



Trabajo Fin de Grado
Grado en Ingeniería Informática
Tecnología Específica de
Ingeniería de Computadores

DISEÑO Y DESARROLLO DE UN ROBOT SIMILAR AL MODELO COMERCIAL MBOT

Autor: Pedro López Marín

Tutores: Francisco Manuel Delicado Martínez
Jesús Martínez Gómez

Curso 2022/23

*Dedicado a mi familia y a todos
aquellos que siempre han confiado
en mí.*

Declaración de Autoría

Yo, Pedro López Marín con DNI 49217546F, declaro que soy el único autor del trabajo fin de grado titulado “Diseño y construcción de un robot similar al modelo comercial mBot” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a 13 de febrero de 2023

Fdo: Pedro López Marín

Resumen

Esta memoria presenta el diseño y desarrollo de un robot similar al mBot. En esta solución, se busca crear un robot programable utilizando componentes estándares de bajo coste y fáciles de conseguir en cualquier tienda de venta de componentes on-line.

La motivación es demostrar como la evolución tecnológica ha permitido acceder a microcontroladores y sensores de muy bajo coste, pero que ofrecen grandes prestaciones hasta el punto de permitir la elaboración de un dispositivo tan relativamente complejo como este, por un coste mínimo y con amplias posibilidades educativas.

Junto con el robot, también se desarrolla una serie de librerías básicas para permitir la interacción con el mismo, de modo que ofrezca una API básica para poder programarlo de forma sencilla con total abstracción del hardware empleado. Durante el desarrollo de las librerías también se ha buscado siempre lograr la mayor modularidad del código, con el objetivo de que este permita cambios en el hardware sin necesidad de reescribir todo el código, así como permitir integrar nuevas funcionalidades fácilmente.

La propuesta planteada consiste en un microcontrolador ESP32 al que se le han añadido sensores y una estructura para obtener un dispositivo con las mismas prestaciones que el robot mBot, además se le ha añadido un ESP32-CAM, que permite identificar una serie de marcadores, y también se ha creado un mando usando otro ESP32 y un joystick, que permite controlar el robot a distancia.

Los resultados muestran que el prototipo es completamente funcional y cumple con los objetivos propuestos.

Agradecimientos

A mis padres, por confiar en mis capacidades y apoyarme siempre, además de darme todas las oportunidades posibles. Gracias a ellos hoy estoy aquí.

A mis hermanos, por siempre preocuparse y apoyarme y estar siempre ahí, en las buenas y en las malas.

A mis amigos, tanto los de fuera de la carrera, como los que he conocido durante la misma, por ser mi fuente de inspiración y darme fuerzas para seguir adelante y superarme día a día.

Por último, a mis tutores Francisco y Jesús, por dedicarme parte de su tiempo y hacer de guías durante este TFG.

Índice general

Capítulo 1	Introducción.....	1
1.1	Introducción.....	1
1.1.1	Sensores de bajo coste.....	1
1.1.2	Importancia de la robótica en la educación	2
1.2	Objetivos	3
1.3	Planificación temporal	3
1.4	Estructura del proyecto	4
Capítulo 2	Estado del Arte	5
2.1	Introducción.....	5
2.2	Sensores y actuadores de bajo coste.....	5
2.2.1	Sensores de luz	5
2.2.2	Sensores ultrasonidos	8
2.2.3	Sensor de joystick	9
2.2.4	Motores de bajo coste	9
2.2.5	Controladores de motor.....	11
2.3	Entornos de programación	11
2.3.1	Arduino IDE (Integrated Development Environment)	11
2.3.2	Mblock.....	12
2.4	Robots educativos	13
2.4.1	LEGO MINDSTORMS.....	13
2.4.2	NAO.....	14
2.4.3	MBOT.....	15
2.5	Marcadores fiduciaros	16
2.5.1	ArToolKit.....	17

2.5.2	ArUco.....	18
2.5.3	AprilTags.....	19
2.6	Placas de desarrollo	21
2.6.1	Comparativa diferentes placas de desarrollo para el coche.....	21
2.6.2	Placa de desarrollo para la cámara.....	23
2.7	Protocolo BLE	24
2.7.1	Roles.....	24
2.7.2	Jerarquía de datos.....	24
Capítulo 3	Metodología y Desarrollo.....	26
3.1	Introducción	26
3.2	Diseño y construcción del robot	26
3.2.1	Primera fase del prototipo.....	26
3.2.2	Segunda fase del prototipo	30
3.2.3	Tercera fase del prototipo.....	31
3.2.4	Cuarta fase del prototipo.....	32
3.2.5	Diseño final del prototipo.....	32
3.3	Desarrollo de funcionalidades	33
3.3.1	Librerías usadas	34
3.3.2	Sensores.....	34
3.3.3	Acciones.....	36
3.3.4	ESP32-CAM	38
3.3.5	Comunicación serial ESP32-ESP32CAM	40
3.3.6	Mando.....	42
3.3.7	BLE.....	43
3.4	Conclusiones	44
Capítulo 4	Experimentos y Resultados	45
4.1	Introducción	45
4.2	Experimentos de calibrado	45
4.2.1	Obtener la distancia focal.....	45
4.2.2	Determinar umbral de detección de obstáculos.....	47
4.2.3	Comprobar funcionamiento del fotorresistor.....	47
4.2.4	Determinar umbrales del joystick	47
4.2.5	Mejorar calidad imagen.....	48

4.3	Desarrollo de casos de uso para validación.....	48
4.3.1	Manejo mediante el mando BLE	48
4.3.2	Búsqueda de ArpilTAGs según orden dado	49
4.3.3	Seguimiento de una línea.....	49
4.4	Conclusiones	50
Capítulo 5	Conclusiones y Trabajo Futuro.....	51
5.1	Conclusiones	51
5.2	Trabajo futuro	52
	Bibliografía.....	55
	Anexo	59
I.1	Coste total.....	59
I.2	Glosario de definiciones	60
I.3	Imprimir ArpilTags.....	62
I.4	Programación de la ESP32-CAM	62

Índice de figuras

Figura 1. Espectro electromagnético	6
Figura 2. Fotodiodo	6
Figura 3. LDR	7
Figura 4. Esquema fototransistor	7
Figura 5. Sensor IR activo	8
Figura 6. Frecuencia del sonido	9
Figura 7. Esquema motor CC.....	10
Figura 8. IDE Arduino.....	12
Figura 9. Añadir dispositivo mBlock	12
Figura 10. Vista general interfaz mBlock.....	13
Figura 11. LEGO Robot Inventor	14
Figura 12. Robot NAO.....	14
Figura 13. mBot.....	16
Figura 14. Marcador ArToolKit	17
Figura 15. Marcador ArUco.....	18
Figura 16. Familias AprilTag.....	19
Figura 17. OpenMV-CAM	23
Figura 18. ESP32-CAM.....	23
Figura 19. Jerarquía datos BLE.....	25
Figura 20. Kit AptoFun.....	27
Figura 21. Sensor HC-SR04.....	28
Figura 22. ESP-WROOM-32 DevKit Diymore	28
Figura 23. Controlador L298N.....	29
Figura 24. Photoresistor module KY-018	30
Figura 25. Sensor sigue-líneas TCRT5000.....	30
Figura 26. Configuración sensor sigue-líneas	31
Figura 27. Joystick module.....	32
Figura 28. Vista superior	32

Figura 29. Vista de frente.....	32
Figura 30. Vista perfil	33
Figura 31. Diseño electrónico del coche realizado con fritzing.....	33
Figura 32. Clase del sensor de ultrasonidos	35
Figura 33. Clase del sensor de luminosidad	35
Figura 34. Clase sensor sigue-líneas.....	35
Figura 35. Jerarquía de funciones de Acciones	38
Figura 36. Protocolo de comunicación serial	41
Figura 37. Rangos teóricos del joystick.....	43
Figura 38. Cálculo de parámetros intrínsecos	47
Figura 39. Caso de uso de encontrar TAGs.....	49
Figura 40. Circuito seguir línea	50
Figura 41. Puente-H.....	61
Figura 42. Seleccionar placa Ai-thinker.....	63
Figura 43. Programación ESP32-CAM	64

Índice de tablas

Tabla 1. Comparativa placas desarrollo	22
Tabla 2. Mapa del joystick.....	43
Tabla 3. Resultados de distancia focal	46
Tabla 4. Coste total aproximado.....	59

Capítulo 1

Introducción

1.1 Introducción

En este capítulo, se empieza situando el contexto del proyecto. A continuación, se detallan los diferentes objetivos que se persiguen, junto con la metodología empleada para conseguirlos. Y por último se ofrece un breve resumen del contenido de la memoria.

1.1.1 Sensores de bajo coste

La constante mejora de las capacidades de miniaturización de componentes electrónicos, junto con la mejora de las técnicas de producción, están generando un aumento en la disponibilidad y accesibilidad de sensores de bajo coste de mejor calidad.

Además, el creciente interés en la robótica y el Internet de las cosas (IoT) ha impulsado un mayor desarrollo en sensores de bajo costo, con un enfoque en hacer que la robótica y el IoT sean accesibles para un público más amplio.

Otro factor importante en la evolución de los sensores de bajo costo es la mejora en la tecnología de transmisión de datos inalámbricos, lo que permite la transmisión de datos en tiempo real desde los sensores a otros dispositivos, lo que mejora la capacidad de los sensores para integrarse en sistemas más grandes.

1.1.2 Importancia de la robótica en la educación

En la educación obligatoria, se plantea que las seis capacidades fundamentales que los estudiantes deben desarrollar durante esta etapa son: Resolución de problemas, Pensamiento crítico, Aprender a aprender, Trabajo colaborativo, Comunicación y Compromiso y responsabilidad [1].

La introducción de la robótica en la escuela permite el trabajar las capacidades fundamentales de manera novedosa, proporcionando un enfoque práctico y lúdico del aprendizaje, y en el que el alumno adquiere un papel más activo en la resolución de los problemas planteados, mediante el diseño y la experimentación, lo cual se traduce en un aprendizaje de mayor calidad.

Otra ventaja que hace que hace la robótica una excelente herramienta de aprendizaje es el feedback que obtienen del robot. La observación de forma física e instantánea de los resultados les permite una autoevaluación, haciendo que cada alumno pueda avanzar a su propio ritmo mientras se estimulan sus habilidades autodidactas. Estos resultados instantáneos también mejoran el autoestima y su afán de superación, al ir resolviendo pequeños problemas poco a poco y ver cómo funcionan sus soluciones, lo que al mismo tiempo también les ayuda a mejorar su tolerancia a la frustración.

Muchos padres e incluso profesores pueden plantearse si introducir la robótica desde primaria es ir demasiado rápido, sin embargo, Alfredo Pineda, profesor de informática de universidad y coordinador del curso de verano de la UPNA (Universidad Politécnica de Navarra) sobre el empleo de robots en los colegios, señala que[2]:

“La robótica educativa ayuda a los alumnos a razonar; eso vale para Informática y para Filosofía”.

“No queremos que nuestros críos, o los adultos, sean robots, sino que sepan trabajar con ellos. Necesitamos movernos en la vida mediante botones, y hay que alimentar la curiosidad de los niños, que sepan qué funciona al apretar ese botón...”

En general, la robótica tiene el potencial de inspirar a los estudiantes a aprender, a participar más en su educación, a pensar más críticamente y a estar más preparados para la futura fuerza laboral.

1.2 Objetivos

Los objetivos específicos del proyecto son los siguientes:

- Diseño y construcción de un robot utilizando sensores y actuadores de bajo coste.
- Desarrollo de librerías que permitan dotar de cierta funcionalidad al robot construido.
- Despliegue y validación, a través de un caso de uso, de un comportamiento que utiliza el robot físico y las librerías desarrolladas previamente.

1.3 Planificación temporal

En este apartado se detalla la planificación temporal que se ha seguido durante el desarrollo del presente trabajo:

1. Búsqueda de información sobre el mBot y otros dispositivos similares y planteamiento de las funciones necesarias para la API.
2. Ensamblaje de las partes y desarrollo de funciones necesarias para ejecutar acciones de movimiento, con control de obstáculos.
3. Búsqueda de información sobre distintos tipos de sistemas fiduciaros y montaje y desarrollo del resto de sensores.
4. Implementación de un protocolo de comunicación por el puerto serie, entre el ESP32-cam y el ESP32 (incluido también el ensamblaje de la cámara).
5. División del código en diferentes librerías y mejora del protocolo de comunicación.
6. Implementación de la comunicación por BLE y desarrollo de comportamientos complejos para validación de casos de uso.
7. Documentación del proyecto y redacción de la memoria.

1.4 Estructura del proyecto

La memoria está estructurada en 5 capítulos, cuyos temas principales son:

- Capítulo 1-Introducción. Se realiza una introducción al proyecto, en la cual se explica la motivación de este proyecto, junto con los objetivos planteados.
- Capítulo 2-Estado del arte. Se recoge toda la información, material y conocimiento recopilados a la hora de empezar a realizar este proyecto.
- Capítulo 3-Metodología y desarrollo. Se explica el proceso de construcción del robot. Empezando por la parte de hardware y siguiendo por la de software.
- Capítulo 4-Experimentos y resultados. Se exponen diferentes casos de uso y las pruebas realizadas para validarlos.
- Capítulo 5-Conclusiones y trabajo futuro. Se exponen las conclusiones personales y una lista de propuestas de mejora para futuro.

Capítulo 2

Estado del Arte

2.1 Introducción

En este capítulo se detallará el hardware, software y conocimientos empleados durante el desarrollo de este proyecto. Incluye una descripción del hardware existente (sensores, microcontroladores...) necesario para el desarrollo, también se mostrarán trabajos relacionados y el software empleado, ya sea entorno de programación o librerías.

2.2 Sensores y actuadores de bajo coste

Esta sección incluye una recopilación de los diferentes tipos de sensores existentes, que son aplicables a este proyecto, describiendo sus características generales y los fundamentos teóricos en los que se sustentan.

2.2.1 Sensores de luz

El espectro visible es la horquilla del espectro electromagnético (longitud de onda de entre 400nm y 750nm aprox., como se muestra en la *Figura 2*) que el ojo humano es capaz de percibir y traducir en los diferentes colores. Por debajo del límite inferior, tenemos la luz ultravioleta y por encima del límite superior, tenemos la luz infrarroja [3].

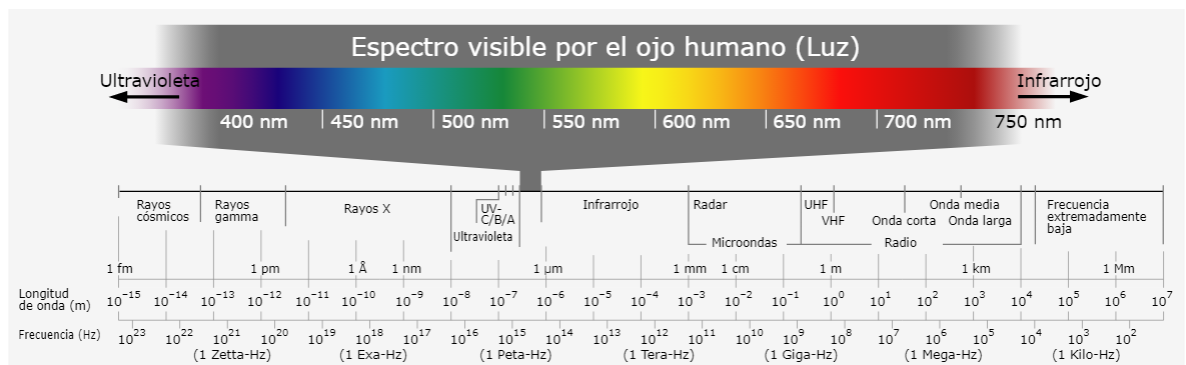


Figura 1. Espectro electromagnético

2.2.1.1 Sensor de luminosidad

Estos sensores, también llamados fotosensores o fotodetectores, trabajan en el espectro de luz visible. Son sensores pasivos capaces de convertir esa energía de la luz (fotones) en una salida de señal eléctrica (electrones)[4].

Según su construcción, los fotosensores se clasifican en:

- Los fotodiodos. Convierten la luz en corriente eléctrica. Son similares a un diodo rectificador (formado por una Unión PN), pero con la característica de que son sensibles a la luz. Ver *Figura 2*.



Figura 2. Fotodiodo

- Los fotorresistores. Son dispositivos que varían su resistencia interna al exponerse a la luz. Normalmente cuando la intensidad de la luz aumenta la resistencia disminuye y viceversa. El fotorresistor convencional es el LDR (Resistencia Dependiente de la Luz, por sus siglas en inglés), ver *Figura 3*. Es importante tener en cuenta que la resistencia del fotorresistor se ve afectada por el tipo de material del que está hecho, la temperatura del entorno y la longitud de onda de la luz a la que está expuesto.



Figura 3. LDR

- Los fototransistores. Estos dispositivos, conmutan o amplifican las señales. Son un tipo de transistor sensible a la luz, similar a los transistores normales. La corriente aplicada a los terminales se genera con la exposición a la luz[5]. Ver *Figura 4*, donde se muestra un esquema de su construcción.

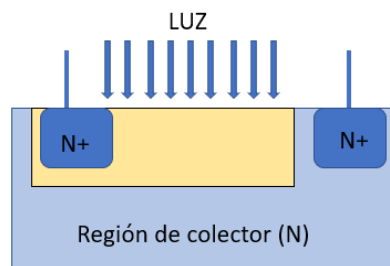


Figura 4. Esquema fototransistor

- Células fotovoltaicas. Estos generan electricidad a partir de la luz. Este proceso conocido como recolección de energía se produce gracias al efecto fotovoltaico de los componentes semiconductores de la célula.

2.2.1.2 Sensor de Infrarrojos

Como su nombre indica, los sensores de infrarrojos son capaces de detectar la luz infrarroja y convertirla en una señal eléctrica que pueda ser interpretada por un sistema de control o un dispositivo de medición.

Existen 3 tipos de sensores de infrarrojos en función de su funcionamiento interno[6]:

- Distribución espectral. Basados en el concepto de espectro visible, miden las perturbaciones en un rango determinado a través de las ondas electromagnéticas gracias al análisis gráfico de las longitudes de dicha onda.
- Sensores activos. Estos sensores disponen de un emisor (normalmente un diodo led infrarrojo) y un receptor (fototransistor) conectados. El emisor distribuye una señal, que rebota en los objetos y la capta el receptor, quien espera la señal para

la que está programado, ver *Figura 5*. Se suelen usar para detectar obstáculos, por ejemplo, en robots o vehículos autónomos.

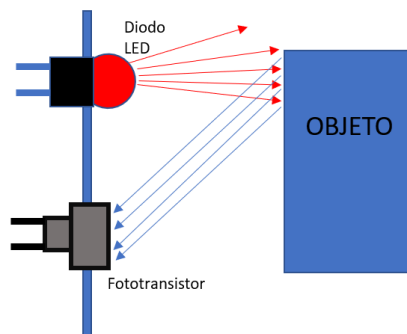


Figura 5. Sensor IR activo

- Sensores pasivos. Están formados únicamente por el fototransistor, por tanto, solo son capaces de medir las radiaciones que provienen de los objetos. Se suelen emplear en detección de movimiento, como, por ejemplo, los sistemas de seguridad, cuando un objeto que se mueve entra en el rango del detector, este es capaz de detectar la variación de los niveles IR.

2.2.2 Sensores ultrasonidos

El sonido está producido por pequeñas variaciones de presión en el medio (normalmente el aire). El sonido tiene dos características principales: la amplitud y la frecuencia[7].

La amplitud es la potencia o magnitud de esas variaciones de presión, es lo que conocemos como el volumen, es decir, cuanto mayor sea la amplitud, más fuerte será la sensación de sonido (volumen más alto). Como el rango de amplitudes es muy amplio se usa una escala logarítmica cuyas unidades son los decibelios (db). La mínima amplitud que podemos percibir es 0dB y la máxima que podemos soportar es 120dB, a partir de ese rango se producen daños irreversibles en el oído.

La otra propiedad es la frecuencia, que indica la velocidad de las variaciones de presión. Esta propiedad determina el tono y se mide en Hercios (Hz), 1Hz equivale a una onda por segundo. El rango que es capaz de percibir el oído humano es de 20Hz a 20KHz, como podemos ver en *Figura 6*. Por debajo tenemos los infrasonidos y por encima de los 20KHz tenemos los ultrasonidos.

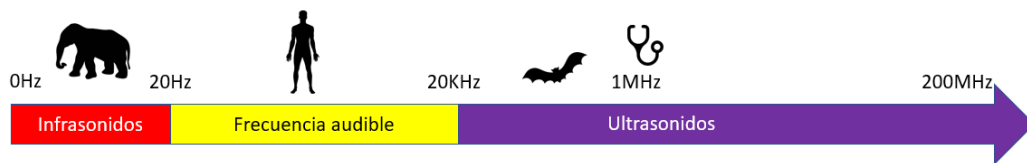


Figura 6. Frecuencia del sonido

Un sensor de ultrasonidos se compone de un emisor, que lanza una señal de ultrasonidos (por encima de 20kHz), esta señal rebota en un objeto y regresa al sensor como un eco y es capturada por el receptor. Utilizando la velocidad del sonido y el tiempo transcurrido en recibir el eco, calcula la distancia con la fórmula:

$$\text{Distancia (m)} = \{\text{tiempo del pulso (ECO)} * \text{Velocidad sonido (340m/s)}\} / 2$$

2.2.3 Sensor de joystick

El objetivo de un joystick[8] es traducir su posición en el eje X y el eje Y, en una señal eléctrica, para que un microcontrolador la pueda procesar. Para lograrlo, emplea dos potenciómetros de 5k para cada eje, conectados mediante un Mecanismo Gymbal.

El joystick emite una señal analógica de entre 0v y 5v, para cada eje, en función de su posición. En reposo, cuando está centrado, su voltaje es aproximadamente la mitad, 2.5v.

Este voltaje de salida lo captan los pines multiplexados con el ADC de nuestra placa para determinar la posición física del joystick. Los valores variaran en función de la resolución de nuestro microcontrolador, de 0 a $2^{\text{resolución}}$.

Algunos también tienen un mecanismo de botón que equivale al eje Z. Para este caso en una salida digital.

2.2.4 Motores de bajo coste

Un motor es un dispositivo capaz de transformar la energía eléctrica en energía mecánica.

Un motor eléctrico está formado por dos partes principales, comunes a todos los tipos, una parte estática denominada estator y una parte móvil, denominada rotor.

El principio de funcionamiento de estos motores es que, si un conductor por el que circula una corriente eléctrica se encuentra dentro de la acción de un campo magnético, este tiende a desplazarse perpendicularmente a la acción del campo magnético [9].

Cuando está funcionando, la corriente eléctrica suministrada al motor se emplea para generar campos magnéticos tanto en el estator como en el rotor. Estos campos, que son opuestos entre sí, se empujan, haciendo que el rotor experimente un par y como resultado gire.

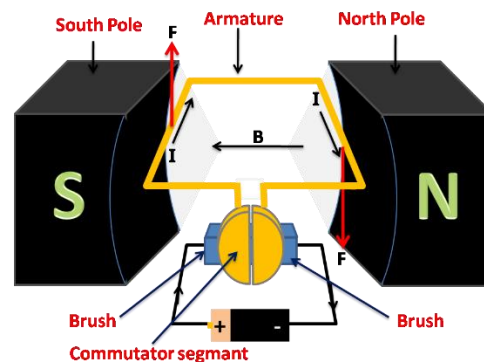


Figura 7. Esquema motor CC

Existen dos tipos de motores según la fuente de energía usada: motores de corriente alterna (CA) y motores de corriente continua (CC o DC). Nos centraremos en el de corriente continua, ya que es el más usado para este tipo de proyectos de robótica y para juguetes.

Un motor de corriente continua utiliza como estator un imán permanente, ver *Figura 7*. Algunas características son:

- Tiene un par de arranque elevado, es decir, al arrancar posee una gran fuerza.
- Es reversible, permite cambiar el sentido de giro invirtiendo la polaridad de su conexión. Aquí es donde gana importancia la configuración de transistores de Puente – H.
- Podemos controlar su velocidad. La velocidad aumenta con la tensión. Para ello se usa la técnica del Señal PWM. Esta característica, es la que hace que se prefiera este tipo de motores en proyectos de robótica y juguetes, pues el de corriente alterna, no permite variar la velocidad.

2.2.5 Controladores de motor

La función de los driver de motor es la de hacer de amplificador del voltaje. Por lo general los microcontroladores, no disponen de potencia suficiente para mover los actuadores. Su función es “ordenar” a los controladores que hagan este trabajo.

La mayoría de estos controladores tienen una tipología de Puente – H, uno para cada salida del motor, esta tipología nos permite controlar el sentido de la corriente.

2.3 Entornos de programación

2.3.1 Arduino IDE (Integrated Development Environment)

Es un entorno de desarrollo que ofrece un conjunto de herramientas software para permitir a los programadores desarrollar, depurar, cargar y ejecutar nuestros programas (llamados “sketches”) en placas Arduino[10]. Ver *Figura 8*.

Aunque el enfoque inicial fue para trabajar con placas Arduino, gracias a ser open-source y a la comunidad, podemos emplearlo para trabajar con otros tipos de placas como es en nuestro caso las ESP32 y ESP32-cam. Usa un lenguaje de programación de alto nivel propio, denominado Processing, el cual está basado en C++, por lo que su sintaxis es muy similar.

Este entorno se puede ampliar mediante el uso de librerías, al igual que la mayoría de entornos de programación. Estas librerías brindan funcionalidades adicionales para usar en los Sketch, por ejemplo, para trabajar con hardware o manipular datos. Existen librerías oficiales como por ejemplo para los protocolos SPI, I2C y UART y luego podemos incluir bibliotecas de terceros o incluso diseñadas por nosotros mismos.

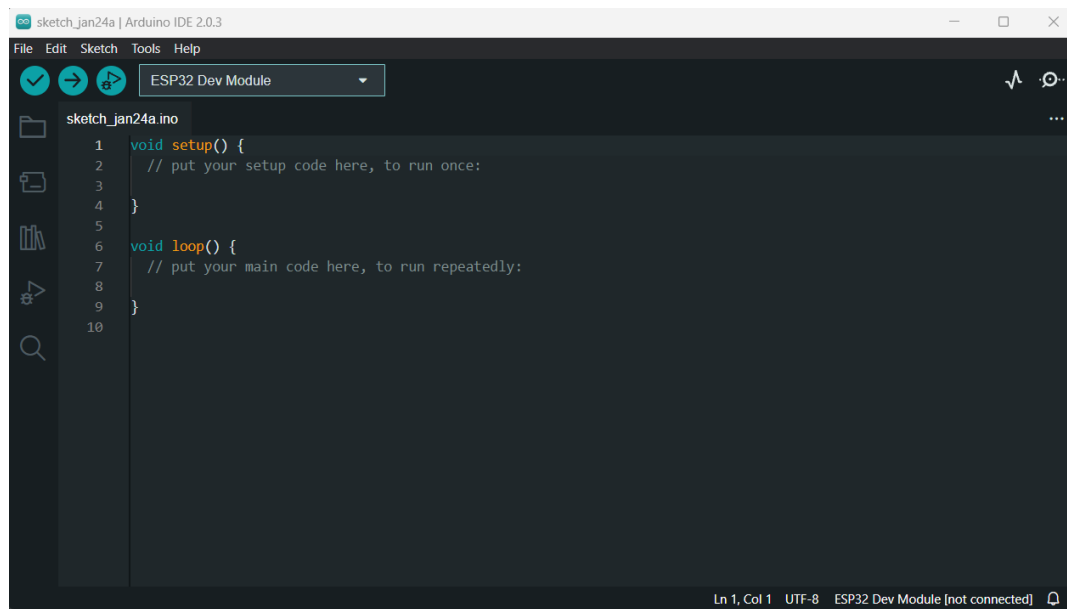


Figura 8. IDE Arduino

2.3.2 Mblock

Es un entorno gráfico de programación creado por la empresa MakeBlock. Está basado en Scratch 2.0, pero enfocado para programar robots, entre ellos el mBot. La interfaz es muy similar a la de Scratch. Para programar algún dispositivo, debemos seleccionarlo: pestaña Dispositivos -> añadir, nos aparece una ventana como en *Figura 9*.

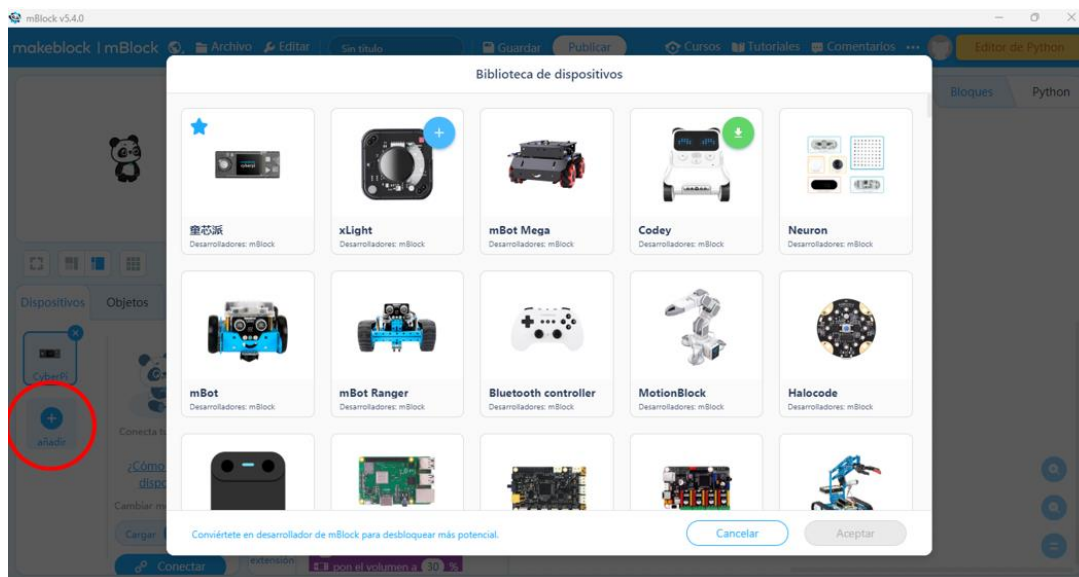


Figura 9. Añadir dispositivo mBlock

Una vez añadido el dispositivo, cuando lo seleccionamos nos aparecen sus instrucciones. Podemos ver en la columna central una serie de pestañas, que clasifican todas las instrucciones.

Como podemos ver en la *Figura 10*, podemos programar usando los bloques predefinidos o en C.

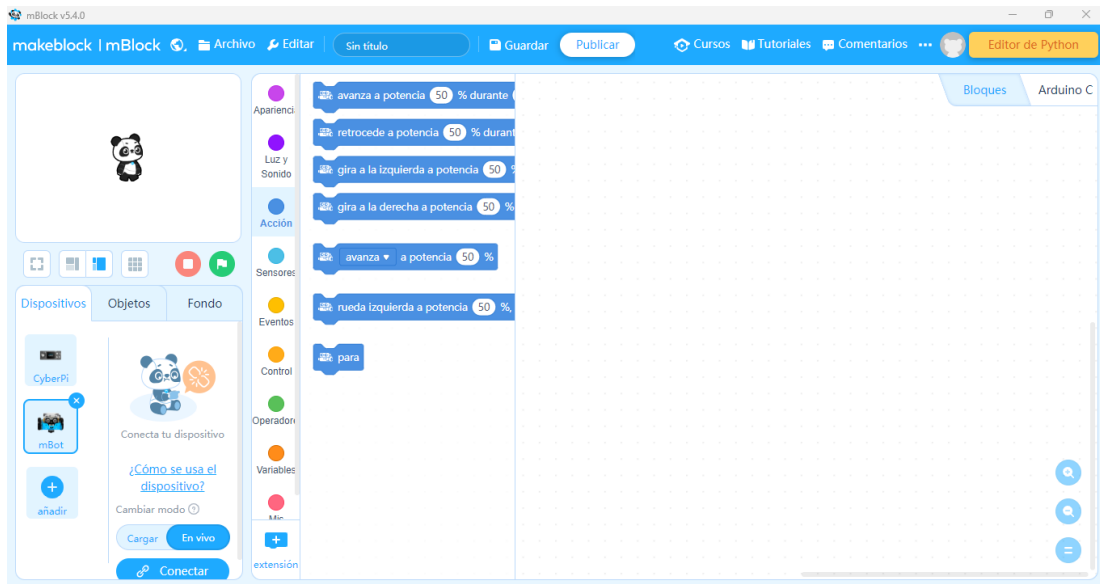


Figura 10. Vista general interfaz mBlock

2.4 Robots educativos

Existe una gran variedad de robots educativos, desde robots prefabricados, hasta kits que permiten construir los robots con la forma que queramos y poder programarlos. En esta sección, se ha recogido los modelos más representativos de ambos tipos:

2.4.1 LEGO MINDSTORMS

Es una línea de robótica para niños perteneciente a la empresa LEGO[11]. Permite crear robots uniendo bloques de LEGO con actuadores, ruedas y sensores que se controlan desde el bloque programable. Este bloque es el que incluye el procesador y el puerto para programarse.

Usa un entorno de programación gráfico, basado en la interfaz de Robolab. Además, no se requieren conocimientos electrónicos ni eléctricos, por lo que resulta muy adecuado para los más pequeños.

Esta línea incluye varios sets diferentes e incluso su versión diseñada para la docencia.

Actualmente en la página de LEGO se pueden adquirir dos kits diferentes: LEGO MINDTORMS EV3 y Robot Inventor, véase en *Figura 11*.



Figura 11. LEGO Robot Inventor

2.4.2 NAO

Es el primer robot creado por la empresa Aldebaran Robotics. NAO es un robot humanoide programable de 58cm de altura y 4.3kg, véase en *Figura 12*. Fue lanzado en 2006 y desde entonces ha ido evolucionando, su última versión fue lanzada en 2018, el NAO v6.

Este robot se puede programar para diversas funciones, aunque su principal enfoque es para fomentar el desarrollo y la interacción entre personas y máquinas. Por este mismo motivo es capaz de reconocer rostros de personas y detectar sus emociones, a la vez que escucha y responde. Puede hablar 19 idiomas diferentes

En cuanto a la programación incorpora una interfaz gráfica, de arrastrar y soltar bloques, para poder programarlo sin conocimientos previos de programación. Sin embargo, también es posible programarlo en otros lenguajes más avanzados como C++, Python, Java o Matlab.



Figura 12. Robot NAO

2.4.3 MBOT

Mbot es un robot educativo STEAM (Science, Technology, Engineering, Arts and Mathematics). El robot está recomendado para niños a partir de 8 años, el objetivo es que los niños aprendan programación, electrónica y robótica de forma práctica interactuando con el dispositivo a modo de juguete.

Este robot está diseñado por Makeblock, una empresa especializada en crear este tipo de kits de robótica. El dispositivo está compuesto por una serie de componentes modulares que se pueden ensamblar fácilmente y programar mediante una gran variedad de lenguajes de programación, entre los que se incluyen Python, C++ y Scratch.

Cabe destacar que es un proyecto open-source, lo que ha permitido que la comunidad cree más kits de expansión, para mejorar las capacidades de este. El kit básico, véase en *Figura 13* tiene un precio de unos 90 euros y está compuesto por:

- **Partes mecánicas.** Son las piezas que permiten el movimiento y las que forman la estructura del robot. El robot está diseñado para que sea fácil de montar y las piezas mecánicas se pueden conectar mediante tornillos y conexiones a presión. En esta categoría se incluyen el chasis, las ruedas, la batería recargable y las demás piezas que forman el cuerpo.
- **Electrónica.** La placa de control principal está basada en Arduino UNO, aunque con modificaciones para adaptarla a las necesidades de este kit, la placa ha sido rebautizada como mCore.

Dispone de una serie de sensores que permiten la interacción con el entorno, entre los que destacan:

- Un sensor de luminosidad.
- Un sensor de ultrasonidos.
- Receptor de Infrarrojos, para la comunicación con el mando.
- Sensor sigue-líneas. Devuelve 0 si detecta línea (negra) y 1, si no (blanco de fondo).
- Modulo bluetooth, permite conectarlo al smartphone, PC o tablet para controlarlo y programarlo.

- Módulo de WiFi 2.4G, tiene el mismo objetivo que el módulo Bluetooth, pero este es útil cuando hay varios mBot juntos, para evitar interferencias.

También son necesarios una serie de actuadores, para permitir el movimiento y poder reaccionar a las señales de entrada de los sensores, estos actuadores son:

- Leds
- Buzzer, para emitir pitidos.
- Motores

Por último, mencionar los conectores para facilitar la conexión de otros componentes con la placa:

- Conectores RJ45. Similares a las habituales clavijas de teléfono, permiten conectar de forma sencilla componentes como el sensor de ultrasonidos o el sigue-líneas.
- Clavijas para los motores.
- **Software.** Para programar este dispositivo, se pueden emplear diferentes lenguajes de programación, pero está especialmente diseñado para usar la herramienta Mblock.



Figura 13. mBot

2.5 Marcadores fiduciaros

El proceso de detección de objetos en la escena es un procedimiento que requiere un gran poder computacional, o en su defecto mucho tiempo. Especialmente para dispositivos de bajo coste, donde el poder computacional o la calidad de imagen de la cámara es reducida, el uso de marcadores fiduciaros nos permite compensar dichas

carencias. Estos marcadores, además de ofrecernos un patrón único para su reconocimiento, proporcionan un medio para una estimación bastante confiable de su pose con respecto de la cámara.

Gracias a estas propiedades, resultan apropiados para aplicaciones de navegación, robótica, realidad aumentada, etc. Por otro lado, tendríamos marcadores como los códigos de barra o los QR, los cuales no son tan precisos para la pose, porque su aplicación está enfocada en la codificación de más información. Algunos de estos marcadores son:

2.5.1 ArToolKit.

Fue uno de los pioneros en este ámbito. Es un sistema de códigos cuadrados. Tiene la ventaja de que nos permite crear etiquetas con patrones más intuitivos, mediante el uso de una plantilla, ver *Figura 14*. Pero por contra, el esfuerzo computacional va aumentando conforme se añaden nuevas etiquetas en la base de datos, ya que cada marcador debe cotejarse con la base de datos completa [12].

El algoritmo de detección consiste en:

1. Capturar un fotograma.
2. Umbralización. Los píxeles que superen en intensidad al umbral se transforman en píxeles negros, el resto en blancos.
3. Se buscan todos los marcos negros, como los de las plantillas.
4. Si la forma de la plantilla y la plantilla almacenada coinciden, se usa la información de la plantilla almacenada para compararla con la de la detectada y así poder obtener la información de su pose, se guarda en una matriz.



Figura 14. Marcador ArToolKit

2.5.2 ArUco

Es otro tipo de marcador cuadrado. Compuesto por un borde negro y una matriz binaria interna, que lo identifica, véase en *Figura 15*. El borde negro facilita su detección y la codificación binaria permite su identificación y corrección de errores. El tamaño del marcador determina el tamaño de la matriz. Por ejemplo, para un tamaño de 4x4, tendríamos 16bits.

En este sistema es importante el concepto de diccionario. Un diccionario, es la lista de codificaciones binarias de cada uno de los marcadores. Las principales propiedades de un diccionario son el tamaño (número de marcadores que lo componen), y el tamaño del marcador (tamaño de cada marcador, en bits).

El algoritmo se compone de dos pasos principales:

1. Detección de los candidatos a marcador. Se empieza por una umbralización adaptativa para segmentar los marcadores, luego se escogen los contornos de imagen y se descartan los que no tengan forma cuadrada.
2. Determinar si son marcadores. Este paso consiste en extraer los bits de cada marcador:
 - a. Se determina su rotación, para identificar cada esquina.
 - b. Se umbraliza para separar los bits blancos y los negros.
 - c. Se divide la imagen en celdas.
 - d. Se cuenta el número de píxeles (blancos o negros) de cada celda.
 - e. Se analizan los bits detectados para ver si el marcador pertenece a un diccionario.

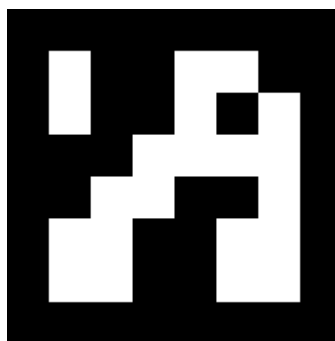


Figura 15. Marcador ArUco

2.5.3 AprilTags

Creado por la Universidad de Michigan[13], al igual que los anteriores, está basado en un sistema lexicográfico, es decir, usa diccionarios, que en este caso se denominan familias, y el software de detección es resistente a la oclusión parcial, la rotación y el escalado.

Estas familias, tienen diferentes prestaciones, para adaptarse a todo tipo de necesidades. Es importante entender la nomenclatura para poder diferenciar las familias. El primer número es el número de bits de datos, dicho de otro modo, el número de “bloques cambiables”, las familias con mayor número permiten generar más etiquetas diferentes de la misma familia. El segundo número indica la distancia de Hamming, el número mínimo de bits que se deben cambiar en el código de una etiqueta para alcanzar el código de otra etiqueta, las familias con mayor número en este caso permiten mayor tolerancia a fallos. Ver *Figura 16*.

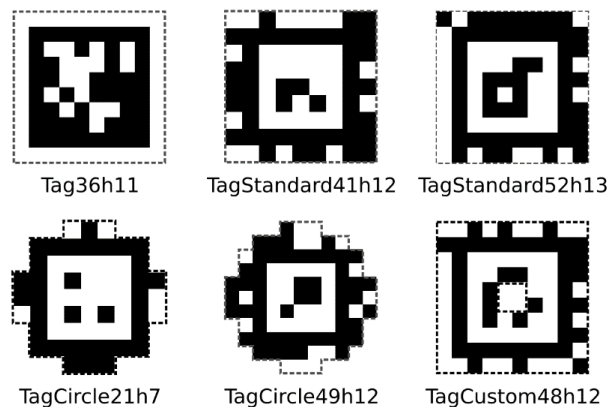


Figura 16. Familias AprilTag

Por ejemplo, para este TFG se emplean las etiquetas de la familia 25h9, indica que son etiquetas de 25 bits (matriz de 5x5) y con una distancia mínima de Hamming mínima de 9 bits entre códigos. Esta familia permite crear hasta 35 etiquetas diferentes, la elección de esta familia esta principalmente condicionada por el hardware, siendo estas las más rápidas de decodificar después de las de 19 bits que tienen mayor tasa de error.

La biblioteca AprilTag proporciona una API en C/C++ y Python y sin dependencias externas para ser fácilmente integradas en otras aplicaciones, además están diseñadas para funcionar incluso en sistemas embebidos y en tiempo real. El sistema AprilTag

puede funcionar con una amplia gama de cámaras, desde cámaras tradicionales hasta cámaras térmicas. Entre sus principales aplicaciones destacan la robótica, drones, realidad aumentada y calibrado de cámaras.

El algoritmo consta de varios pasos[13]:

1. Convierte la imagen a escala de grises de punto flotante (valores de píxel entre 0.0 y 1.0) y aplica un desenfoque Gaussiano.
2. Para cada píxel, calcula su gradiente local: magnitud y dirección.
3. Agrupa los píxeles conectados según atributos de gradiente similares. Para ello aplica un algoritmo de búsqueda en unión. Estos grupos se denominan componentes conectados.
4. Mediante una técnica de mínimos cuadrados ponderados, se ajusta la línea de cada componente, la dirección de la línea ajustada es determinada por la dirección de los gradientes. De este modo, podemos observar en cada línea un lado más oscuro junto a uno más claro.
5. Se van detectando estas líneas y se agrupan formando segmentos. Para cada segmento, se buscan segmentos que empiecen donde termina este segmento, hasta encontrar una forma cuadrada, lo que se denomina Quad. Cada quad representa el borde negro alrededor de una etiqueta candidata.
6. Se decodifican los quads detectados, mirando los píxeles dentro del borde para ver si representan un marcador válido y generar una lista de TagDetections.
7. Finalmente, se buscan las TagDetections superpuestas y se extraen las mejores (distancia de Hamming menor o mayor perímetro), desechando el resto.

El proceso del detector se puede optimizar mediante el uso de diferentes técnicas y algoritmos de procesamiento de imágenes, como el umbral, la detección de bordes y la coincidencia de patrones. La biblioteca también proporciona un conjunto de parámetros que se pueden ajustar para mejorar el rendimiento de detección en diferentes escenarios.

Existe un estudio[14], en el que se compararon los sistemas de ArUco, AprilTags y ARToolkit, las conclusiones que se pueden extraer de ese estudio son que AprilTags

ofrece mayor distancia de detección y desde un mayor rango de ángulos. A cambio es un poco más lento en el proceso de detección.

2.6 Placas de desarrollo

Las placas de desarrollo son dispositivos que integran un microcontrolador, permitiendo un fácil acceso a sus pines y ofreciendo componentes adicionales que amplíen sus capacidades, como por ejemplo interfaces USB, reguladores de voltaje, antenas...

En general este tipo de placas se caracterizan por:

- Facilidad de uso. Estas placas están diseñadas para principiantes en electrónica.
- Gran poder y capacidad. Poseen microcontroladores con una buena capacidad de procesamiento.
- Periféricos. Permiten conectar diversos tipos de dispositivos de entrada y salida, como, por ejemplo: LED, sensores de todo tipo, motores...
- Tamaño. Gracias a su reducido tamaño, se pueden integrar fácilmente en cualquier proyecto.
- Prototipos. Estas placas son reprogramables y permiten conectar cables sin tener que soldar.

2.6.1 Comparativa diferentes placas de desarrollo para el coche

En el mercado existen diversas placas de desarrollo, con diferentes características para adaptarse a todo tipo de necesidades. Para la elaboración de este proyecto, los requisitos mínimos que debe tener la placa son:

- Conectividad inalámbrica. Para poder conectar un mando a distancia. Puede valer WiFi o Bluetooth/BLE.
- Tamaño reducido. Debe ser una placa de pequeñas dimensiones que se pueda integrar fácilmente con el resto de componentes del coche.
- Pines. Debe de disponer de un considerable número de pines para poder conectar todos los sensores.
- Canales PWM. Para poder manejar la velocidad de los motores y controlar el servomotor.

Con estas características las placas que mejor se adaptan son:

	Arduino Nano RP2040 Connect [15]	Arduino Nano 33 BLE [16]	ESP32 * [17]
CPU	RP2040 dual-core ARM Cortex-M0+ (133MHz)	ATMega 328 (64MHz)	Tensilica Xtensa LX6 de 32bits dual-core (hasta 240Mhz)
RAM (KB)	264	256	520
ROM	16 MB	32KB	448KB
USB	Micro USB tipo A	Mini USB tipo B	Según la placa
GPIO	30	14	34 programables
Otras interfaces	1 x UART, 4 x ADC (12bits), 20 x PWM, 1 x SPI	1 x UART, 1 x ADC (10bits), 14 x PWM, 1 x SPI	3 x UART, 18 x ADC (12bits), 16 x PWM, 2 x DAC (10bits), 4 x SPI
Conectividad inalámbrica	WiFi: 802.11 b/g/n Bluetooth: 4.2, BLE	Bluetooth: BLE	WiFi: 802.11 b/g/n Bluetooth: 4.2, BLE
Dimensiones (mm)	43.18 x 17.78	45 x 18	Según la placa
Extras	Botón RESET	Botón RESET	10 sensores táctiles Interfaz ethernet
Precio (€)	22.80	22.80	<15, según modelo

Tabla 1. Comparativa placas desarrollo

*ESP32, para esta comparativa, se han enumerado las características de base que ofrece cualquier placa de desarrollo que incorpore este microcontrolador. Por tanto, algunas especificaciones como las dimensiones o el interfaz USB, dependerán de la placa que se seleccione.

2.6.2 Placa de desarrollo para la cámara

Para la cámara del robot, necesitamos una placa de desarrollo con soporte para un sensor de cámara. Para este caso, las dos principales opciones son:

- OpenMV Cam. Especialmente diseñada para proyectos de visión artificial. Viene programada con scripts de Python de alto nivel, lo que permite el manejo de algoritmos complejos de visión artificial. Ver *Figura 17*.

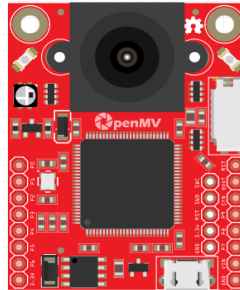


Figura 17. OpenMV-CAM

- ESP32-CAM. Esta placa incorpora el microcontrolador ESP32-S, véase en *Figura 18*, en este caso el procesador solo tiene un núcleo. En cuanto a especificaciones:
 - GPIOs: tiene menos disponibles debido a que varios de ellos se usan para conectar el sensor de cámara. En concreto dispone de 9
 - Interfaces: UART, SPI, I2C y PWM
 - Soporta dos sensores diferentes de cámara OV2640 y OV7670.
 - WiFi: Dispone una antena integrada PCB con conectividad WiFi 802.11b/g/n/e/i.
 - Bluetooth: BLE y Bluetooth 4.2.
 - RAM: 520KB SRAM, más 4MB de PSRAM
 - Voltaje de alimentación de 5V
 - No posee interfaz USB, debe programarse mediante el puerto serie.

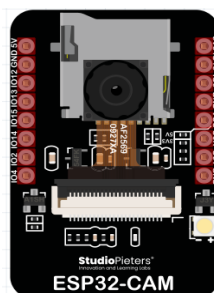


Figura 18. ESP32-CAM

2.7 Protocolo BLE

Bluetooth Low Energy[18], por sus siglas BLE, tiene sus inicios en Wibree, una tecnología de muy bajo consumo, diseñada por Nokia en 2006 con el objetivo de reemplazar a Bluetooth en aplicaciones inalámbricas de bajo consumo. Finalmente, Wibree fue incorporada al estándar principal de Bluetooth en 2010.

BLE está diseñado para tener un consumo lo más bajo posible. Opera en la misma Banda ISM que Bluetooth (2.4GHz) y comparte gran parte de su tecnología, pero aun así, son tecnologías diferentes y con objetivos de mercado diferentes.

Mientras que Bluetooth se ha centrado en mejorar la velocidad de transferencia de grandes cantidades de datos; BLE, renuncia a transferencias grandes de datos, pero a cambio logra un consumo mínimo, que lo hace adecuado para dispositivos alimentados con pequeñas baterías, como son microcontroladores o sensores.

2.7.1 Roles

- Servidor. Anuncia su presencia para que otros dispositivos (Clientes) puedan encontrarlo. Este dispositivo, es el que obtiene la información de sus sensores y la envía al cliente.
- Cliente. Este dispositivo, inicia los comandos y peticiones al servidor. Este dispositivo, es el que recibe las respuestas, indicaciones o notificaciones del servidor.

Este es el esquema general de roles, pero no son fijos, dependiendo de la complejidad de la red de conexión, es posible que un dispositivo tenga los dos roles. Por último, un cliente puede conectarse a múltiples servidores, pero un servidor, solo puede conectarse a un cliente a la vez.

2.7.2 Jerarquía de datos.

La estructura del protocolo de envío de datos de BLE, se divide en 4 niveles, ver *Figura 19*.

1. Perfil. En el nivel superior. Los perfiles están diseñados para ser usados por una aplicación u otro perfil. Cada perfil, define la estructura en la que los datos de

intercambian. La estructura se divide en dos elementos: servicios y características.

2. Los servicios. Cada perfil está compuesto de uno o más servicios para cada caso de uso. Los servicios, están formados por las características o por referencias a otros servicios. Generalmente, los servicios se centran en la información de un solo sensor. Existen varios tipos de categorías para cada tipo de datos.
3. Características. Contienen los datos reales (valores), propiedades de las características (propiedades) y opcionalmente información sobre dichos datos (descriptores).
4. Valores-propiedades-descriptores.
 - a. Valor. Datos reales
 - b. Propiedades. Un campo interpretado como flag, define el comportamiento de la característica (Broadcast, Read, Write whitout response, Write, Notify, Indicate).
 - c. Descriptores. Proporcionan metadatos de la característica.

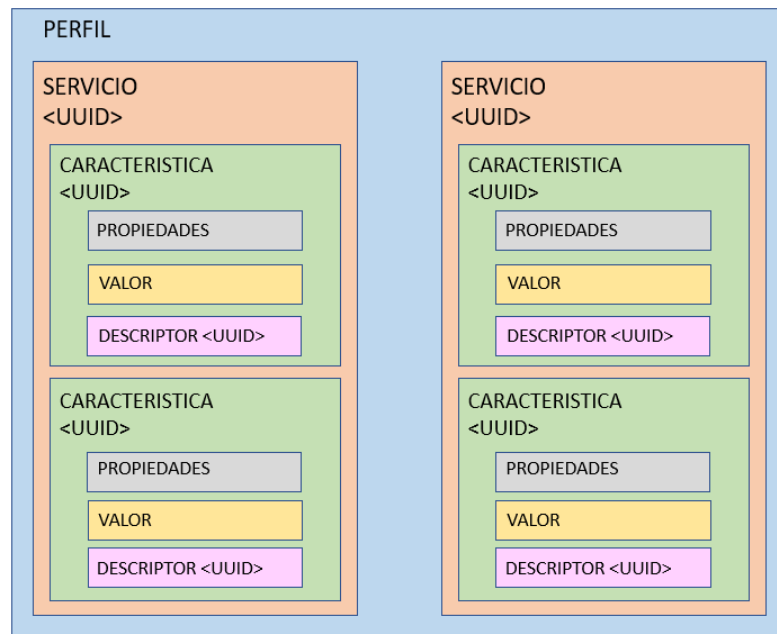


Figura 19. Jerarquía datos BLE

Capítulo 3

Metodología y Desarrollo

3.1 Introducción

Este capítulo describe el método de trabajo utilizado para la consecución de los objetivos planteados. También recoge el proceso de juntar todas las tecnologías mencionadas en Capítulo 2, para conformar el proyecto.

El método de trabajo seguido consta de un prototipo incremental, desarrollado en diferentes fases en las que se incorporaba nuevo hardware y se implementaba sus funciones, realizando pruebas previas, para comprobar que permite alcanzar los objetivos planteados y pruebas unitarias de funcionamiento e integración con el resto de componentes.

3.2 Diseño y construcción del robot

Esta sección describe todas las fases del prototipo. Todas las fases están formadas por tres partes: elección y pruebas del hardware, desarrollo de librerías y validación a través pruebas unitarias.

3.2.1 Primera fase del prototipo

En esta primera fase, el objetivo es la construcción de un robot capaz de desplazarse en todas las direcciones del plano horizontal, permitiendo variar su velocidad. Además debe ser capaz de detectar obstáculos, reaccionando a través del frenado para evitar la colisión.

En primer lugar, para armar el cuerpo del vehículo, se ha empleado el kit *AptoFun 2WD Motor Smart Car Chassis for Arduino- with 2 Gear Motor and Battery Box*. Este kit se puede encontrar en Amazon [19], como podemos ver en *Figura 20* incluye:

- El chasis, una plataforma de metacrilato con ranuras para poder añadirle accesorios.
- Dos motores y sus respectivas ruedas.
- Una rueda universal. Para mantener el dispositivo en equilibrio
- Un zócalo para 4 pilas AA*, sus conectores y un interruptor. No usado, ya que se alimenta con baterías.
- Toda la tornillería necesaria para armarlo.



Figura 20. Kit AptoFun

En segundo lugar, para la detección de obstáculos, se han empleado dos sensores de ultrasonidos, uno para la parte delantera y otro para la parte trasera. El modelo de sensor escogido es el HC-SR04, ver *Figura 21*, ya que es el más conocido y, por tanto, existen muchos tutoriales en internet, además es muy fácil de conseguir a un precio inferior a 3€ la unidad. Este sensor tiene un rango de detección de 2-400cm con una precisión de ± 3 mm, lo que lo hace bastante apto.

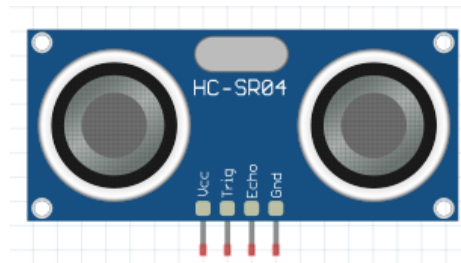


Figura 21. Sensor HC-SR04

En tercer lugar, es necesario hardware para controlar los motores y procesar la información de los sensores. Como controlador de motores, se ha escogido el modelo l298n, ver *Figura 23*, por su bajo coste de apenas 4€ y porque permite control de velocidad por *Señal PWM* y control de sentido de giro, para dos salidas independientes. En el caso de la placa de desarrollo, se ha escogido una placa basada en ESP32, ya que ofrece mayor cantidad de GPIOs, necesarios porque se requiere conectar un gran número de componentes y porque su coste es aproximadamente la mitad que las otras analizadas en [2.6.1]. En concreto, el modelo elegido es el ESP-WROOM-32 de diymore, ver *Figura 22*, aunque existen más fabricantes que desarrollan modelos con especificaciones muy parecidas.

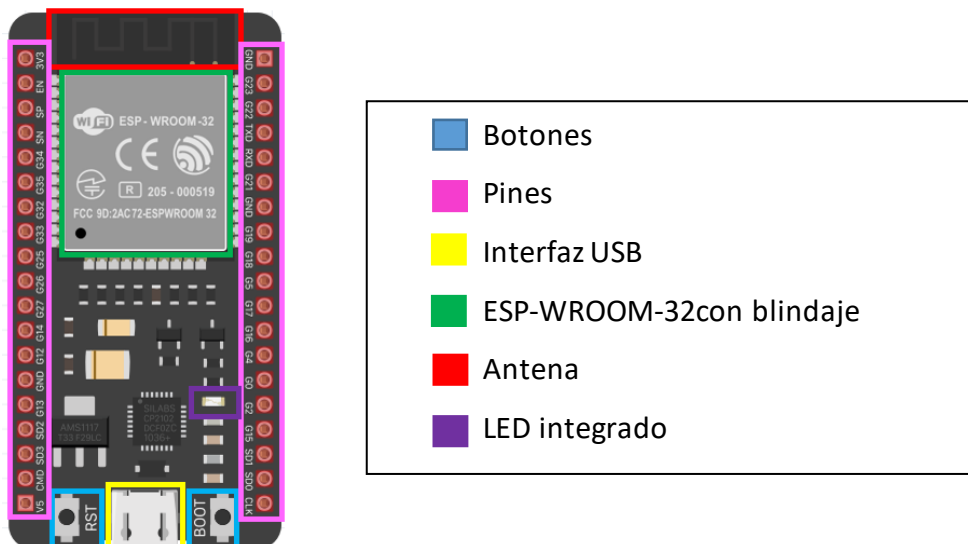


Figura 22. ESP-WROOM-32 DevKit Diymore

Para conectar ambos sensores de ultrasonidos, es importante que ambos estén conectados a la misma alimentación (debe ser de 5v) y también a la misma tierra, esto es porque necesitan la misma referencia para que los resultados sean correctos. El pin

de TRIGGER, debe estar conectado a un GPIO configurado como salida y el pin ECHO a uno configurado como entrada.

Para el controlador de motor, observando en *Figura 23*. La configuración es [20]:

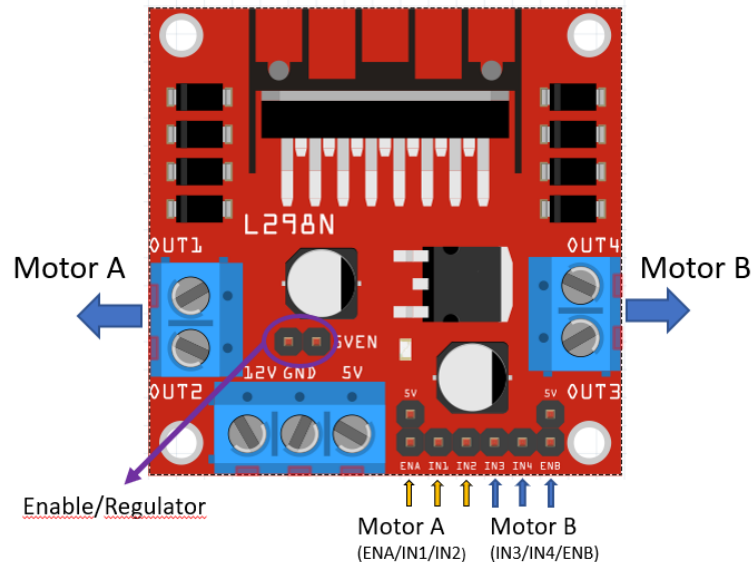


Figura 23. Controlador L298N

- Los pines IN1, IN2, controlan la dirección de giro del motor A. Los respectivos IN3, IN4, son para el motor B. Estos van conectados pines GPIO del microcontrolador, configurados como salida.
- Los pines ENA y ENB son para activar/desactivar la salida, podemos tenerlos conectados siempre mediante un jumper o como es nuestro caso controlarlos mediante una Señal PWM para variar la velocidad de giro. También van conectados a GPIOs de salida.
- En la parte inferior, también tenemos una bornera para la alimentación, cuyo comportamiento depende del “Regulador”:
 - Si el regulador está activado (jumper cerrado), el pin 5v se comporta como una salida de 5v que podemos usar para alimentar otros dispositivos. En este caso se debe alimentar mediante el pin 12v con una tensión de 6-12v.
 - Si el regulador está desactivado (jumper abierto), el pin 5v se comporta como una entrada a la que tendremos que proporcionar un voltaje de entre 4.5-5v.

En nuestro caso el regulador esta activado (jumper cerrado), se alimenta con dos baterías de 3.7v y 750mAh, conectadas en serie para alcanzar los 6v mínimos que requiere el driver. El pin 5v se usa para alimentar al ESP32.

- Por último, a ambos lados tenemos las borneras que suministran la salida a ambos motores.

En cuanto al ESP32, como comentaba, se alimenta con la salida de 5v del driver, también se debe conectar la tierra del driver con la del microcontrolador.

3.2.2 Segunda fase del prototipo

Para esta segunda fase, el objetivo es lograr añadir las funcionalidades de seguir una línea y medir el nivel de luminosidad.

Para medir la luminosidad, el sensor escogido es KY-018, es un fotorresistor de resistencia LDR, véase en *Figura 24*, para este caso de entre 50Ω y 500Ω . El objetivo es obtener una referencia del nivel de luminosidad, por tanto, no es necesario ningún dispositivo demasiado sofisticado, por lo que cumple los requisitos.

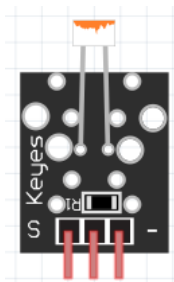


Figura 24. Photoresistor module KY-018

Para seguir las líneas, se necesita un sensor de infrarrojos activo, el sensor escogido es TCRT5000, véase en *Figura 25*. Tiene una distancia de detección de entre 1 y 8mm, ajustable mediante un potenciómetro. Esta distancia es suficiente pues los sensores van colocados en la parte baja del coche muy cerca del suelo.

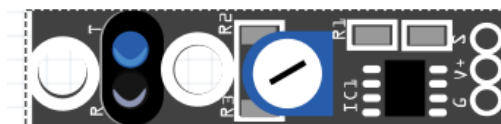


Figura 25. Sensor sigue-líneas TCRT5000

Para poder desarrollar el comportamiento de seguir líneas, se deben emplear dos sensores para detectar hacia qué lado nos hemos desviado y los sensores deben de estar colocados en la parte delantera, para lograr detectar los desvíos con rapidez. En *Figura 26* podemos ver el modo de operación.

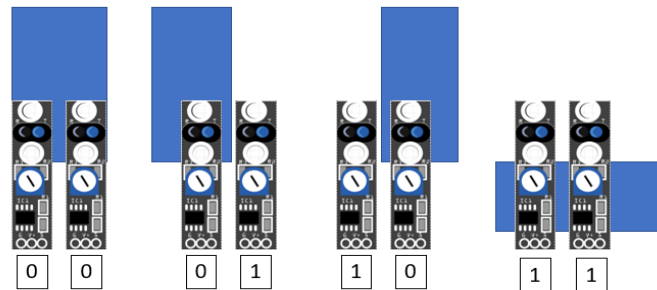


Figura 26. Configuración sensor sigue-líneas

3.2.3 Tercera fase del prototipo

En esta fase, el objetivo es que el robot sea capaz de detectar diferentes marcadores fiduciaros y reaccionar a ellos. Esta es la fase más ambiciosa y con mayor complejidad, pues requiere de añadir una nueva placa con microcontrolador, en este caso con sensor de cámara, que esta placa pueda reconocer diferentes tags y también desarrollar un protocolo de comunicación que permita comunicarse a los dos microcontroladores.

La placa con sensor de cámara escogida ha sido la ESP32-CAM con el sensor OV2640, ya que ofrece mejor resolución. El porqué de esta cámara es porque para un sistema de marcadores fiduciaros no se necesita una cámara tan potente como la de Openmv, y la ESP32-CAM, tiene capacidad de sobra para implementar estas funciones y además a un coste mucho más bajo.

Como medio para el protocolo de comunicación, se ha optado por usar el puerto serie (UART). En el caso del ESP32 del coche, se emplea el UART0, pines Tx y Rx y en el caso de la cámara, el único puerto serie del que dispone, que son los pines GPIO1 => Tx y GPIO3 => Rx. Para la comunicación, se ha optado por un medio físico, como es el caso del puerto serie porque esta solución nos aporta mayor velocidad de transmisión de datos, junto con un menor consumo de batería, aspectos muy importantes para un robot de estas características.

3.2.4 Cuarta fase del prototipo

En esta fase el objetivo es lograr controlar el dispositivo de forma remota, mediante un mando.

En primer lugar, para construir el mando, necesitamos al menos un joystick para hacer de interfaz y un microcontrolador para procesar las señales y para comunicarlas al vehículo, además este microcontrolador debe disponer de algún tipo de conectividad inalámbrica. Por estos motivos, se ha vuelto a emplear la placa ESP-WROOM-32 de diymore.

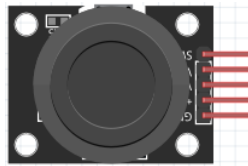


Figura 27. Joystick module

Para conectar el ESP32 y el joystick, se debe tener en cuenta que los pines VRX y VRY del joystick, deben ir conectados a GPIOs analógicos, configurados como entradas. El pin SW, va a un GPIO digital, pero en este caso no se usa, se puede dejar desconectado.

Para alimentar el ESP32 del mando, se puede tener conectado a un ordenador por el cable USB. Para el joystick, se debe conectar al pin de 3.3V del ESP32, porque con 5V no funciona correctamente.

El protocolo empleado para comunicar ambos ESP32 es BLE, por su mayor eficiencia energética.

3.2.5 Diseño final del prototipo

Esta sección incluye el resultado final de la parte física del robot y una representación gráfica de las conexiones y los diferentes componentes usando el software fritzing.

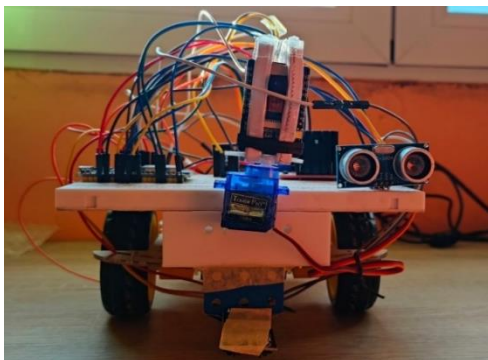


Figura 28. Vista de frente

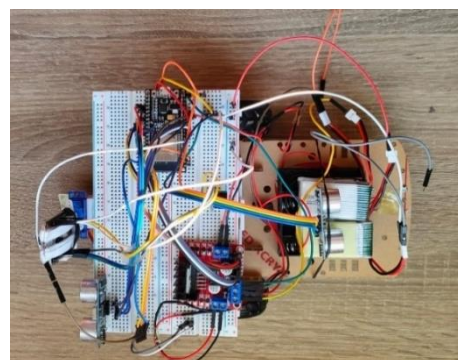


Figura 29. Vista superior

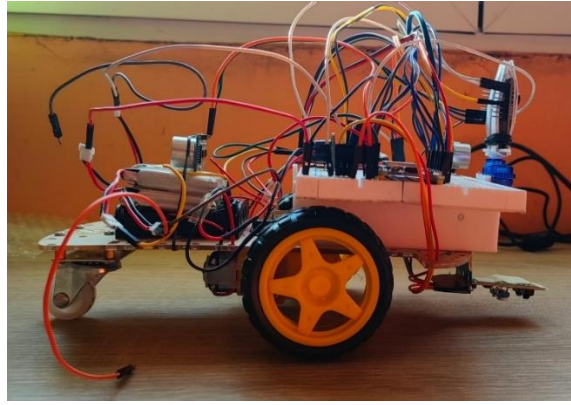


Figura 30. Vista perfil

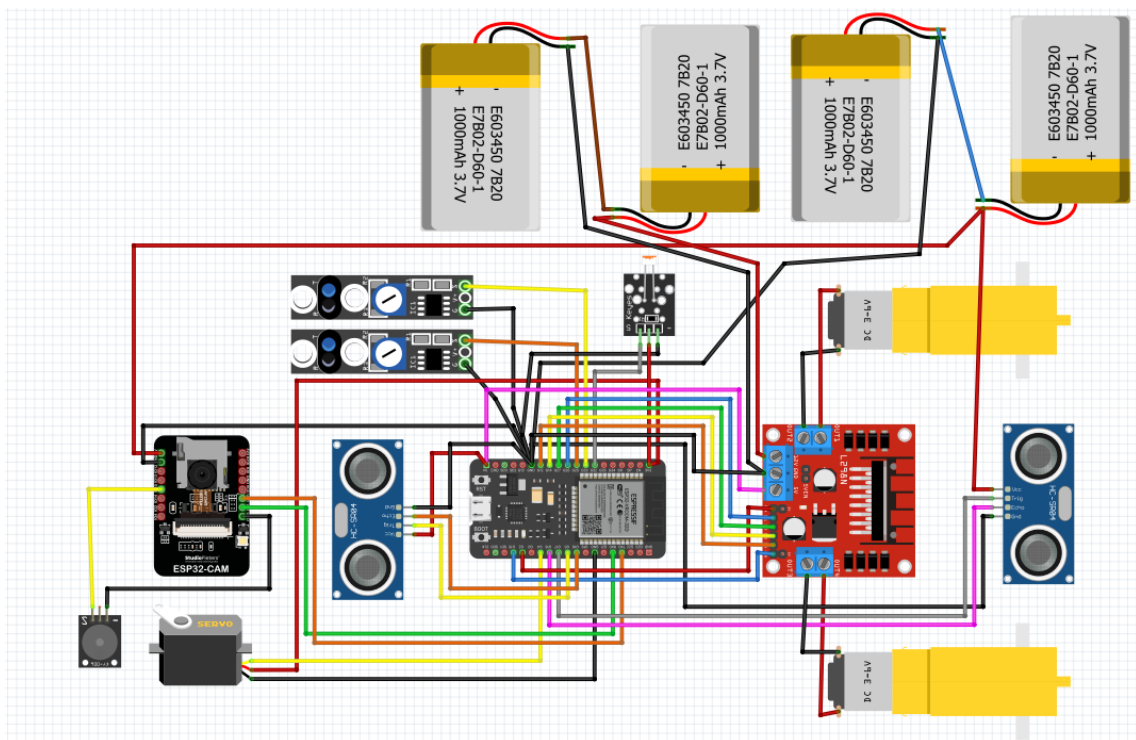


Figura 31. Diseño electrónico del coche realizado con fritzing

3.3 Desarrollo de funcionalidades

Una vez descritas las diferentes fases de construcción del prototipo y presentado el hardware empleado, lo siguiente es explicar la parte de software.

El código está dividido en diferentes librerías según su funcionalidad. El objetivo que se persigue con esto es crear un código más organizado y modular, que permita ser usado en otras aplicaciones, aprovechando únicamente lo que sea necesario. Las diferentes

librerías, están divididas en directorios que contienen el fichero de cabecera (.h), el de código (.cpp) y un ejemplo de uso.

3.3.1 Librerías usadas

Antes de empezar a detallar el proceso de desarrollo del código, es importante mencionar las librerías empleadas durante el proceso. Estas librerías son:

- Librerías de ESP32. Para poder usar las placas con el chip ESP32 en el IDE de Arduino es necesario añadir las URLs de estas placas para poder descargar el núcleo y las librerías necesarias de ESP32 para Arduino. Link: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
- Librerías de AprilTag. Existe la librería oficial creada por AprilRobotics, pero para nuestro caso usaremos una adaptación de la librería para dispositivos de esp-idf (como es el caso de la ESP32-CAM), que proporciona los mismos métodos de detección y decodificación. Esta adaptación ha sido diseñada por Satoshi Tanaka y la podemos descargar desde su repositorio de GitHub: <https://github.com/stnk20/apriltag/tree/esp-idf>

3.3.2 Sensores

Para definir los sensores se han empleado clases, de modo que podemos añadir más sensores de cada tipo simplemente creando un objeto de su clase. De este modo, se logra un código más modular, simplificando y evitando repetir código. Cada clase incluye su método constructor, que lo que hace es configurar los pines y un método para obtener el valor leído por el sensor.

- Sensor de ultrasonidos. El constructor recibe como parámetros los pines de ECHO y de TRIGGER, referenciados en *Figura 32* como “echo” y “trig” respectivamente. En cuanto al método de lectura, para que podamos calcular distancias, el modo de operación es:
 1. Poner a HIGH (5v) el pin TRIGGER durante 10µs, con esto el emisor lanza 8 pulsos de ultrasonidos a 40KHz.
 2. Pasados los 10 µs, volver a poner a LOW.

3. Leer el pin ECHO, con la instrucción `pulseIn(<pin ECHO>, HIGH)`. Con esta función, medimos lo que dura el pulso en HIGH.
4. Calcular la distancia con su fórmula.

Sensor_ultrasonidos	
- echo_pin : integer	
- trigger_pin : integer	
- distancia : float	
- tiempo : long	
+ Sensor_ultrasonidos(in echo: integer, in trig: integer)	C0
+ lectura_ultrasonidos(): integer	

Figura 32. Clase del sensor de ultrasonidos

- Sensor de luminosidad. Su constructor recibe como parámetro el pin de “Signal”, referenciado en *Figura 33* como `lumin_pin` para configurarlo como entrada. En cuanto al método de lectura, consiste en realizar una lectura del valor analógico del pin “Signal”, esto se hace con `analogRead(pin)`. Como los datos se leen de un pin analógico y el ADC tiene una resolución de 12bits, los valores obtenidos son de entre 4095 para total oscuridad, y más bajos cuanto mayor luminosidad.

Sensor_luminosidad	
- luminosidad_pin : integer	
+ Sensor_luminosidad(in lumin_pin: integer)	C0
+ lectura_luminosidad(): integer	

Figura 33. Clase del sensor de luminosidad

- Sensor sigue-líneas. Su constructor recibe como parámetro el pin de “Signal”, referenciado en *Figura 34* como “`lineas_pin`” para configurarlo como entrada. En cuanto al método de lectura, consiste en realizar una lectura del valor digital del pin “Signal”, esto se hace con `digitalRead(pin)`, como es un valor digital, los valores son 0 o 1.

Sensor_sigue_lineas	
- sigue_lineas_pin : integer	
+ Sensor_sigue_lineas(in lineas_pin: integer)	C0
+ lectura_linea(): integer	

Figura 34. Clase sensor sigue-líneas

3.3.3 Acciones

Incluye todas las funciones para configuración y manejo de los motores junto al controlador. Estas librerías están desarrolladas formando una jerarquía de funciones, partiendo de las más básicas, que controlan los aspectos de más bajo nivel del hardware, a modo de driver, ver *Figura 35*. El objetivo con este desarrollo es crear un código más robusto y que permita cambios en el hardware sin necesidad de rescribir toda la librería. Dicho de otro modo, si cambiamos algún dispositivo, que solo sea necesario modificar las funciones de bajo nivel.

3.3.3.1 Desarrollo de controladores

- Driver inicializador de motores. Esta función configura los parámetros necesarios para que los motores funcionen. Realiza tres tareas:
 1. Configura las entradas IN1-4 del I298n como salida.
 2. Configura una señal de PWM para cada motor con las propiedades especificadas, en este caso una frecuencia de 1000, una resolución de 8bits y los canales de PWM 0 y 1.
 3. Adjuntar a un GPIO un canal de PWM de los mencionados antes. Esto lo hace también para ambos motores.
- Driver para control de motores. Se divide en dos funciones que permiten controlar cada motor por separado. Cada una de estas funciones recibe como parámetros el nivel de voltaje de sus respectivos pines y la velocidad de giro. La velocidad es el ciclo de trabajo, que va de 0 a 255. Las tareas que realiza son:
 1. Vuelve a adjuntar el GPIO con el canal PWM. Esto es porque al usar otros dispositivos que emplean PWM, este se desconfigura.
 2. Ajusta la velocidad del rango 0-100 a 0-255, que es el rango del ciclo de trabajo. Y modifica el ciclo de trabajo de la señal PWM.
 3. Pone los pines a los voltajes indicados, haciendo así que la rueda gire en un sentido u otro, o frene en caso de poner ambos a nivel bajo.

3.3.3.2 Desarrollo de capa auxiliar

Para los casos de avanzar o acciones derivadas de avanzar (como puede ser girar a la derecha avanzando hacia delante), debemos de comprobar que no existen obstáculos

enfrente. Por este motivo, se ha implementado esta capa auxiliar encima de los drivers de los motores. Esta capa, la componen 2 funciones, una para cada sentido de giro. Cada función, recibe como parámetros las velocidades de ambas ruedas.

En primer lugar, comprueba que no hay obstáculos a menos de 25cm. Para llevar a cabo esta acción emplea el objeto sensor de ultrasonidos con su método de obtención del valor. En caso afirmativo, frena los motores. En caso negativo, establece el sentido de giro y las velocidades indicadas a cada motor.

3.3.3.3 Desarrollo de capa intermedia

Para las acciones primarias de avanzar, tanto delante como detrás. Se dividen en dos funciones, una para cada acción, que reciben como parámetros: la velocidad y el tiempo. Primero invocan a la correspondiente función de la capa auxiliar. Por último, durante el periodo de tiempo que se está ejecutando la acción, se está comprobando continuamente que no hay obstáculos a menos 25cm.

Para las acciones primarias de girar izquierda o derecha, no es necesario comprobar si existe riesgo, pues gira sobre su eje y nunca chocaría, no es necesario la capa auxiliar, por lo que trabaja directamente con el driver. En este caso también existen dos funciones, una para cada sentido de giro (izquierda/derecha). Los parámetros son los mismos: velocidad y tiempo (en este caso la velocidad es para una rueda solo. Estas funciones lo que hacen es poner a girar la rueda contraria al sentido indicado y esperar sin hacer nada el tiempo indicado, mientras la rueda gira.

Para la acción de frenar, simplemente invoca a los driver estableciendo todos los pines a voltaje bajo y con velocidad 0.

3.3.3.4 Desarrollo de capa de alto nivel

Estas son las funciones de usuario, que se corresponden con las acciones básicas. Estas funciones actúan como manejador, invocando a la correspondiente función de la capa intermedia y luego invocan al driver de frenar motores, si no se hace los motores continúan ejecutando la acción indicada indefinidamente.

- Avanza (<sentido>, <velocidad>, <tiempo>). Para moverse hacia delante o hacia atrás.
- Gira (<sentido>, <velocidad>, <tiempo>). Para girar izquierda o derecha.

En el caso de la función de frena, esta llama directamente a la de la capa intermedia y espera el tiempo indicado.

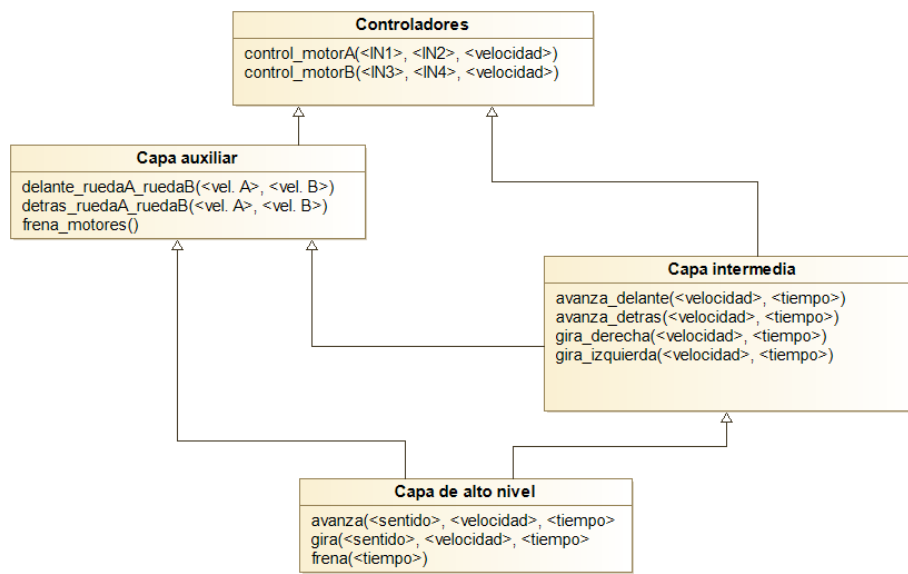


Figura 35. Jerarquía de funciones de Acciones

3.3.4 ESP32-CAM

Para el caso de la cámara, solo existe una única librería, esto es porque el número de funcionalidades es más reducido y, al fin y al cabo, todas forman parte de una funcionalidad que es la de detección de tags y comunicación.

Las diferentes funciones que incorpora la librería de la cámara se pueden clasificar en diferentes apartados:

- Inicialización de parámetros. Estas tareas son realizadas por la función `inicializa_camara()`. Esta función es la que configura los pines del sensor de cámara y el formato de imagen. Para el formato, la resolución se ha establecido en QQVGA (160x120), se opta por una resolución pequeña para ganar en velocidad de detección, con resoluciones más grandes, mejoraba la distancia de detección, pero el tiempo de detección también aumentaba; el formato de píxeles es en escala de grises, porque es con el formato que trabaja el algoritmo

de detección. En cuanto a la configuración de pines, esta se obtiene de la librería “camera_pins.h”, de manera que solo tenemos que indicar que modelo de cámara estamos usando, para este caso: `#define CAMERA_MODEL_AI_THINKER`

- Detección y decodificación. Se encarga la función `detecta_tag(<*n_tags>)`, recibe un puntero a una variable que almacena el número de tags detectados, y nos devuelve un array con la información de cada tag. El modo de operación es:
 1. Captura un frame.
 2. Aplica el método de detección de TAGs de la librería de AprilTag y se obtiene un array que contiene estructuras para cada tag, incluyendo información relativa a este como es su identificador, distancia de hamming...
 3. Para cada tag detectado se estima su pose con el método para estimar pose proporcionado por la librería de AprilTag.
 4. Finalmente, la función nos devuelve un array creado dinámicamente en función del número de tags detectados. Este array incluye una estructura para cada tag, que incluye su identificador y su posición en el espacio 3D.
- Funciones auxiliares. Funciones que complementan el funcionamiento de las demás. Encontramos tres:
 - `ordena_tags(<*array>, <longitud>, <sortby>, <orden>)`. Esta función permite ordenar el array de detecciones. Recibe un puntero al array obtenido por el método `detecta_tag(...)`, el número de tags, si se ordena por ID o por distancia y si el orden es ascendente o descendente.
 - `busca_id(<*array>, <id>, <*distancia>, <*pose_x>)`. Esta función nos indica si el tag que buscamos ha sido detectado. Recibe un puntero al array de detecciones del método `detecta_tag(...)`, el identificador que buscamos y dos punteros a variables que almacenan la distancia y la pose en X.
 - `buzzer(<nota>, <duración>)`. Esta función permite hacer sonar un buzzer. Usando una nota, consultar librería `pitches.h`, durante un tiempo indicado.
- Comunicación con coche. Detallado en [3.3.5].

3.3.5 Comunicación serial ESP32-ESP32CAM

Durante el diseño se tomó la decisión de que la cámara se encargase de la parte de inteligencia y el coche de la parte de control de actuadores, por tanto, se ha optó por un modelo maestro-esclavo, donde el ESP32-CAM es el maestro, inicia la comunicación y envía las órdenes; mientras que el ESP32 es el esclavo el cual se limita a responder a las órdenes del maestro y transmitir las al resto de componentes del vehículo

En cuanto al protocolo plantado, este dispone de un mecanismo de establecimiento de conexión, de control de integridad mediante checksum y con control de timeout.

La estructura de la librería de ambos dispositivos es similar, consiste en una función principal que actúa como handler, la cual invoca a las correspondientes funciones auxiliares.

El método para calcular el checksum, es el conocido como checksum XOR [21], ampliamente usado por su sencillez y rapidez de cálculo. Consiste en procesar una secuencia de datos en fragmentos de 8 bits y ejecutando la operación XOR en todos los fragmentos. El valor se devuelve negado, así podemos diferenciar un fallo completo (todo sería cero), frente a un envío de datos formado por 0's.

En *Figura 36*, podemos ver como se desarrolla el protocolo de comunicación.

1. Primero la cámara envía el carácter 'SYN' al coche, si este responde con 'ACK', la conexión está establecida.
2. Una vez establecida la conexión, cuando la cámara va a enviar una orden, primero escribe el carácter 'STX', el coche al recibir este carácter interpreta que va a recibir una orden.
3. Tras obtener los datos y comprobar la integridad con el checksum y el carácter 'ETX'. El coche responde con un 'ACK' y ejecuta la acción. Si no fueran correctos no responde ni ejecuta la acción.
4. Si la cámara recibe el 'ACK' a tiempo todo correcto, si no lo recibe la conexión se corta y se reinicia.
5. Cuando el coche termina de ejecutar la acción envía un 'EOA' para indicar que ha terminado.

En cualquier caso de error, el procedimiento es cortar la comunicación y que esta se reinicie de nuevo. Para estos casos está la función de `termina_comunicacion(<timeout>)`, que escribe el carácter 'EOT', para que el coche lo interprete como cierre abrupto de la comunicación.

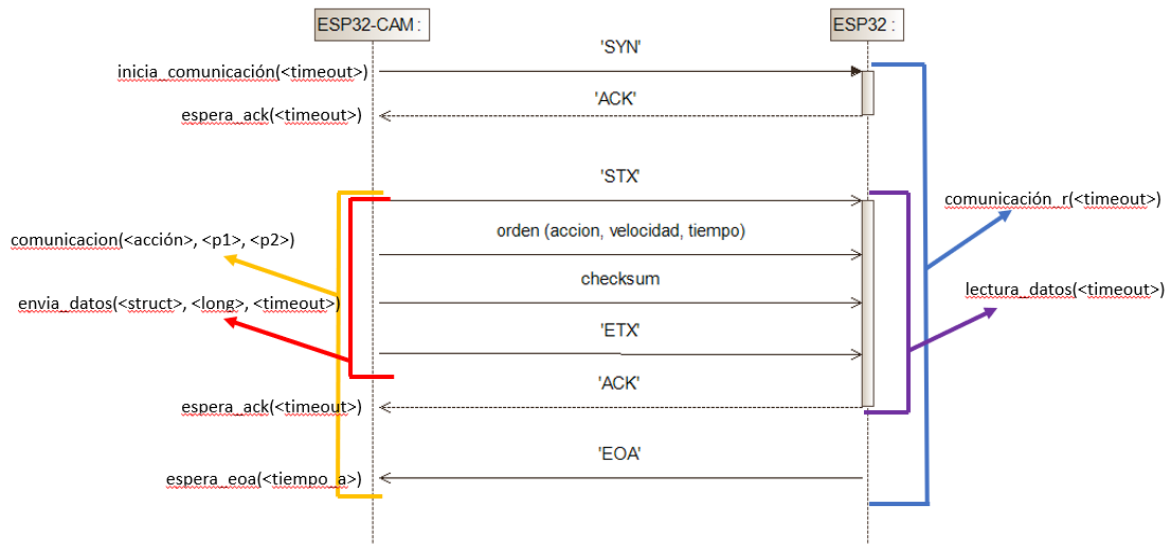


Figura 36. Protocolo de comunicación serial

3.3.5.1 Parte de la cámara.

- Método `inicia_comunicación(<timeout>)`. Este método escribe por el puerto serie el carácter 'SYN' y espera la recepción del 'ACK' por parte del cliente, con el método `espera_ack(<timeout>)`. Si todo es correcto, la conexión está establecida y se activa el flag *conectado*.
- Método `termina_comunicación(<timeout>)`. Lo contrario que el anterior, este escribe 'EOT' y desactiva el flag *conectado*.
- Método `espera_ack(<timeout>)`. Este método comprueba el puerto serie, indica si el mensaje recibido es 'ACK' o si ha habido un error o expiración del timeout.
- Método `espera_EOA(<tiempo_accion>)`. Este método es como el de `espera_ack`, pero en este caso espera la recepción del carácter de fin de acción 'EOA'. Para este caso no se usa timeout, se emplea el tiempo que va a durar la acción ordenada.
- Método `envia_datos(<estructura_accion>, <longitud>, <timeout>)`. Este método, en primer lugar, escribe el 'STX', para indicar que va a transmitir una

orden. Luego transmite la orden <estructura_accion>. Computa el checksum y lo envía. Por último, escribe el 'ETX', para indicar que termina el mensaje.

Llama a espera_ack().

- Método comunicación(<acción>, <parametro1>, <parametro2>). Este método hace de handler, primero comprueba que la conexión está establecida, con el flag conectado, en caso negativo invoca a inicia_comunicacion(). Si está establecida, llama al método envia_datos() con la estructura ya creada. Ejecuta el espera_EOA() y si da error termina.

3.3.5.2 Parte del coche

- Método comunicación_r(<timeout>). Este método es el handler. Lo que hace es revisar el puerto serie. Cuando llega un mensaje, lee su cabecera y en función de su valor ejecuta la acción correspondiente.
- Método lectura_datos(<timeout>). Este método es llamado desde el handler, cuando la cabecera recibida es 'STX', que indica que se va a recibir una orden. Este método, decodifica la orden recibida y llama al método ejecuta_accion() y cuando termina escribe el carácter 'EOA'.
- Método ejecuta_accion(). Este método invoca a la correspondiente función de movimiento de [3.3.3.4], según la acción decodificada.

3.3.6 Mando

Para el controlador del joystick, primero se definen sus pines, en este caso 35 para el pin X y 34 para el pin Y (recordar que deben ser analógicos). En cuanto a la función del controlador, se ha dividido cada eje en 5 áreas, esto es porque el rango de valores es muy grande (0-4096), ver *Figura 37*, y manejar el coche con tal grado de presión supondría una gran sobrecarga innecesaria, además de que el valor normalmente tiene una ligera oscilación.

En este caso el valor máximo también viene determinado por la resolución del ADC ($2^{\text{resolución}} = 4096$), puesto que los pines son analógicos.

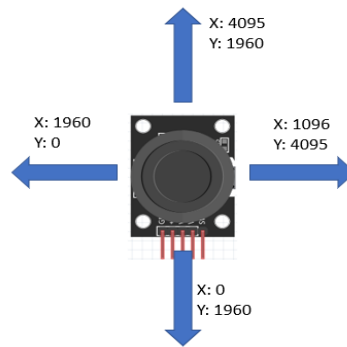


Figura 37. Rangos teóricos del joystick

Las divisiones están en 400-1700-2100-3800, siendo el estado de reposo 1700-2100. En los extremos la velocidad es la máxima (en una dirección u otra) y en los rangos centrales es media. El reposo se determina con un valor de 1900 para X e Y.

A cada división se le asigna un número de 1 a 5, primero las de la X y luego se concatena las de la Y. De este modo el mapa de dirección nos quedaría:

51	52	53	54	55
41	42	43	44	45
31	32	33	34	35
21	22	23	24	25
11	12	13	14	15

Tabla 2. Mapa del joystick

Finalmente devuelve la concatenación de ambas posiciones como un String, esto es por compatibilidad con el protocolo BLE.

3.3.7 BLE

Para la comunicación de los dos ESP32 que componen el mando y el coche, se emplea el protocolo BLE. En cuanto a la definición de los roles, el mando es el servidor, ya que es quién toma los valores de posición del joystick y los envía al coche.

Dado que solo tenemos un sensor (joystick), solo se ha definido un servicio, con una única característica que recoge las posiciones en X e Y. El contenido de esta característica se obtiene del controlador del joystick [3.3.6].

La característica se define con la propiedad NOTIFY, de este modo, el cliente solo tiene que suscribirse a esta y automáticamente será notificado cuando cambie. Cuando la característica cambie de valor, el dispositivo cliente, en este caso el coche, decodifica el valor notificado e invoca a las funciones de la capa auxiliar [3.3.3.2] con los valores de velocidad adecuados. En caso de giro o de frenar, se invoca a las funciones de la capa intermedia [3.3.3.3].

3.4 Conclusiones

Tener las diferentes funciones separadas en clases y el diseño modular de estas, me ha permitido ir añadiendo las modificaciones de las fases posteriores con mayor facilidad. Por tanto, para trabajos futuros, cuando se requiera añadir nuevas funciones el proceso de integración, resultará más sencillo.

Capítulo 4

Experimentos y Resultados

4.1 Introducción

Este capítulo recoge las diferentes pruebas que se han realizado. Tanto para determinar ciertos parámetros de configuración como para validar que la solución planteada cumple los objetivos planeados.

4.2 Experimentos de calibrado

Esta sección incluye experimentos llevados a cabo para determinar el valor de parámetros de configuración del sistema.

4.2.1 Obtener la distancia focal

El método de estimar la pose proporcionado por la librería de AprilTag, necesita configurar una serie de parámetros que le permitan hacer los cálculos. Estos parámetros son la distancia focal en X e Y, el centro focal en X e Y, y el tamaño en metros de la etiqueta impresa.

Ante la ausencia de programas específicos para el ESP32-CAM, para determinar la distancia focal, se empleó el propio método de estimar la pose proporcionado por la librería de AprilTag, realizando modificaciones en los parámetros hasta que la distancia estimada fuera igual que la real.

El centro focal es el centro de la imagen, luego sus coordenadas vendrán determinadas por el centro de la imagen. En nuestro caso usamos una resolución QQVGA (160x120). Luego el centro de dicha imagen será el $X=80$, $Y=60$. Las etiquetas impresas usadas tienen un tamaño de 0.158m.

Ahora para obtener ambas distancias focales, partimos de la premisa de que es muy probable que estas sean la misma. Por tanto, fijamos los valores del centro focal y vamos variando ambas distancias focales.

Para realizar las pruebas, se hizo un código sencillo que usaba el método de estimar la pose de la librería de AprilTag y mostraba por pantalla las tres coordenadas.

Para una distancia de 0.5m en el eje Z, se tomaron 5 resultados y se calculó su media, ver Tabla 3.

Distancia focal (píxeles)	Valor medio (metros)
150	0.567
130	0.493
135	0.52
131	0.504

Tabla 3. Resultados de distancia focal

A la vista de los resultados, el valor que mejor se ajusta es 131 píxeles. Tras configurar con este valor, se comprobó en distancias de 0.5, 1.5 y 2m, ver ***¡Error! No se encuentra el origen de la referencia.*** Obteniendo una precisión similar.



Figura 38. Cálculo de parámetros intrínsecos

4.2.2 Determinar umbral de detección de obstáculos

Para determinar el umbral de frenado por riesgo de colisión, se realizaron pruebas ajustando la distancia desde los 15cm en adelante. Con 15 chocaba frecuentemente, con 20 nunca llegaba a chocar, pero cuando iba a velocidad máxima (255 ciclo de trabajo), se quedaba muy cerca, por lo que por seguridad se ha añadido 5cm más de margen.

4.2.3 Comprobar funcionamiento del fotorresistor

Se conecta al ordenador y se muestran por pantalla los valores obtenidos. Primero se tapa el sensor para dar la máxima oscuridad y luego se alumbra con una linterna para obtener la mayor luminosidad posible. Los valores para la máxima oscuridad fueron de 4095 y luego los valores cuando se le apunta con una linterna llegaron hasta 220.

4.2.4 Determinar umbrales del joystick

Para determinar que umbrales definir para identificar la posición del joystick. Se conectó el joystick al ordenador y se leyeron los datos que ofrecía. Para los valores extremos el resultado fue el esperado [0-4095]. En cuanto a la posición en reposo, los datos estaban alrededor de 1900.

4.2.5 Mejorar calidad imagen

Con el fin de mejorar la calidad de imagen para mejorar la detección de los tags. Se configuró la cámara como servidor para transmitir video en directo, se giró la ruleta de enfoque hasta obtener una imagen nítida.

4.3 Desarrollo de casos de uso para validación

A continuación, se presentan tres casos de uso, el objetivo es determinar si el prototipo construido es usable y que cumple con los objetivos establecidos para este proyecto.

4.3.1 Manejo mediante el mando BLE

En este experimento, se conectan el mando y el coche, mediante el protocolo BLE, haciendo uso de las librerías proporcionadas. El procedimiento consiste en:

1. Conectar ambos dispositivos
2. Mediante la posición del joystick, ejecutar diferentes acciones: avanzar delante, detrás, girar... y variar su velocidad.
3. Avanzar hacia un obstáculo y ver qué pasa.

Con este caso de uso lo que se pretende validar es:

- Que ambos dispositivos son capaces de conectarse y mantener una conexión estable en el tiempo. Se establece la conexión y esta no sufre cortes ni reinicios.
- Que el protocolo de comunicación BLE funciona correctamente. El servidor, en este caso el mando, envía órdenes al coche que este ejecuta:
 - Que el controlador del joystick funciona correctamente, siendo capaz de diferenciar las diferentes posiciones, que se traducen en las diferentes acciones y velocidades.
 - Que el coche ejecuta correctamente las acciones del joystick.
- Que las funcionalidades de acciones del coche funcionan:
 - El coche se mueve y por tanto los controladores y funciones son correctos
 - Si detecta un obstáculo frena y por tanto el sensor de ultrasonidos está bien configurado.

4.3.2 Búsqueda de ArpilTAGs según orden dado

Para este experimento se han impreso 3 AprilTag de la familia 25h9, en concreto han sido los de los identificadores: 0, 2, 3. Estos tags se han dispuesto en una habitación, como se puede ver en *Figura 39*. El procedimiento es:

1. El robot estará ejecutando movimientos aleatorios
2. Cuando detecte el tag que toca en ese momento (primero el 0, luego el 2 y por último el 3), este emite un pitido y se acerca al tag.
3. Tras acercarse, vuelve hacia atrás, gira sobre sí mismo y vuelve al paso 1, buscando ahora el siguiente tag.

Con este caso de uso, se pretende validar es:

- Que el robot es capaz de identificar los diferentes tags. Esto se comprueba porque solo se acerca al tag cuando estes es el que busca.
- Que el buzzer funciona. Emite un pitido cuando se detecta el tag.
- Que el protocolo de comunicación cámara-coche funciona. Tanto las órdenes de movimientos aleatorios, como las órdenes de acercarse al tag, retroceder y girar, son emitidas desde la cámara y ejecutadas por el coche.
 - También verifica que el coche es capaz de decodificar las órdenes que envía la cámara.
- Que el protocolo de decodificación de tags permite conocer con precisión la distancia del tag. Esto se demuestra porque el coche avanza a la etiqueta en función de la distancia en Z.



Figura 39. Caso de uso de encontrar TAGs

4.3.3 Seguimiento de una línea

Para este experimento se ha creado un circuito circular consistente en una línea negra de aproximadamente 3.5cm, ver *Figura 40*. El procedimiento es que el robot siga la línea, corrigiendo su trayectoria cuando se desvíe.

Con este caso de uso se pretende validar:

- Que el sensor sigue-líneas es capaz de identificar la línea negra y detectar cuando se ha desviado.
- Que mediante la API ofrecida se puede crear un comportamiento que permita detectar hacia que lado nos hemos desviado y actuar para corregir la trayectoria.



Figura 40. Circuito seguir línea

4.4 Conclusiones

Los experimentos 4.3.1 y 4.3.2 han resultado satisfactorios, cumpliendo en ambos casos con los objetivos propuestos, por tanto, queda demostrado que es un prototipo completamente funcional y por tanto es apto para terminar de desarrollarlo y emplearlo como una herramienta de aprendizaje.

También con el segundo experimento demuestra que con la API ofrecida se pueden construir comportamientos complejos.

Sin embargo, presenta algunas limitaciones como, por ejemplo:

- La máxima distancia de detección es de aproximadamente 2 metros.
- Según el ángulo, el sensor de ultrasonidos puede fallar y llegar a colisionar.

Por último, he de mencionar que, aunque el caso de uso de seguir una línea no ha podido llevarse a cabo por cuestiones técnicas, pues uno de los sensores estaba defectuoso, estoy convencido de que, sustituyendo el sensor defectuoso, usando la funcionalidad de detectar líneas junto con la librería de movimientos podremos construir un robot sigue-líneas.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1 Conclusiones

Con el presente trabajo, se ha diseñado un robot utilizando microcontroladores y sensores de bajo coste. Se ha realizado su interconexión y desarrollado el software necesario para controlar el movimiento del robot y obtener los valores de los distintos sensores. También se ha desarrollado una API de bajo nivel para la programación de los movimientos del robot, y la detección de tags

Es posible crear un robot usando dispositivos de bajo coste. Esto implica que el coste no es un factor limitante a la hora de promover la robótica. Ver en *I.1* un coste aproximado.

La programación y el control de robots con sensores de bajo coste son accesibles para personas con conocimientos básicos en programación. Esto significa que se pueden utilizar en entornos educativos y se pueden promover habilidades técnicas y de programación.

Los sensores de bajo coste tienen una precisión limitada comparados con sensores de alta calidad, pero todavía son útiles para aplicaciones simples. Esto significa que se deben tener en cuenta sus limitaciones y que se deben elegir los sensores adecuados para cada aplicación.


5.2 Trabajo futuro

Este Proyecto se da por completado, sin embargo, durante el desarrollo de este he ido detectando posibles cambios que se podrían aplicar en posteriores versiones, y que por falta de tiempo o limitaciones técnicas derivadas de las capacidades del hardware o falta de colaboradores no se han podido llevar a cabo. Estas posibles modificaciones, si se implementan correctamente podrían aumentar considerablemente las capacidades de este robot, mejorando su desempeño e incluso añadiendo nuevas funcionalidades. Estas posibles mejoras son:

- Como el ESP32-CAM cuenta con conexión WiFi, esta se puede emplear para derivar el procesamiento de la imagen a un computador u otro microcontrolador y liberar de esta carga al ESP32-CAM, así de este modo su única función sería la de capturar las imágenes y enviarlas. Esto también nos permitiría procesar las imágenes en un dispositivo más potente como es el caso de un ordenador con tarjeta gráfica dedicada y añadir funciones como por ejemplo detección de objetos.
- Modificar los parámetros de la cámara para poder obtener una mayor distancia de detección de los tags. La opción de sustituir el sensor queda descartada porque el módulo ESP32-CAM solo es compatible con el OV2640 (usado en el proyecto) y el OV7670, el cual ofrece menos resolución, por tanto, no mejoraríamos. Siguiendo con este objetivo la mejor solución pasaría por sustituir el módulo por otro más potente, aunque esto encarecería los costes.
- Se puede usar la conectividad Bluetooth del ESP32, de la parte del coche para conectar con nuestros smartphones para poder controlarlo.
- Usar motores con encoder para un mejor control de la velocidad y desplazamiento del robot.
- Diseñar un circuito de gestión de carga para las baterías, que permita la carga de las misma a través de un conector USB.
- Diseño de un circuito impreso para la integración en el mismo de todos los componentes.
- Diseño en 3D de un chasis para el robot integre todos los componentes.

En cualquier caso, este proyecto se ha desarrollado en diferentes librerías separadas según la naturaleza de cada componente y siempre buscando ser lo más modular y sencillo de entender posible, con el objetivo de que este proyecto sirva como base para el desarrollo de nuevos proyectos de la misma naturaleza.

Bibliografía

- [1] M. Peña *et al.*, «Presidente Mauricio Macri Jefe de Gabinete de Ministros».
- [2] «“La robótica educativa ayuda a los alumnos a razonar; eso vale para Informática y para Filosofía”». https://www.eldiario.es/navarra/ultimas-noticias/robotica-educativa-alumnos-informatica-filosofia_1_4697390.html (accedido 6 de febrero de 2023).
- [3] «Espectro Visible - Qué es, longitud de onda y colores». <https://concepto.de/espectro-visible/> (accedido 11 de febrero de 2023).
- [4] «Light Sensor including Photocell and LDR Sensor». https://www.electronicstutorials.ws/io/io_4.html (accedido 11 de febrero de 2023).
- [5] «Fototransistor – TallerElectronica.com / Blog». <https://tallerelectronica.com/fototransistor/> (accedido 6 de febrero de 2023).
- [6] «Detector Infrarrojo  ¿Qué es? Funcionamiento y Aplicaciones». <https://www.edsrobotics.com/blog/detector-infrarrojo-que-es-funcionamiento-aplicaciones/> (accedido 11 de febrero de 2023).
- [7] «Sonido y audición - Apple (ES)». <https://www.apple.com/es/sound/> (accedido 6 de febrero de 2023).
- [8] «In-Depth: How 2-Axis Joystick Works? Interface with Arduino & Processing». <https://lastminuteengineers.com/joystick-interfacing-arduino-processing/> (accedido 6 de febrero de 2023).
- [9] «Cómo funciona un motor eléctrico: tipos y partes que lo componen». <https://www.cursosaula21.com/como-funciona-un-motor-electrico/> (accedido 11 de febrero de 2023).

- [10] «Software de Arduino | Arduino.cl - Compra tu Arduino en Línea». <https://arduino.cl/programacion/> (accedido 6 de febrero de 2023).
- [11] «¿Qué es LEGO Mindstorms? - The Green Monkey Sarriá». <https://www.thegreenmonkey.es/sarria/que-es-lego-mindstorms/> (accedido 6 de febrero de 2023).
- [12] «Realidad Aumentada: ARToolkit para animación de personajes». http://www.disca.upv.es/magustim/val/pfcs_anteriores/arToolkit/ARToolkit.html (accedido 6 de febrero de 2023).
- [13] J. Wang y E. Olson, «AprilTag 2: Efficient and robust fiducial detection», Accedido: 6 de febrero de 2023. [En línea]. Disponible en: <http://april.eecs.umich>.
- [14] D. B. dos Santos Cesar, C. Gaudig, M. Fritsche, M. A. dos Reis, y F. Kirchner, «An evaluation of artificial fiducial markers in underwater environments», *MTS/IEEE OCEANS 2015 - Genova: Discovering Sustainable Ocean Energy for a New World*, sep. 2015, doi: 10.1109/OCEANS-GENOVA.2015.7271491.
- [15] «Arduino® Nano RP2040 Connect».
- [16] «Arduino® Nano 33 BLE Target areas: Maker, enhancements, basic IoT application scenarios».
- [17] «ESP32 Series Datasheet Including», 2023, Accedido: 6 de febrero de 2023. [En línea]. Disponible en: <https://www.espressif.com/en/support/download/documents>.
- [18] «ble-documentation - Dispositivos y Redes Inalámbricos (DRI - 2021/22) - Perusall». <https://app.perusall.com/courses/dispositivos-y-redes-inalambricos-dri-2021-22/ble-documentation?assignmentId=2BfZxiJQoKGaeTqs7&part=1> (accedido 6 de febrero de 2023).
- [19] «AptoFun 2WD Motor Smart Car Chassis for Arduino- with 2 Gear Motor and Battery Box : Amazon.es: Juguetes y juegos». <https://www.amazon.es/AptoFun-Motor-Chassis-Arduino-Battery/dp/B01LW6A2YU> (accedido 6 de febrero de 2023).
- [20] «Controlar motores de corriente continua con Arduino y L298N». <https://www.luisllamas.es/arduino-motor-corriente-continua-l298n/> (accedido 11 de febrero de 2023).

- [21] «Comprobar la integridad de datos en Arduino con checksum». <https://www.luisllamas.es/arduino-checksum/> (accedido 11 de febrero de 2023).
- [22] «Glosario: Espectro electromagnético». https://ec.europa.eu/health/scientific_committees/opinions_layman/es/lamparas-bajo-consumo/glosario/def/espectro-electromagnetico.htm (accedido 11 de febrero de 2023).
- [23] «¿Qué es un fotón? Usos en la energía solar | Svea Solar». <https://sveasolar.es/es-es/que-es-un-foton> (accedido 11 de febrero de 2023).
- [24] «The P-N Junction». <http://hyperphysics.phy-astr.gsu.edu/hbasees/Solids/pnjun.html> (accedido 11 de febrero de 2023).

Anexo

I.1 Coste total

Una aproximación del coste total del proyecto

Dispositivo	Coste
x2 ESP-WROOM-32	22€ (las dos unidades)
ESP32-CAM + sensor OV2640	12€
Sensor KY018	2€
x2 HC-SR04	6€ (las dos unidades)
Joystick	3€
Kit aptofun (chasis, motores y ruedas)	17€
L298N	4€
Passive buzzer	2.50€
x2 TCRT5000	2€ (dos unidades)
x2 protoboard	<10€ (dos unidades)
Set de cables	<10€
X4 baterías (3.7v y 750mah)	<10€
Adaptador UART to USB	2€
PRECIO TOTAL	102.5€

Tabla 4. Coste total aproximado

Estos precios han sido sacados principalmente de Amazon y son una aproximación, en la mayoría de casos al precio superior. Es importante mencionar que se pueden encontrar más baratos en tiendas como AliExpress o incluso comprando los dispositivos en kits.

I.2 Glosario de definiciones

I.2.1. Espectro electromagnético

El espectro electromagnético es el conjunto de longitudes de onda de todas las radiaciones electromagnéticas[22].

I.2.2. Fotón

Un fotón es una partícula elemental portadora de la radiación electromagnética. Los fotones se caracterizan por su energía, la cual depende la frecuencia (en el caso de la luz visible, un fotón azul tiene más energía que uno rojo) [23].

I.2.3. Banda ISM

ISM (Industrial, Scientific and Medical) son bandas de radiofrecuencia electromagnética reservadas internacionalmente para uso no comercial en las áreas de la industria, la ciencia y la medicina.

La ventaja de estas bandas es que están abiertas a todo el mundo sin necesidad de licencia, pero respetando las regulaciones que limitan los niveles de potencia transmitida.

I.2.4. Mecanismo Gymbal

Un gimbal es un soporte giratorio que permite la rotación de un objeto alrededor de un eje.

I.2.5. Señal PWM

PWM son las siglas de Pulse Width Modulation (Modulación por Ancho de Pulso, *gyçen* español). Para transmitir cualquier señal, ya sea analógica o digital, se debe modular para que se pueda transmitir sin perder señal o sufrir distorsión por las interferencias.

PWM es una técnica que permite “emular” señales analógicas mediante una señal digital. Esta técnica permite modificar el ciclo de trabajo (duty cycle) de una señal, mientras se mantiene constante el periodo. El ciclo de trabajo de una señal periódica es

el tiempo que la señal se mantiene en estado alto en relación al periodo y se expresa en porcentaje.

$$Duty\ cycle\ (\%) = \frac{tiemp_{HIGH}}{Periodo} * 100$$

Básicamente, consiste en generar pulsos positivos que se repiten de manera constante activando una salida digital durante un tiempo y apagándola durante el resto. El promedio de esta tensión de salida, a lo largo del tiempo, será igual al valor analógico deseado. El voltaje promedio se calcula con:

$$V_{medio} = (V_{HIGH} - V_{LOW}) * \frac{Duty\ cycle}{100}$$

Normalmente V_{LOW} es 0v y V_{HIGH} será 5v o 3.3v.

I.2.6. Puente – H

A modo de resumen un puente-H podemos decir que es un circuito electrónico que permite activar los motores eléctricos, hacer que giren hacia un sentido u otro y al mismo tiempo permite controlar variables como, velocidad y torque de estos.

La construcción de un puente-H consiste en 4 transistores conectados entre VCC y GND, actuando sobre estos 4 transistores podemos variar el sentido de la corriente que lo atraviesa. El termino de “Puente-H” viene de la representación gráfica del *circuito Figura 41*.

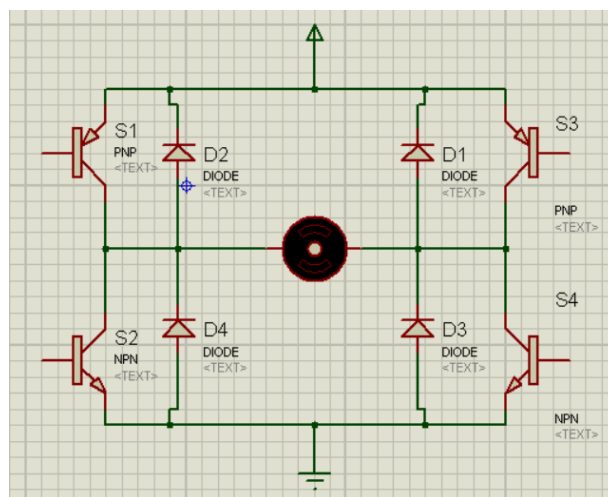


Figura 41. Puente-H

I.2.7. Unión PN

Existen dos tipos de semiconductores extrínsecos, los P y los N. Esta unión hace que la corriente pueda fluir en una dirección, pero al contrario no, creando así un diodo básico[24].

Un semiconductor tipo P, es un material semiconductor al que se le añaden impurezas que crean deficiencias de electrones de valencia, llamados “huecos”.

Un semiconductor tipo N, es un material semiconductor al que se le añaden impurezas que crean “sobras” de electrones de valencias.

I.3 Imprimir AprilTags

Junto con esta memoria se adjunta una carpeta llamada “TAGs para imprimir”, que incluye todos los tags organizados por familias. Para imprimirlos debemos de escalarlos, en mi caso lo hacía con GIMP: Imagen -> escalar imagen. Luego en la ventana de escalado ponía un tamaño de 2000x2000 píxeles y en interpolación es importante dejar en: ninguna.

I.4 Programación de la ESP32-CAM

En esta sección se describen los requisitos previos y los pasos necesarios para poder cargar nuestros programas en la ESP32.

I.4.1. Requisitos previos

- Tener actualizada la librería de ESP32.
- Disponer de un adaptador UART a USB o un programador específico, esto es debido a que este módulo no dispone de puerto USB, debemos de programarlo usando el interfaz serie (pines Rx yTx).
- Si usamos una distribución de Linux, tendremos que instalar, si no lo está, el módulo de comunicación serial de Python. Es tan sencillo como usar este comando: *pip3 install pyserial*.

I.4.2. Posibles complicaciones

- Si usamos Windows 11 es posible tener problemas con los drivers del adaptador serie. Si accedemos al administrador de dispositivos, aparece cualquiera de estos dos mensajes:

PL2303TA DO NOT SUPPORT WINDOWS 11 OR LATER PLEASE CONTACT YOUR SUPPLIER

THIS IS NOT PROLIFIC PL2303. PLEASE CONTACT YOUR SUPPLIER

La mejor solución en mi caso fue usar una máquina virtual de Linux.

- Si necesitamos activar el PSRAM, tenemos que cambiar la placa a “ESP32 Dev Module”. Vamos a Tools->PSRAM->Enable. Volvemos a cambiar a “Ai Thinker”.

I.4.3. Cargar programa en ESP32-CAM

1. En el IDE de Arduino debemos seleccionar la placa: “Ai Thinker ESP32”

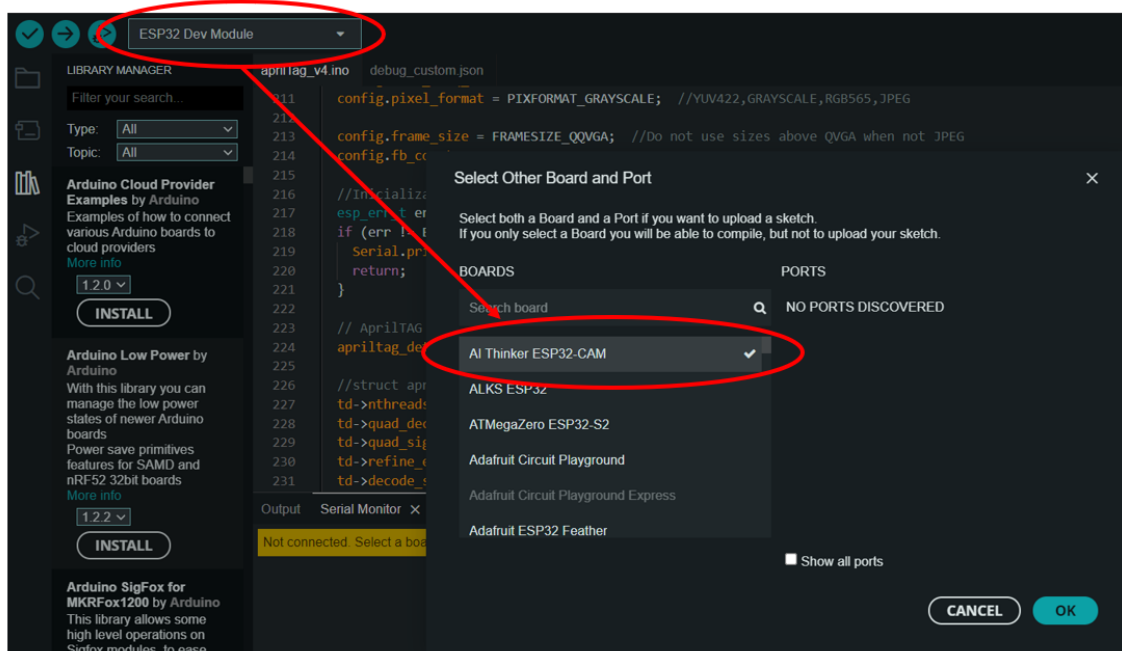


Figura 42. Seleccionar placa Ai-thinker

2. Para conectarlo al PC mediante algún tipo de adaptador, las conexiones de los pines de Rx y Tx deben ir cruzadas con los del adaptador, para poder

comunicarse. En cuanto a la alimentación es recomendable conectarla al pin de 5v, con el de 3.3v no suele funcionar correctamente.

Para poder programarla, además deberemos de puentear los pines GPIO 0 y GND. Para usarla luego, es quitar el puente y reiniciar con el botón de RESET.

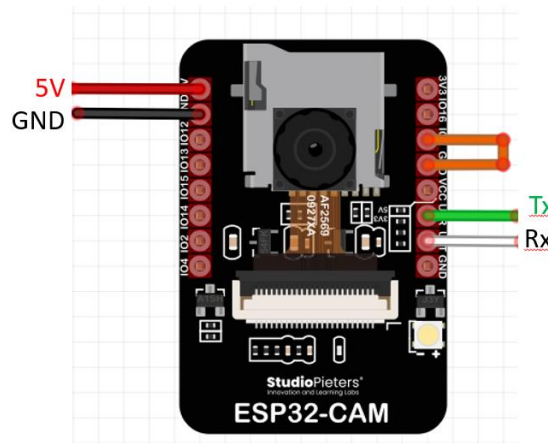


Figura 43. Programación ESP32-CAM