código está preparado para intentar suplir el secuestro de la CPU ante inputs, tales como leer voltaje, amperaje, distancia, escribir en pantalla… en la medida de lo posible, aunque hay muchas situaciones en las que es más costoso intentar emular que todo funcione en el hilo principal frente a esperar a que vuelva al loop principal teniendo la CPU secuestrada unos milisegundos o pocos segundos (durante la acción).

- Modo debug. Dado que el robot tiene implementadas zonas del código a las que no se accede nunca por estar incompletas o bloqueadas durante la demo técnica en la presentación (como el movimiento adelante, atrás y giro), se implementó un modo testing y otro debug en el que por medio de dos variables podía controlar si quería todo el movimiento, solo el movimiento estático (bailar, tumbarse y saludar) o ningún movimiento. Esta implementación ha sido realizada para poder hacer pruebas de detección de comandos durante el desarrollo y evitar que los motores se moviesen de forma errática y fueran molestos, y para la presentación permitir solo lo que funciona correctamente.

- Relacionado con un punto anterior, la función de secuencia secuestra el procesador. Se ha intentado evitar mediante el uso de variables de control adicionales y usando el main loop como loop principal de la secuencia, pero ha sido extremadamente complicado y ya que estaba llevando demasiado tiempo se ha decidido dejarlo de esta manera de cara a la primera versión prototipo funcional.

- A la hora de recibir información por el puerto serie del bluetooth, esa información se estaba metiendo en un String. Estos Strings acaban con un "null terminator" por lo que no podían ser comparados con comillas dobles, solo con comillas simples para escapar ese "null terminator". Esto dio muchos problemas hasta que encontré la solución de las comillas simples, usando hasta ese momento una función llamada "startsWith()" para poder usar solo el primer carácter del String.

- Utilizar una plataforma como Arduino o ESP-32 conlleva una serie de problemas como no disponer de puntos de parada del código y poder examinar el contenido de las variables. La única forma de conocer el estado de las variables y si entra a determinadas zonas del código es usando el monitor serie que une el PC con el Arduino por el puerto USB. Con una función llamada "serial.Write("contenido del mensaje o variable"); y habiendo configurado la conexión serie (la carga del código en el Arduino no utiliza esta

configuración, es independiente), podemos ver en el ordenador una pantalla con los mensajes y variables que se ha configurado con el serial.Write(). De la misma manera, la comunicación es en ambos sentidos, se pueden usar funciones como serial.Read(); para leer los mensajes que le mandamos desde el ordenador o cualquier puerto serie.

El módulo bluetooth se conecta por un segundo puerto serie del Arduino, por lo que la comunicación con el móvil es similar solo que usando un serial2.Write() y un serial2.Read().

7.2.4 Aplicación

- La idea de usar una aplicación como control remoto surgió como una opción más viable, libre y personalizable a un mando a distancia o control remoto clásico. Primero se pensó en utilizar Android estudio, pero la curva de aprendizaje era demasiado pronunciada para el poco tiempo que podía invertirle, por lo que opté por usar una web creada por el MIT con la que se puede programar visualmente aplicaciones sencillas (como requería).
  Esto ahorró mucho tiempo de desarrollo dado que esta web tiene módulos sencillos de utilizar para enviar comandos serial por bluetooth.

    - La propuesta de mejora en este ámbito es usar Android Studio. La solución de MIT APP inventor 2 es viable a corto plazo, funciona, cumple con lo que se necesita, pero tiene poca capacidad de mejora. Con Android estudio se puede hacer lo mismo y da muchas más posibilidades de mejora para usuarios más expertos en programación.

- El principal problema de la APP fue la posibilidad de incluir PDF. Estas aplicaciones creadas con APP inventor 2 están limitadas en tamaño, suficiente para meter imágenes y gráficos para decorar, pero insuficientes para documentos más pesados.
  Existen módulos de terceros (de pago) que permiten tener un visor de PDF almacenados en tu teléfono, pero no es una buena solución porque requiere que el usuario se descargue esos documentos previamente. Podría hacer lo mismo sin la aplicación.
  La solución dada a esto es un botón que redirige a un PDF almacenado en Google drive, con la posibilidad de ser actualizado sin la necesidad de que el usuario haga nada por su parte y con el tamaño del documento prácticamente ilimitado. Tambien puede enlazar a videos de YouTube y otras páginas web, útil para futuras mejoras de la aplicación.

    - Lo ideal sería que la propia aplicación tuviese esa visualización del PDF o incluso tuviese una propia pantalla con imágenes y texto sin tener que ser un documento externo. Esta pantalla podría ser capaz de actualizarse si en el servidor se ha realizado algún cambio de mejora en las instrucciones.

Por los motivos arriba indicados, la presentación se realizará una demo funcional de los 3 estados (dormir, saludar y bailar), el Arduino será alimentado por medio de una power bank y los motores por el sistema de alimentación diseñado.
No estará cerrado y el sistema de alimentación estará fuera del robot porque en el prototipo se han empleado cables largos para poder trabajar con más holgura y rango de movimiento durante las pruebas.

7.3 Contribuciones de este proyecto respecto a otros similares.

El proyecto resultante contiene varias diferencias con otros proyectos similares existentes. Hay que recordar que la base de este proyecto es una iniciativa opensource llamada spot micro del que mucha gente hace sus propias versiones, por lo que es altamente probable que este prototipo tenga especificaciones similares en algunas funciones con otras personas, otras funciones con otras personas… pero este caso ha sido investigado entre ejemplos de otros proyectos similares y la elección personal desde cero de todo lo que contiene (salvo el diseño 3D impreso, que solo han sido modificadas algunas piezas).

Este proyecto concretamente se ha basado en varios puntos clave:

- Seguridad: contiene control de voltaje y corriente para evitar problemas durante el funcionamiento que pueden causar incendios (contiene baterías).

- Sus componentes son relativamente baratos para otros dispositivos de similares características.

- Los componentes que lleva son universales, utilizan protocolos como el I2C, PWM, Serie… muy utilizados y con infinidad de documentación y ejemplos existentes. Los conectores de igual manera son fáciles de conseguir en el caso de necesitar actualización o reemplazo.

- La aplicación está realizada en un entorno apto para gente con pocos conocimientos sobre programación y los protocolos son aptos para cualquier otra aplicación que se quiera realizar con otros lenguajes o IDEs.

- Todo es opensource, tanto este desarrollo (el cual sería alojado en una página de GitHub propia) como otros similares pueden ser utilizados de forma libre para aprendizaje, investigación e incluso entretenimiento.

7.4 Variaciones respecto a la propuesta inicial.

En un inicio estaba planeado que el robot pudiese moverse hacia adelante, atrás y girar, (incluso la opción de girar automáticamente al chocar en una pared, motivo por el cual lleva los dos sensores y no uno solo) pero por dificultades, problemas, falta de presupuesto y tiempo solo puede realizar los movimientos que no implique moverse en el espacio, lo que corresponde a bailar, saludar y tumbarse. También estaba planeado que utilizase cinemática inversa a la hora de mover los motores, pero al final solo se pudo usar para generar las posiciones iniciales que luego se necesitaría realizar un ajuste fino sobre el propio robot.
De la misma manera, en el estado final del prototipo y por falta de piezas, necesita usar dos fuentes de alimentación en vez de poder alimentar todo de la misma.

El resto de los objetivos que corresponden a diseñar una aplicación que interactúe con el robot, realizada con la web del MIT llamada APP inventor, incluir sensores que reaccionen en tiempo real (voltímetro, amperímetro y ultrasonidos) y modifiquen el comportamiento del código, diseñar todo el circuito eléctrico capaz de hacer funcionar el sistema (salvo la resistencia que falta pero con el fallo localizado) y la documentación del proceso de investigación y creación del

prototipo, han sido completadas excediendo el tiempo designado para este proyecto pero dentro de unos límites personales (podía invertir ese tiempo en aprendizaje y desarrollo personal).

El tiempo real no está contabilizado dado que este proyecto lleva en desarrollo un año compaginando otras tareas ajenas con la búsqueda de información e investigación, y calcularlo, sobre todo en las primeras fases de proyecto, era difícil.

8 Conclusiones

Como se ha podido ver a lo largo de este documento, un proyecto de investigación lleva mucho más trabajo del que inicialmente se puede prever en la mayoría de los casos. La cantidad de problemas que pueden surgir incluso de tareas que se creía tener controladas han causado que la extensión del tiempo del proyecto aumentase en gran medida, y por ese metido no lograr llegar a todas las metas inicialmente propuestas. Lo ideal es tener una buena organización con una buena metodología y ser capaz de prever estos problemas en la medida de lo posible.

Durante este proyecto han sido realizadas tareas de diseño, mecánica, electrónica, electricidad y programación, de las cuales únicamente la parte de programación fue aprendida durante la carrera. El resto de los campos fue estudiado de forma superficial por lo que hubo que hacer una investigación extensa por cuenta propia para poder tener un prototipo funcional que poder mostrar.

Diseñar un producto funcional para un gran público es extremadamente complejo, hay muchas variables a tener en cuenta (y más si va orientado a gente joven) pero el ser capaz de sacar algo hacia adelante, desde iniciar el proyecto con unas ideas previas hasta conseguir tener un prototipo que puede comercializarse permite tener una visión amplia de todo el entorno profesional.

Como resultado del robot se han presentado algunos problemas que no se han podido resolver, pero también se han presentado muchos que si se han solucionado, como por ejemplo, se ha conseguido que los ejes de los motores agarren mejor gracias a los engranajes de metal. También se ha conseguido que el relé de alimentación funcione en el 100% de escenarios y, lo más importante, encontrar una placa Arduino capaz de alojar todas las características que se necesitaban en cuestión de salidas disponibles y memoria. Se puede ver de forma más exhaustiva estas soluciones y mejoras tomadas durante el desarrollo en el apartado 7.3 (pruebas, problemas y posibles mejoras durante el desarrollo).

También se han conseguido diseñar tres kits de diferentes dificultades y niveles de forma teórica, cada uno de ellos pensando en la audiencia que pueden tener y relacionando esa dificultad con los problemas encontrados durante el desarrollo. No han podido ser fabricados cada uno de ellos por falta de presupuesto y exceso de horas. Aunque estos kits están repartidos por edades, cualquier persona independiente de la edad que tengan pueden adquirir los kits y aprender con ellos según su nivel de conocimientos en el ámbito tecnologico.

El desarrollo del robot continuará cuando haya más presupuesto para intentar lograr las metas propuestas inicialmente y mejorarlas con lo propuesto en el apartado 7.3 (pruebas, problemas y posibles mejoras durante el desarrollo) tales como modificar el diseño del robot para darle más rigidez a las juntas y engranajes, usar materiales de mayor calidad, fabricar la parte baja de las patas en caucho, utilizar placas más avanzadas como una raspberry, añadir el relé de corte de alimentación a los motores...

9 Tablas y figuras

Tabla 1: Tabla comparativa entre RUP, SCRUM y XP (SumariMiguel) 18
Tabla 2: Tiempo asignado a cada conjunto de tareas 19
Tabla 3: Inventario y presupuesto de piezas 47
Tabla 4: Estimación de coste por kit 48

Figura 1: Spot Mini de Boston Dynamics 13
Figura 2: Atlas de Boston Dynamics 13
Figura 3: OpenCat de Petoi 13
Figura 4: Spot micro 14
Figura 5: Spot micro de Michael Kubina 14
Figura 6: Metodologías Ágiles frente a tradicionales 17
Figura 7: Tablero de tareas pendientes, en curso y terminadas 19
Figura 8: Distribución de tareas a lo largo del tiempo 1 20
Figura 9: Distribución de tareas a lo largo del tiempo 2 21
Figura 10: Modelo 3D de spot micro 22
Figura 11: Entorno de desarrollo del proyecto. 22
Figura 12: Eje del hombro/cadera 23
Figura 13: Pata trasera 23
Figura 14: Frontal con sensores de ultrasonido y leds 24
Figura 15: Vista superior 24
Figura 16: Esquema eléctrico 25
Figura 17: Divisor de voltaje 26
Figura 18: Tabla relacional entre voltaje de la batería y su nivel de carga (Ampow, s.f.) 26
Figura 19: Amperímetro ACS712 26
Figura 20: Relé 27
Figura 21: Transformador SZBK07 27
Figura 22: Controladora PWM PCA9685 27
Figura 23: Servomotor DS3218 PRO 28
Figura 24: Especificaciones servomotores 28
Figura 25: Buzzer 28
Figura 26: Pantalla OLED SSD1306 28
Figura 27: Esquema de conexiones Arduino Mega 29
Figura 28: Especificaciones técnicas Arduino Mega 29
Figura 29: Bluetooth HC-05 29
Figura 30: Ultrasonidos HC-SR04 29
Figura 31: Funcionamiento de una señal PWM 30
Figura 32: Relación entre una señal PWM y giro del motor 30
Figura 33: Protocolo I2C 31
Figura 34: Conexión en serie contra paralelo 32
Figura 35: Pasos de un gato 33
Figura 36: Opciones de descarga de Arduino 33
Figura 37: Seudocódigo de la función loop 36
Figura 38: Máquina de estados básica 38
Figura 39: Función de la app que añade la S a la secuencia 39
Figura 40: Función que añade 0s y convierte "sequence" en el array "splitSequence" en el robot 39
Figura 41: Interfaz principal de diseño de MIT App inventor 39
Figura 42: Código de cambio de ventana 40
Figura 43: Pantalla de selección de instrucciones 40
Figura 44: Pantalla de control remoto 40
Figura 45: Pantalla de secuencias 41
Figura 46: Código para enviar las secuencias por bluetooth 41

FIGURA 47: PCB DE CONEXIONES DE IMPRESORA 3D 43
FIGURA 48: JUNTA ENTRE LA PATA Y EL CUERPO DEL ROBOT 50
FIGURA 49: ENGRANAJES DE PLÁSTICO ROTO Y METAL 51
FIGURA 52: DISEÑO FINAL DE LOS ENGRANAJES DE LOS EJES 51
FIGURA 51: PIEZA CON SOPORTE EN ESTRELLA 51
FIGURA 50: PROPUESTA DE CAMBIO DEL EJE DEL HOMBRO 52
FIGURA 53: ALMOHADILLA DE LÁTEX 52
FIGURA 54: COMPARATIVA DE DIFERENTES PROCESADORES ARDUINO 53
FIGURA 55: ESPECIFICACIONES RELÉ RLP12V 53
FIGURA 56: TRASERA DE PCA9685 CON ALIMENTACIÓN PUENTEADA 54
FIGURA 57: PLACA BASE PARA PROTOTIPOS CON CABLEADO 55
FIGURA 58: CÓDIGO DEL VOLTÍMETRO Y AMPERÍMETRO 55

10 Glosario de siglas

- AWG: American wire gauge.
- CNC: Computer numerical control.
- GND: Ground.
- I2C: Inter-Integrated Circuit.
- IDE: Integrated development environment.
- LED: Light Emitting Diode.
- OLED: organic light-emitting diode.
- PCB: Printed Circuit Board.
- PLA: Polylactic acid.
- PWM: Pulse Width Modulation.
- SCL: System clock.
- SDA: System data.
- TTL: Transistor-Transistor Logic.
- UART: Universally Asynchronous Reciver/Transmitter.
- USB: Universal Serial Bus.

11 Bibliografía

AllDataSheet. (s.f.). Obtenido de https://www.alldatasheet.com/view.jsp

Ampow. (s.f.). *Lipo Voltage Chart: Show the Relationship of Voltage and Capacity*. Obtenido de https://blog.ampow.com/lipo-voltage-chart/

Arduino. (s.f.). *Arduino*. Obtenido de https://store.arduino.cc/products/arduino-uno-rev3

Arduino. (s.f.). *Arduino Code Online*. Obtenido de https://create.arduino.cc/editor

Arduino. (s.f.). *Arduino forum*. Obtenido de https://forum.arduino.cc

Arduino. (s.f.). *Arduino Mega 2560 rev3.* Obtenido de http://store.arduino.cc/products/arduino-mega-2560-rev3

Arduino. (s.f.). *Arduino reference*. Obtenido de https://www.arduino.cc/reference/en/

Arduino, A. (s.f.). *AprendiendoArduino*. Obtenido de https://aprendiendoarduino.wordpress.com/2015/03/26/lenguaje-de-programacion-de-arduino-estructura-de-un-programa/

BRACCIO, T. (s.f.). *Mouser Electronics*. Obtenido de https://www.mouser.es/ProductDetail/Arduino/T050000?qs=5aG0NVq1C4xeqskidzhl7g%3D%3D

Bruton, J. (s.f.). *xRobots*. Obtenido de http://www.xrobots.tech

Dynamics, B. (s.f.). *BostonDynamics*. Obtenido de https://www.bostondynamics.com

Dynamics, B. (s.f.). *Youtube*. Obtenido de https://youtu.be/fn3KWM1kuAw

Gudino, M. (s.f.). *arrow*. Obtenido de https://www.arrow.com/en/research-and-events/articles/arduino-uno-vs-mega-vs-micro

Keyestudio. (s.f.). *Keyestudio mini tank v3.0*. Obtenido de https://www.keyestudio.com/products/keyestudio-diy-mini-tank-v30-smart-robot-car-kit-for-arduino-robot-car-stem

Keyestudio. (s.f.). *Keyestudio Self balancing Car*. Obtenido de https://www.keyestudio.com/products/keyestudio-self-balancing-car-kit-for-arduino-robot-1

Kiwiko. (s.f.). *Kiwico*. Obtenido de https://www.kiwico.com

Kubina, M. (s.f.). *Github*. Obtenido de https://github.com/michaelkubina/SpotMicroESP32

Llamas, L. (s.f.). *Comunicacion de arduino con puerto serie*. Obtenido de https://www.luisllamas.es/arduino-puerto-serie/

Llamas, L. (s.f.). *El bus I2C en arduino*. Obtenido de https://www.luisllamas.es/arduino-i2c/

Mechatronics, N. (s.f.). *Configuracion del modulo bluetooth HC-05 usando comandos AT*. Obtenido de https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usando-comandos-at.html

Mechatronics, N. (s.f.). *Tutorial uso de servomotores con arduino*. Obtenido de https://naylampmechatronics.com/blog/33_tutorial-uso-de-servomotores-con-arduino.html

MIT. (s.f.). *App inventor*. Obtenido de http://appinventor.mit.edu

PetoiCamp. (s.f.). *Github*. Obtenido de https://github.com/PetoiCamp/OpenCat

PlainConcepts. (s.f.). *PlainConcepts*. Obtenido de https://www.plainconcepts.com/es/spot-boston-dynamics-robot/

RootSaid. (s.f.). *Arduino, PCA9685 Servo Motor Driver 16 Chanel Module Tutorial*. Obtenido de https://rootsaid.com/pca9685-servo-driver/

SowoD. (s.f.). *Devianart*. Obtenido de https://www.deviantart.com/sowod/art/How-to-Cat-Walk-Cycle-682254186

StackOverflow. (s.f.). *https://stackoverflow.com*. Obtenido de https://stackoverflow.com/questions/3683602/single-quotes-vs-double-quotes-in-c-or-c

SumariMiguel. (s.f.). *Metodologias Agiles.* Obtenido de http://sumarimiguel.blogspot.com/2017/11/cuadro-comparativo-ente-la-metologias.html

12 Anexos

Código final cargado en el Arduino:

- Robot_Code

Aplicación Android:

- APK
- Proyecto importable .aia

Hojas de datos:

- ACS712 Datasheet
- HC-05 Datasheet
- HC-SR04 Datasheet
- MEGA2560_Rev3e_sch
- PCA9685 Datasheet
- RLP12V Datasheet
- SSD1306 Datasheet
- SZBK07 Datasheet
- Voltage Sensor-170640_SGT Datasheet

Códigos de testing de Arduino:

- Bluetooth_test_config
- Bluetooth_test_control
- Error_codes
- Kinematics_Test
- Oled_Test
- PWM_Servo_Test
- Servo calibration
- Test_1_servo_cinematica_inversa
- Test_servos_input_console
- TestWalking
- Ultrasonic_Sensor_Test
- Voltage_Sensor_Test

Actas de reuniones.

Propuesta de PFG.