Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Plataforma de introducción a la programación en la infancia: Aprendiendo a jugar con los algoritmos

Alumno: **Francisco Latorre Jiménez**

Tutores: Lina Guadalupe García Cabrera
José María Serrano Chica

Dpto: Departamento de Informática

Septiembre, 2023

(Página intencionalmente en blanco)

Universidad de Jaén

Departamento de Informática

Dña. Lina Guadalupe García Cabrera y D. José María Serrano Chica, tutores del Trabajo Fin de Grado titulado: ‘Plataforma de introducción a la programación en la infancia: Aprendiendo a jugar con los algoritmos’, que presenta D. Francisco Latorre Jiménez, otorga el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2023

El alumno:

LATORRE JIMENEZ FRANCISCO - 53916968T

Firmado digitalmente por LATORRE JIMENEZ FRANCISCO - 53916968T
Fecha: 2023.09.06 01:11:50 +02'00'

Francisco Latorre Jiménez

El tutor:

GARCIA CABRERA LINA GUADALUPE - 26000612D

Firmado digitalmente por GARCIA CABRERA LINA GUADALUPE - 26000612D
Fecha: 2023.09.06 10:52:32 +02'00'

SERRANO CHICA JOSE MARIA - 26490278M

Firmado digitalmente por SERRANO CHICA JOSE MARIA - 26490278M
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-26490278M, givenName=JOSE MARIA, sn=SERRANO CHICA, cn=SERRANO CHICA JOSE MARIA - 26490278M
Fecha: 2023.09.06 08:50:09 +02'00'

Lina Guadalupe García Cabrera

José María Serrano Chica

(Página intencionalmente en blanco)

Agradecimientos

Quiero dar mi más sincero agradecimiento a todas las personas que me han apoyado y ayudado en este periodo de enfrentarme a la carrera. Primero que nada, a mi familia y amigos, que siempre me han apoyado en todo lo que he hecho y me han animado en los peores momentos a seguir adelante. A todos los compañeros que he conocido a lo largo de la carrera y que nos han servido de apoyo en los momentos más difíciles para superar cualquier reto, así como algunos profesores. Por último, a mis tutores, por darme la libertad de afrontar este proyecto de una manera más original y distinta de la que en su momento se planteó y que me ha dado fuerzas para seguir adelante.

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Tecnologías de la información
Mención (solo TFG)	Técnicas para la información y comunicación
Idioma	Español
Tipo	TFG Específico
TFG en equipo	No
Autor/a	**Francisco Latorre Jiménez**
Fecha de asignación	Curso 2021/2022
Descripción corta	Aprender a programar fomenta el pensamiento lógico y estructurado, favoreciendo el desarrollo de habilidades útiles para la resolución de problemas en nuestro día a día. Cada vez resulta más patente que la programación debería introducirse en la educación desde edades muy tempranas. El principal objetivo del presente TFG es el desarrollo de una plataforma interactiva basada en el Internet de las Cosas, donde, por medio de un juego, los niños puedan comprender cómo se construye un programa a partir de la secuencia de pasos necesarios para resolver un problema.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
Estas normas se han utilizado como base o referencia para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como proyecto la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.	

Contenido

Índice de ilustraciones

Índice de tablas

1 ESPECIFICACIÓN DEL TRABAJO

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*".

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado 8 (Definiciones y abreviaturas).

1.1 Introducción

La programación ayuda al desarrollo cognitivo de las personas, por tanto, vamos a estudiar la posibilidad de aplicar esto a los más pequeños, de manera que aprendan técnicas de lógica en una edad temprana mientras van resolviendo diferentes puzles que se les van presentando en forma de juego.

1.2 Objetivos del trabajo

El objetivo principal de este proyecto es dar a conocer las ventajas que tiene enseñar la programación en una edad temprana, mostrar algunos sistemas sencillos con los que los niños podrán divertirse a la vez que desarrollan ciertas habilidades que les serán muy importantes en el futuro, desarrollar un sistema innovador en el que los más jóvenes aprovechen las nuevas tecnologías para aprender los conceptos más

básicos de la programación y ver algunas alternativas que ya existen en el mercado actual.

1.3 Antecedentes y estado del arte

Aunque esto puede parecer una idea muy nueva, miles de escuelas, principalmente en los Estados Unidos, comenzaron a introducir la programación en su plan de estudios en la década de 1980 con la introducción del lenguaje de programación Logo. Esto llamó la atención de diferentes grupos de investigación, quienes observaron cómo los niños gracias a estas lecciones, aprendían de forma divertida, con una participación activa, en entornos reales y en directo.

LOGO es un lenguaje de programación funcional y estructurado de alto nivel que es utilizado en su mayor parte para la enseñanza a los niños y jóvenes de la programación. Su uso más frecuente es el de producir un modelo de Tortuga virtual, que se encarga de realizar las instrucciones que se van dando para que los niños puedan verlas de manera gráfica.

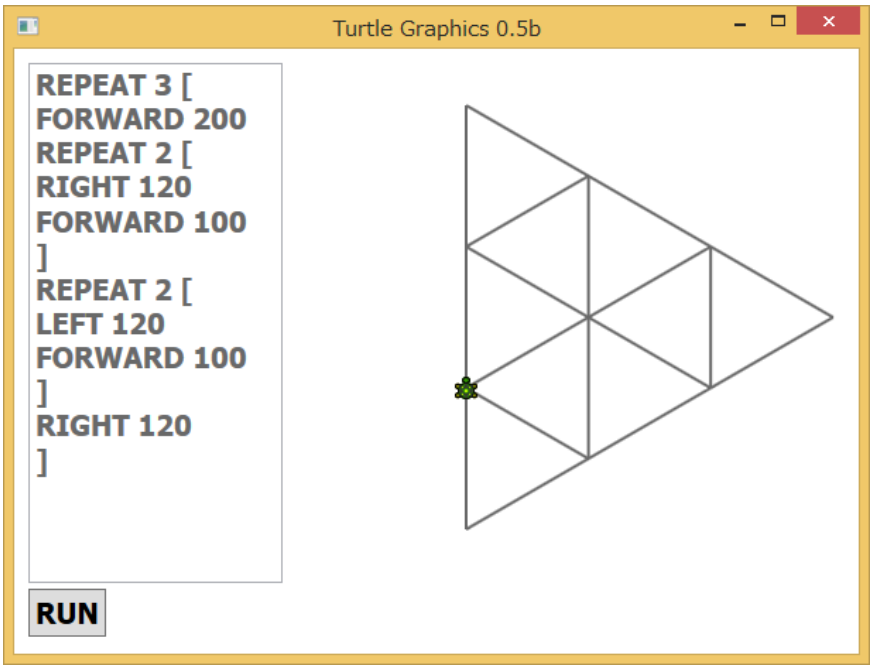


Ilustración 1.1: Lenguaje LOGO(Fuente:[6])

Gracias a este lenguaje, los niños aprendían sin dar se cuenta conceptos básicos de programación, mejorando sus capacidades lógicas y de pensamiento. En la siguiente imagen podemos ver cómo se comportaba la tortuga virtual al darle un conjunto de instrucciones básicas

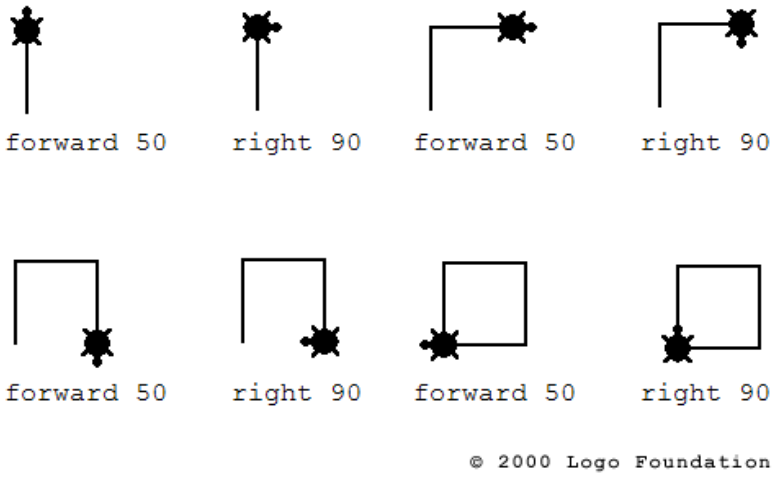


Ilustración 1.2: Movimientos LOGO(Fuente:[6])

Así, en 1986, el artículo "The Effects of Logo and CAI Environments on Cognition and Creativity" dio como conclusión que los niños que usaron Logo en la primera infancia mostraron una mayor capacidad de atención, más autonomía y se mostraron más interesados al descubrir nuevos conceptos. Asimismo, un estudio a gran escala de niños en edad preescolar que programaron con Logo (Logo y Geomtry) mostró que les fue mejor en las pruebas de matemáticas, razonamiento y resolución de problemas.

Además, investigaciones recientes han demostrado que aprender programación tiene un impacto positivo en la creatividad, las respuestas emocionales y el desarrollo de habilidades cognitivas y socioemocionales en niños con dificultades de aprendizaje.

1.4 Descripción de la situación de partida

1.4.1 Descripción del entorno actual

En la actualidad disponemos de aplicaciones y servicios web enfocados a los más pequeños que ayudan a mejorar el pensamiento lógico de los más pequeños, algunas de las más conocidas son las siguientes:

- Scratch: Un lenguaje de programación visual desarrollado por el MIT que se utiliza para enseñar a los niños a programar de manera intuitiva.

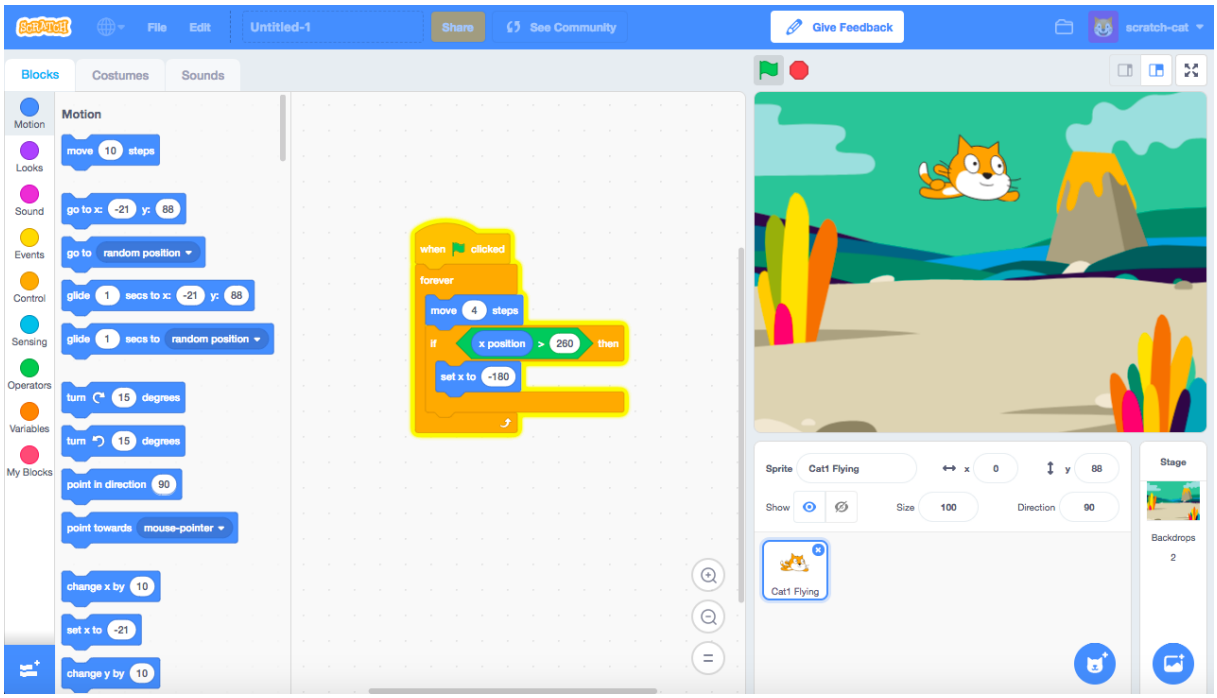


Ilustración 1.3: Programación SCRATCH(Fuente:[32])

- Code.org: Una plataforma educativa que ofrece cursos de programación para niños de todas las edades.

- Lightbot: Un juego educativo en línea que enseña a los niños los conceptos básicos de programación a través de un juego de lógica.

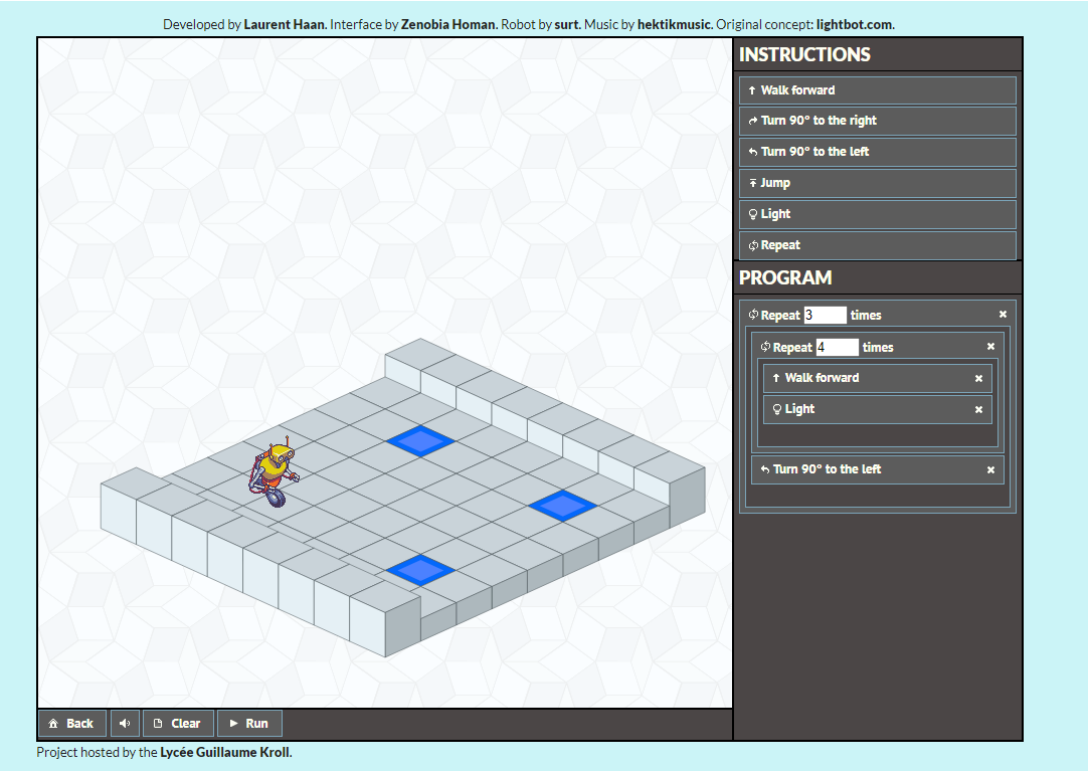


Ilustración 1.4: Juego Lightbot(Fuente:[31])

- Tynker: Una plataforma educativa que ofrece cursos de programación para niños de todas las edades, incluyendo cursos de programación de juegos y aplicaciones móviles.

Ilustración 1.5: Juego Tynker

- Code Combat: Un juego educativo en línea donde los niños aprenden programación mientras juegan un juego de rol.

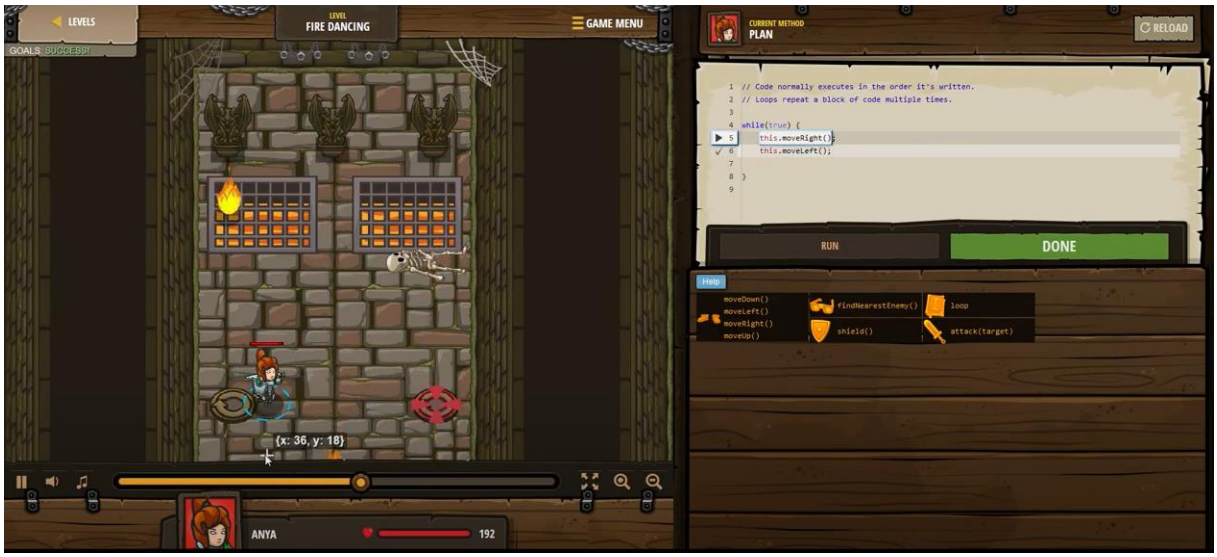


Ilustración 1.6: Juego Code Combat

- Codeacademy: Una plataforma educativa que ofrece cursos en línea de programación para niños y adultos.

Estas son algunas de las aplicaciones más utilizadas, pero existen muchas más que ayudan al aprendizaje de la programación a una temprana edad.

1.4.2 Resumen de las deficiencias y carencias identificadas

Todas las aplicaciones anteriormente mencionadas son muy buenas contribuyendo al desarrollo de los más pequeños en el mundo de la programación, pero sí que es cierto que no son perfectas.

En las primeras etapas de la educación, existe una dificultad a la hora de realizar ciertos ejercicios con un dispositivo electrónico, como por ejemplo comprender el problema que se intenta resolver, visualizar bien todas las posibilidades que tienen en un entorno desconocido para ellos, o no tener claro el uso de las herramientas de las que dispone la aplicación.

Para resolver esto se debe optar por un paisaje híbrido en el que el niño juegue con juguetes en el mundo real mientras que aprende la programación fuera de la pantalla de un ordenador. Con ayuda de un robot programado controlado por una aplicación para realizar su secuencia de movimientos. Aparte, también podrá ejercitar su imaginación creando circuitos personalizados que pueden resolver de forma activa.

Las nuevas funcionalidades que se quieren implementar son:

1. Un robot teledirigido que actúe como interfaz.
2. Una aplicación móvil con la que poder indicarle al robot cuales son los movimientos que debe realizar para llegar al destino y completar un determinado objetivo o desafío.
3. No necesitar el uso de internet para poder jugar.

4. Un tablero con el que los niños puedan interactuar y crear sus propios circuitos.
5. Utilizar símbolos y colores para hacer de la programación algo más sencillo.

1.5 Requisitos iniciales

Nuestro entorno para el aprendizaje de programación en los niños tiene que tener los siguientes requisitos iniciales:

- Debe ser un juguete en un entorno real controlado por el niño.
- Debe de tener una aplicación propia en donde poder indicarle los movimientos.
- Debe de poder ser funcional sin necesidad de internet.
- El juguete debe de poder interactuar con un tablero especializado para el proyecto.
- El niño debe de poder interactuar con el tablero para poder realizar sus propios circuitos.
- Debería de estar bajo la supervisión de un adulto para garantizar el aprendizaje del niño.

1.6 Alcance

Al finalizar nuestro trabajo, se entregará los siguientes recursos:

- Guía de uso.
- Prototipo de la aplicación móvil.
- Código fuente de la aplicación móvil.
- Prototipo de coche teledirigido.
- Código fuente del coche.

1.7 Hipótesis y restricciones

El TFG se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

Hemos determinado que la duración óptima del proyecto sería de alrededor de 3 meses. Un ingeniero informático trabajará en el proyecto a tiempo parcial.

En lugar de optar por software de pago, hemos decidido utilizar software libre para evitar cualquier costo excesivo. Ya que se trata de una aplicación web, puede ser fácilmente adaptada a dispositivos móviles o tabletas, sin necesidad de recursos o aplicaciones externas con licencias de pago. Todo lo que se requiere es tener un smartphone para poder acceder a la aplicación de control del movimiento de nuestro coche radiocontrol.

1.8 Descripción de la solución propuesta

Para nuestro vehículo, vamos a necesitar una placa arduino en donde cargar el programa con el que podamos manejar y conectar por bluetooth la aplicación de los controles desde el móvil hasta esta. Para ello hemos elegido el ESP32 sobre la placa de Arduino 1, ya que, por su capacidad de cómputo, su tamaño pequeño y su conectividad por bluetooth integrada en la placa nos viene perfecto para nuestro proyecto.

Para la programación del software, tanto para la aplicación móvil y para programar la placa vamos a utilizar en los entornos de desarrollo integrado Android Studio y Arduino IDE respectivamente. Esta decisión se ha tomado basándose en la facilidad de uso y de instalación que nos dan ambos

programas, también se ha tenido en cuenta la cantidad de información que hay en internet de estos en comparación a las otras alternativas, ya que los entornos elegidos son los más usados por los usuarios.

1.9 Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFG, no existen restricciones de tipo económico, sino de tipo temporal (un número aproximado de horas). Por consiguiente, los cálculos de tamaño del proyecto están supeditados al tiempo disponible. En cuanto al esfuerzo, se dispone de tan solo un efectivo (la persona autora del trabajo).

1.10 Planificación temporal

Para nuestro proyecto vamos a seguir el siguiente gráfico *burndown* con su correspondiente cronograma de tareas, en donde nos planificamos las 300 horas de trabajo en 52 semanas totales a partir de la fecha de inicio del proyecto.

El inicio real de nuestro proyecto se ha establecido el 31 de Julio de 2022 y la fecha de finalización estipulada sería después de 1 año (52 semanas) el 31 de Julio de 2023.

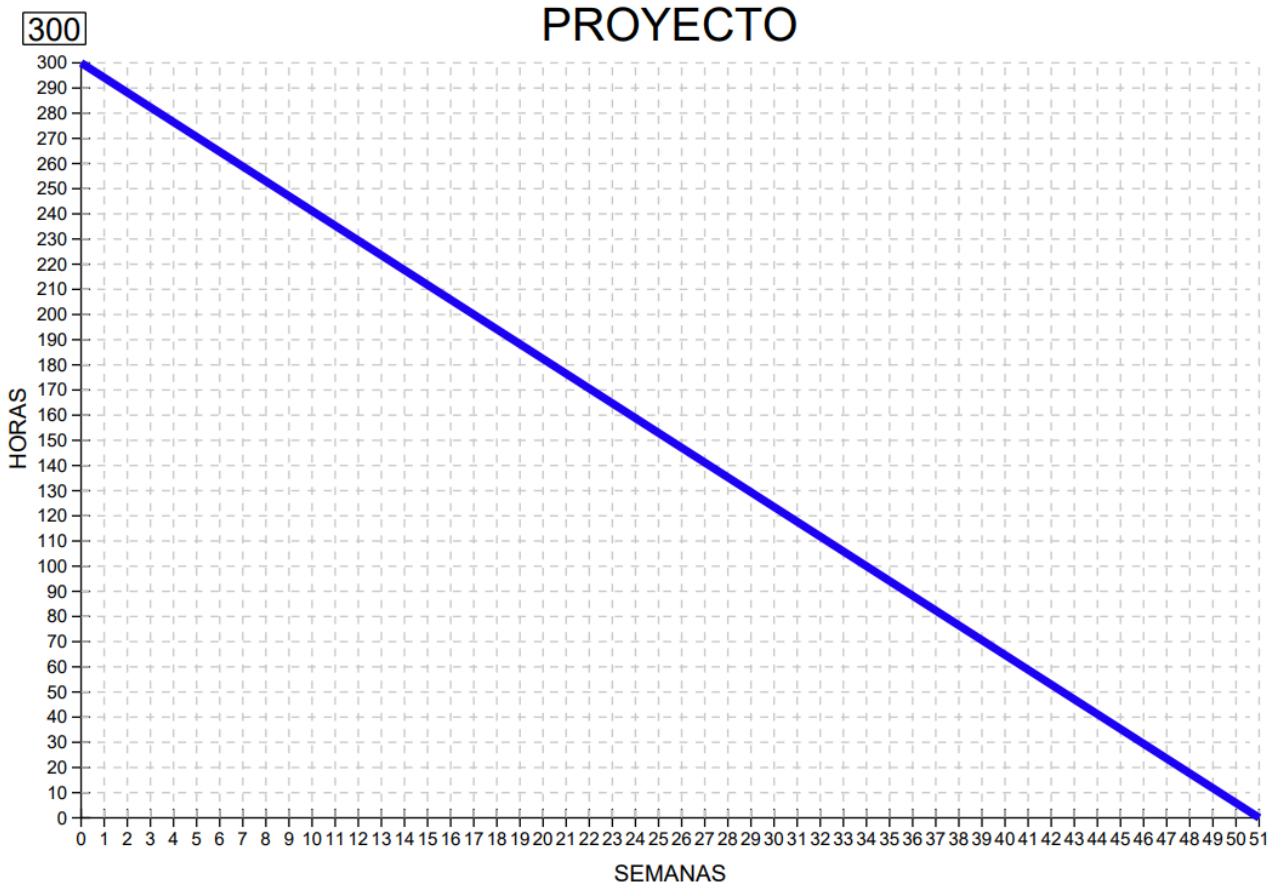


Ilustración 1.7: Gráfico burndown

En el siguiente cronograma se puede ver la asignación de las tareas y cuantas semanas totales se ha trabajado en cada una de ellas:

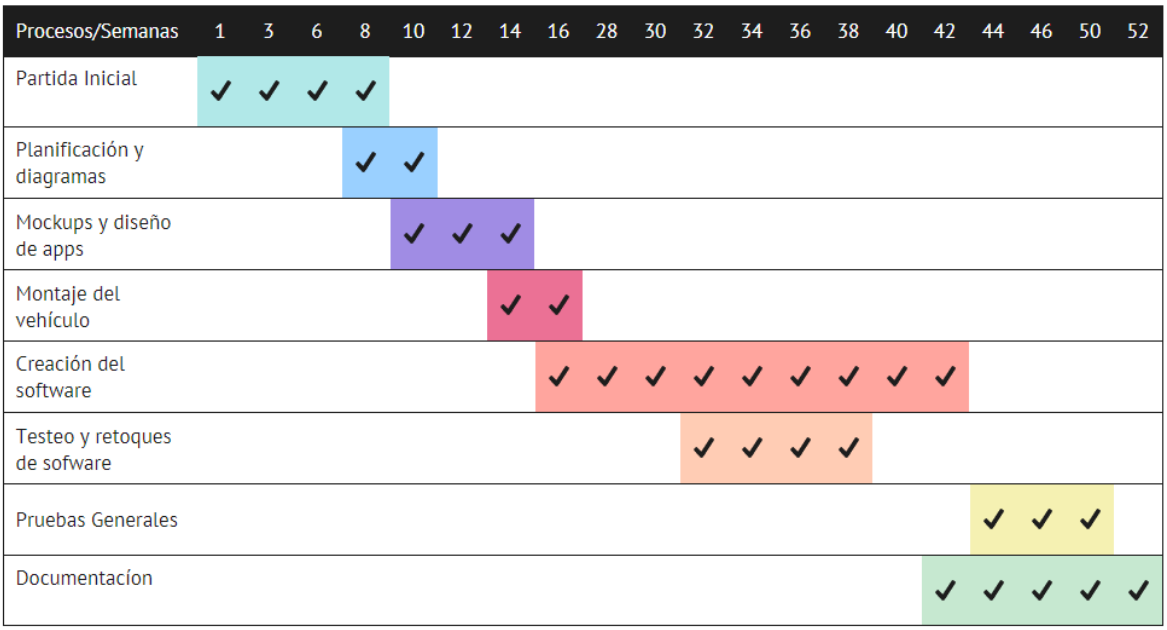


Ilustración 1.8: Cronograma de tareas

1.11 Presupuesto

RECURSO	UNIDADES	TIEMPO DE USO	COSTE GLOBAL	COSTE PARCIAL
Placa ESP32	1	1 AÑO	10,99 €	10,99 €
Pack Base del vehículo + Ruedas + Motores	1	1 AÑO	23,85 €	23,85 €
Controladora L298N	1	1 AÑO	3,99 €	3,99 €
Sensor *siguelíneas*	2	1 AÑO	6,00 €	6,00 €
Sensor de color TCS3200	1	1 AÑO	9,99 €	9,99 €
Pilas 12V	5	1 AÑO	10,00 €	10,00 €

Cableado	1	1 AÑO	6,98 €	6,98 €
Batería portátil	1	1 AÑO	8,00 €	8,00 €
Monitor	1	1 AÑO	220 €	110,00 €
PC de sobremesa	1	1 AÑO	1700,00 €	850,00 €
Windows 11	1	1 AÑO	127,99 €	63,99 €
Ratón	1	1 AÑO	23,00 €	11,50 €
Teclado	1	1 AÑO	30,00 €	15,00 €
Sueldo ingeniero Junior	1	1 AÑO	20.450,00 €	10.225,00 €
Costes indirectos (Suplemento del 10%)	1	1 AÑO	2.263,08 €	1.131,54 €
COSTE TOTAL				**12.486,83 €**

Tabla 1.1: Presupuestos

Los precios del hardware son los precios más repetidos en la consultas realizadas en la web y en su mayoría del precio que me costó cuando lo compré. El salario del ingeniero junior se ha consultado en la web de la Universidad de Alfonso X el Sabio, en un artículo donde se cita: "*Un Ingeniero Informático recién egresado, con menos de 3 años de experiencia laboral, puede esperar un salario medio de 20.450 euros brutos por año*"(disponible en **https://www.uax.com/blog/ingenieria/cuanto-cobra-un-ingeniero-informatico**)

1.12 Estructura de la memoria

La memoria se ha estructurado en 7 apartados principales, cada uno con sus respectivos apartados secundarios. La memoria de este proyecto cuenta con las siguientes secciones:

1. Es el apartado en el que nos encontramos y es un resumen de los antecedentes de nuestro proyecto y de todo lo que vamos a ver a lo largo de la memoria. Sirve como introducción de la misma.
2. El segundo apartado está enfocado en el análisis y diseño del sistema que vamos a desarrollar, revisando todos los apartados del desarrollo como el diseño del software y hardware, un análisis de las mejores herramientas que hay en el mercado para nuestro trabajo y una elección de las mismas. También se ve una breve explicación de la metodología de trabajo que vamos a implementar y todo lo necesario para nuestro desarrollo.
3. En el tercer apartado se verá en profundidad qué se ha desarrollado y cómo se ha organizado el proyecto. Se verán las distintas iteraciones realizadas y cómo se han solventado los problemas encontrados. También veremos las pruebas que se han realizado en cada una de las iteraciones y las pruebas finales para comprobar si nuestro sistema funciona correctamente.
4. En el apartado número cuatro se verá todo lo relacionado con las posibles aplicaciones que tiene nuestro sistema y un buen uso del mismo, con algunos de los muchos ejemplos que tiene para el sector de la enseñanza.
5. El quinto apartado lo dedicaremos a realizar una conclusión general del proyecto, así como un análisis propio y unas posibles mejoras futuras.
6. En el apartado seis se verán los anexos adjuntos al proyecto: un manual de instalación de la app móvil, un manual de uso de nuestra aplicación, y por último, unas plantillas con el diseño para que los profesores o padres de los niños puedan construir el circuito que quieran y donde quieran.
7. Por último, en el apartado número siete se verá la bibliografía que se ha consultado para la realización de todo nuestro trabajo.

2 DISEÑO INICIAL

En el siguiente apartado se va a especificar todo el diseño previo que se ha tenido en cuenta a la hora de empezar a desarrollar nuestro trabajo, acompañado de

elementos visuales con los que poder hacernos una idea de que es lo que se quiere desarrollar.

2.1 Especificaciones del sistema

El proceso de desarrollo de nuestro sistema ha consistido en las siguientes fases:

- Diseñar un vehículo motorizado hecho por un módulo programable preparado para interpretar el entorno y recibir los comandos de movimiento del usuario.
- Realizar una aplicación que se complemente con el entorno en cuestión, en donde indicarle al vehículo motorizado cuales son los movimientos que debería de realizar para poder llegar a su destino sin salirse del camino.
- Por último, diseñar el tablero donde, de manera visual se construirá nuestro circuito. El tablero estará compuesto de unas líneas o carriles que indicarán por donde se ha de mover nuestro vehículo sin salirse y dispondrá de unas señalizaciones de colores para que el coche interprete el entorno y sepa reconocer si el camino es correcto o si por el contrario se ha salido de él.

De este modo, se satisface el objetivo principal del proyecto, proporcionar un entorno en el que los niños puedan aprender las bases de la programación con una interacción fácil, divertida y tangible.

2.2 Metodología de desarrollo de software

La metodología de desarrollo que se ha decidido seguir ha sido la metodología Scrum.

La metodología de desarrollo ágil Scrum es un enfoque iterativo e incremental utilizado para gestionar proyectos de desarrollo de software. Se basa en principios de colaboración, adaptabilidad y entrega continua de valor. Scrum se organiza en ciclos

de trabajo llamados Sprints, que suelen tener una duración de 1 a 4 semanas. A continuación, se describen los elementos principales de Scrum:

1. Roles:

- *Product Owner*: Es responsable de definir y priorizar los requisitos del proyecto, representando los intereses de los stakeholders.

- *Scrum Master*: Es el facilitador del equipo, asegurando que se sigan los principios y prácticas de Scrum y eliminando obstáculos.

- Equipo de Desarrollo: Son los miembros encargados de realizar el trabajo, colaborando estrechamente para entregar incrementos de valor al final de cada Sprint.

2. Artefactos:

- *Product Backlog*: Es una lista priorizada de requisitos y funcionalidades que se mantienen y actualizan durante todo el proyecto.

- *Sprint Backlog*: Es una lista de elementos seleccionados del Product Backlog para el Sprint actual. Define el trabajo a realizar durante el Sprint.

3. Eventos:

- *Sprint Planning*: Reunión donde se planifica el trabajo a realizar durante el próximo Sprint, definiendo los objetivos y seleccionando los elementos del Product Backlog que se abordarán.

- *Daily Scrum*: Reunión diaria breve en la que el equipo de desarrollo sincroniza su trabajo, revisa el progreso y detecta posibles obstáculos.

- *Sprint Review*: Reunión al final del Sprint en la que se revisa y se demuestra el trabajo completado, obteniendo retroalimentación de los stakeholders.

- *Sprint Retrospective*: Reunión en la que el equipo reflexiona sobre el Sprint completado, identifica mejoras y define acciones para implementarlas en los siguientes Sprints.

Ventajas de Scrum:

- Mayor flexibilidad y adaptabilidad a los cambios en los requisitos del proyecto.
- Entrega temprana y continua de valor al cliente o usuario final.
- Mayor transparencia y colaboración dentro del equipo y con los stakeholders.
- Capacidad de aprendizaje y mejora continua a través de la retroalimentación.

Desventajas de Scrum:

- Requiere un compromiso y participación activa de todos los miembros del equipo.
- No es adecuado para proyectos de gran escala o altamente regulados sin adaptaciones.
- Puede haber una curva de aprendizaje inicial para aquellos que no están familiarizados con los conceptos y prácticas de Scrum.

En resumen, Scrum es una metodología ágil que promueve la entrega iterativa y continua de valor, fomenta la colaboración y la adaptabilidad. Al dividir el proyecto en ciclos de trabajo cortos y enfocarse en la entrega de incrementos de valor, Scrum permite una mayor flexibilidad y capacidad de respuesta a los cambios en los requisitos del proyecto.

2.3 Estudio de alternativas y viabilidad

Se han estudiado varias alternativas para la realización del proyecto, tanto en el software utilizado para la aplicación móvil como en el tipo de placa a utilizar en el coche como su software de programación correspondiente.

2.3.1 Software para la aplicación móvil

Para el desarrollo de la aplicación móvil se han considerado las siguientes tecnologías:

2.3.1.1 Android Studio

Ilustración 2.1: Android Studio logo(Fuente:[10])

Android Studio es un entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android, basado en el software JetBrains IntelliJ IDEA. Para soportar el desarrollo de aplicaciones sobre el sistema operativo Android, Android Studio utiliza un sistema de construcción basado en Gradle, emuladores, plantillas de código e integración con Github. Android Studio es el entorno de desarrollo integrado (IDE) oficial de Google para el sistema operativo Android, construido sobre el software

JetBrains IntelliJ IDEA y diseñado específicamente para el desarrollo de Android. El sistema de compilación de Android es un conjunto de herramientas que se utilizan para crear, probar, ejecutar y empaquetar las aplicaciones.

Ventajas:

- Integración con Google Play Store: Android Studio está completamente integrado con la tienda de aplicaciones de Google, lo que permite a los desarrolladores publicar y actualizar sus aplicaciones con facilidad.

- Depuración eficiente: Android Studio cuenta con una herramienta de depuración robusta que permite a los desarrolladores solucionar problemas y mejorar la experiencia del usuario.

- Código ordenado: La herramienta de Android Studio utiliza un sistema de formato automático para mantener el código limpio y ordenado, lo que facilita la lectura y mantenimiento.

- Amplia documentación: Android Studio cuenta con una amplia documentación y recursos en línea que ayudan a los desarrolladores a aprovechar al máximo la plataforma y a aprender sobre el desarrollo de aplicaciones móviles.

Desventajas:

- Recursos exigentes: Android Studio es una herramienta pesada que requiere una gran cantidad de recursos, especialmente RAM, lo que puede ser un problema para computadoras antiguas o de baja gama.

- Instalación y actualización lentas: La instalación y actualización de Android Studio pueden ser tediosas debido a su tamaño y a la cantidad de dependencias necesarias.

- Curva de aprendizaje: Aunque Android Studio es una herramienta poderosa, puede ser complicada de aprender para aquellos acostumbrados a otras plataformas o herramientas.

- Problemas de compatibilidad con plugins: Aunque Android Studio incluye muchos plugins integrados, es posible que algunos plugins externos no sean compatibles con la última versión de la herramienta, lo que puede causar problemas para los desarrolladores.

2.3.1.2 Flutter

Ilustración 2.2: Flutter logo(Fuente:[12])

Flutter es un framework de interfaz de usuario móvil gratuito y de código abierto creado por Google y lanzado en mayo de 2017. Permite crear una aplicación móvil nativa con una sola base de código. Esto significa que puedes usar un lenguaje de programación y una base de código para crear dos aplicaciones diferentes (para iOS y Android).

Ventajas:

- Desarrollo rápido: Flutter permite a los desarrolladores crear rápidamente aplicaciones móviles de alta calidad gracias a una biblioteca de widgets personalizables y un sistema de "recarga en caliente" que acelera el ciclo de desarrollo.

- Diseño atractivo: Flutter proporciona una variedad de herramientas para diseñar hermosas interfaces de usuario personalizadas, lo que permite a los

desarrolladores crear aplicaciones móviles visualmente sorprendentes y fáciles de usar.

- Compatibilidad multiplataforma: Las aplicaciones creadas con Flutter se pueden usar en dispositivos móviles iOS y Android, lo que reduce los costos y el tiempo de desarrollo al eliminar la necesidad de crear aplicaciones separadas para cada plataforma.

- Alta productividad: Las aplicaciones creadas con Flutter funcionan muy bien, con una experiencia de usuario fluida y fluida, incluso en dispositivos más antiguos.

Desventajas:

- Tamaño del archivo de la aplicación: Las aplicaciones Flutter pueden tener archivos de mayor tamaño que las aplicaciones nativas, debido a la necesidad de incluir el motor Flutter en la aplicación. Esto puede afectar los tiempos de descarga y la cantidad de almacenamiento requerido por las aplicaciones.

- Recursos y bibliotecas limitadas: Aunque Flutter tiene una comunidad de desarrolladores en crecimiento, la disponibilidad de paquetes y bibliotecas de terceros puede ser más limitada que en plataformas más antiguas. Puede tomar más tiempo y esfuerzo desarrollar ciertas funciones o encontrar soluciones específicas.

- Curva de aprendizaje: Si bien Flutter usa el lenguaje de programación Dart, que es relativamente fácil de aprender, la curva de aprendizaje puede ser más pronunciada para aquellos que no están familiarizados con el desarrollo móvil

o la arquitectura de Flutter. Puede llevar más tiempo acostumbrarse a los conceptos y patrones de Flutter.

- Problemas de rendimiento en animaciones complejas: Aunque Flutter es conocido por su alto rendimiento, aún pueden ocurrir problemas de rendimiento cuando se usan animaciones complejas o de alto nivel. Esto se debe a la necesidad de procesar y renderizar gráficos y animaciones en tiempo real, lo que puede afectar el rendimiento en dispositivos más antiguos o con recursos limitados.

- Framework joven con poca comunidad: Aunque la comunidad de Flutter está creciendo rápidamente, aún es más pequeña que las comunidades de otras plataformas de desarrollo móvil más antiguas. Esto puede significar que puede haber menos recursos y apoyo disponibles que las opciones más establecidas.

En general, Flutter es una tecnología eficiente y poderosa para el desarrollo de aplicaciones móviles, que permite a los desarrolladores crear aplicaciones de alta calidad rápidamente, con un diseño atractivo, multiplataforma y alto rendimiento.

2.3.1.3 Ionic

Ilustración 2.3: Ionic logo(Fuente:[14])

Ionic Framework es un SDK de frontend de código abierto para desarrollar aplicaciones híbridas basado en tecnologías web (HTML, CSS y JS). Es decir, un framework que nos permite desarrollar aplicaciones para iOS nativo, Android y la web, desde una única base de código.

Ventajas de ionic:

- Desarrollo multiplataforma: Ionic le permite desarrollar aplicaciones móviles para múltiples plataformas, incluidos iOS, Android y la web, utilizando tecnologías web estándar como HTML, CSS y JavaScript. Esto facilita el desarrollo de aplicaciones que funcionan en diferentes dispositivos y sistemas operativos.

- Comunidad grande y solidaria: Ionic tiene una comunidad de desarrolladores activa, lo que significa que hay muchos recursos, tutoriales y complementos disponibles. También cuenta con el respaldo de un desarrollador líder de herramientas de desarrollo móvil.

- Facilidad de uso y velocidad de desarrollo: Ionic proporciona una interfaz de usuario fácil de usar y una biblioteca de componentes preconstruidos que facilitan la creación de hermosas interfaces de usuario. Además, su enfoque basado en la web permite un desarrollo rápido y eficiente al reutilizar el código existente y el conocimiento de desarrollo web.

- Integración con Angular: Ionic está estrechamente integrado con el popular marco Angular, lo que proporciona un marco sólido y una arquitectura modular para el desarrollo de aplicaciones. Esto facilita la creación de aplicaciones robustas y escalables.

Desventajas de Ionic:

- Eficiencia: Dado que Ionic se basa en tecnología web, las aplicaciones pueden tener un rendimiento ligeramente inferior al de las aplicaciones nativas. Esto puede notarse especialmente en aplicaciones que requieren un alto rendimiento de gráficos o usan varios procesadores. Restrinja el acceso a funciones nativas:

- Si bien Ionic ofrece una amplia gama de complementos y extensiones para acceder a la funcionalidad nativa en los dispositivos, puede haber limitaciones para proporcionar aplicaciones nativas de acceso completo. Esto puede ser difícil si se requiere una integración profunda con dispositivos específicos o características del sistema operativo.

- Tamaño de la aplicación: Las aplicaciones iónicas pueden tener tamaños de archivo relativamente más grandes que las aplicaciones nativas. Esto se debe a la necesidad de incluir el motor Ionic y las dependencias requeridas en la aplicación.

- Adicción al sitio web: Ionic se basa en tecnología web, por lo que depende del navegador web interno para representar y ejecutar la interfaz de usuario. Esto puede causar limitaciones y un comportamiento inconsistente en diferentes dispositivos y navegadores.

2.3.2 Placa utilizada

A continuación, se describen las características de las placas utilizadas en el diseño del vehículo motorizado.

2.3.2.1 Arduino UNO

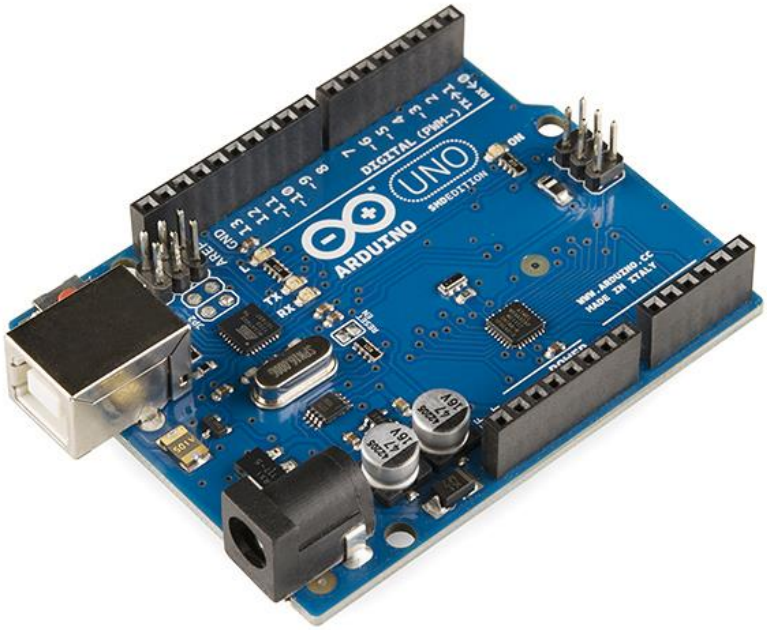


Ilustración 2.4: Arduino UNO(Fuente:[16])

Arduino Uno es una placa de desarrollo de código abierto basada en el microcontrolador ATmega328P. Es una de las placas Arduino más populares y utilizadas debido a su facilidad de uso y versatilidad. Aquí hay una descripción más detallada y pros y contras del Arduino Uno:

Descripción de Arduino Uno:

- Microcontroladores: Arduino Uno utiliza el microcontrolador ATmega328P, que tiene una velocidad de reloj de 16 MHz y 32 KB de memoria flash programable.

- Pines de entrada/salida (E/S): La placa tiene 14 pines de E/S digitales, 6 de los cuales se pueden usar como salidas PWM (modulación de ancho de pulso) y 6 como entradas analógicas.

- Conectar: El Arduino Uno incluye un puerto USB para comunicarse con la computadora, así como pines de alimentación y tierra para conectar dispositivos y componentes externos. Programa:

- Se puede programar utilizando el entorno de desarrollo integrado (IDE) de Arduino, basado en el lenguaje de programación C++ simplificado.

Ventajas de Arduino Uno:

- Fácil de usar: El Arduino Uno es conocido por ser apto para principiantes en electrónica y programación. El IDE intuitivo y el lenguaje de programación simple facilitan el aprendizaje y el desarrollo de proyectos.

- Compatibilidad extendida: La placa es compatible con una amplia gama de sensores, actuadores y módulos electrónicos disponibles en el mercado, lo que facilita la ampliación de la funcionalidad del proyecto.

- Gran comunidad y apoyo: Arduino Uno tiene una comunidad de usuarios activa y una amplia base de conocimientos. Esto facilita el acceso a tutoriales, ejemplos de código y foros de soporte, brindando ayuda y soluciones a los usuarios.

- Costo asequible: El Arduino Uno es una alternativa económica a otras placas y sistemas integrados actualmente en el mercado, lo que lo hace accesible para estudiantes y proyectos de bajo presupuesto.

Desventajas de Arduino Uno:

- Recursos limitados: La tarjeta tiene limitaciones de memoria y capacidad de procesamiento, lo que puede ser una limitación en proyectos más complejos o que requieran mayor capacidad.
- Límite de conexión: El Arduino Uno no tiene conectividad Wi-Fi o Bluetooth incorporada, lo que puede requerir módulos adicionales para agregar estas funciones.

- Tamaño físico y límite de batería: La placa Arduino Uno es relativamente grande y tiene una cantidad limitada de pines de E/S, lo que puede limitar la cantidad de componentes que se pueden conectar directamente.

En resumen, Arduino Uno es una placa de desarrollo popular debido a su facilidad de uso, amplia compatibilidad y comunidad de apoyo. Sin embargo, sus limitaciones en términos de recursos, conectividad y tamaño.

2.3.2.2 ESP32

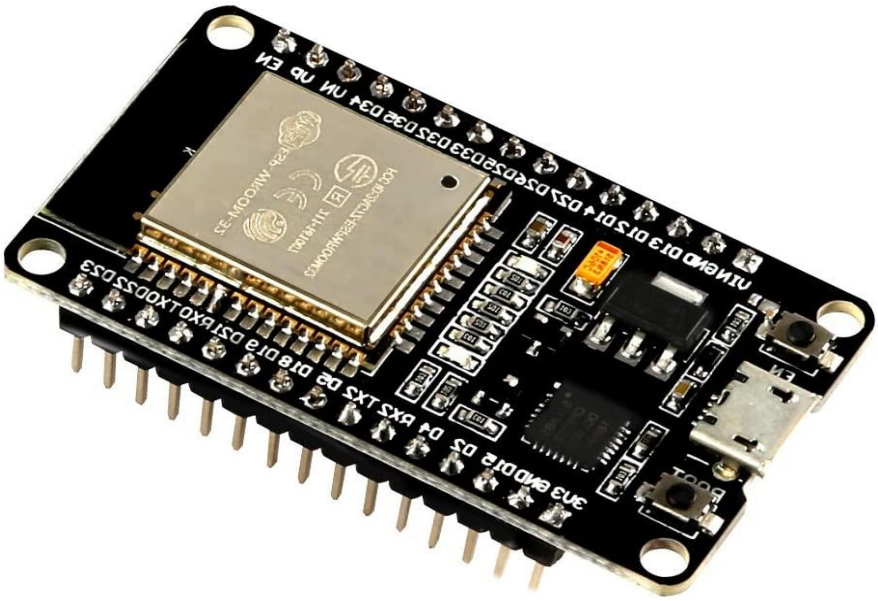


Ilustración 2.5: Placa ESP32(Fuente:[15])

El ESP32 es una placa de desarrollo basada en el microcontrolador Espressif Systems ESP32. Estas son algunas de las ventajas que ofrece esta placa:

- Potencia y rendimiento: El ESP32 es un microcontrolador de doble núcleo con una velocidad de reloj de hasta 240 MHz, lo que le proporciona una gran potencia de procesamiento, lo que lo hace adecuado para aplicaciones que requieren computación intensiva y alto rendimiento.

- Conectividad inalámbrica: El ESP32 viene con conectividad Wi-Fi incorporada, lo que permite la conexión a redes Wi-Fi y comunicación inalámbrica. También está habilitado para Bluetooth, lo que brinda la capacidad de conectarse y comunicarse con otros dispositivos Bluetooth, como teléfonos inteligentes, tabletas y otros dispositivos IoT.

- Varios tipos de periféricos: El ESP32 tiene una variedad de periféricos integrados como puertos GPIO, UART, SPI, I2C y ADC que facilitan la conexión y el control de sensores, actuadores y otros dispositivos externos.

- Flexibilidad y flexibilidad: La placa ESP32 es muy versátil y se puede utilizar para muchos proyectos y aplicaciones. Se puede programar utilizando el lenguaje de programación Arduino y también es compatible con el entorno de desarrollo ESP-IDF de Espressif.

- Bajo consumo de energía: El ESP32 ofrece opciones de administración de energía eficientes, lo que permite su uso en aplicaciones de bajo consumo o con restricciones de energía. También cuenta con modos programables de activación y suspensión profunda, que prolongan la duración de la batería en los dispositivos compatibles.

- Comunidad grande y solidaria: ESP32 tiene una gran comunidad de desarrolladores, lo que facilita la búsqueda de recursos, tutoriales y ejemplos en línea. Además, Espressif Systems brinda documentación y soporte técnico para ayudar a los desarrolladores a usar la tarjeta de manera eficiente.

En resumen, el ESP32 ofrece una combinación de potencia, conectividad, flexibilidad y eficiencia energética, lo que lo convierte en una opción popular para IoT, proyectos y aplicaciones de domótica, electrónica y muchos otros campos.

2.3.3 Software para programar la placa

2.3.3.1 Arduino IDE

El software que vamos a utilizar para programar la placa ESP32 va a ser el de Arduino IDE, se ha realizado una comparativa con distintos IDEs en el mercado como por ejemplo Visual Studio con Arduino o Eclipse para Arduino entre otros, pero frente a estos, se ha elegido la IDE de Arduino por las siguientes razones:

- Fácil de usar: El IDE de Arduino está diseñado para ser intuitivo y fácil de usar, especialmente para que los principiantes aprendan programación y electrónica. La interfaz simple y clara facilita la escritura, compilación y carga de código en la placa Arduino.

- Compatibilidad con una variedad de placas base: El IDE de Arduino es compatible con una amplia variedad de placas Arduino, lo que significa que puede programar y trabajar con diferentes modelos de Arduino sin tener que aprender varios IDE.

- Gran comunidad y apoyo: Arduino tiene una gran comunidad de usuarios y una base de conocimiento activa. Esto significa que hay muchos tutoriales, ejemplos de código, proyectos compartidos y foros de soporte en línea para ayudarlo a aprender y solucionar problemas.

- Prioriza la simplicidad: Arduino IDE tiene como objetivo hacer que la programación y el desarrollo de proyectos sean accesibles para todos. Utiliza un lenguaje de programación simplificado basado en C/C++, lo que facilita la escritura y la comprensión del código, incluso para aquellos que no tienen mucha experiencia en programación.

- Integración con Bibliotecas y Ejemplos: El IDE de Arduino incluye una amplia gama de bibliotecas y ejemplos predefinidos que facilitan la integración de funciones adicionales en sus proyectos. Te ahorra tiempo al no tener que escribir todo el código desde cero y te permite beneficiarte de la experiencia de otros desarrolladores.

- Soporte multiplataforma: El IDE de Arduino está disponible para diferentes sistemas operativos, como Windows, macOS y Linux, lo que le permite usarlo en la plataforma que elija.

Con todo, Arduino IDE se destaca por su simplicidad, facilidad de uso, compatibilidad con varias placas Arduino y el amplio soporte brindado por su comunidad. Estos beneficios hacen que Arduino IDE sea una opción popular y asequible para aquellos que desean involucrarse en la programación y el desarrollo de proyectos con placas Arduino.

2.3.3.2 Visual Studio Con Arduino

Ilustración 2.6: Visual Studio con Arduino

Visual Studio con Arduino es una combinación del entorno de desarrollo integrado (IDE) Visual Studio de Microsoft y la plataforma de desarrollo Arduino de código abierto. Permite a los desarrolladores programar y cargar código en la placa Arduino utilizando la familiaridad y la potencia del entorno de Visual Studio. Estos son los pros y los contras de usar Visual Studio con Arduino:

Descripción de Visual Studio con Arduino:

- Totalmente integrado: Visual Studio ofrece una integración completa con Arduino, lo que le permite desarrollar y depurar proyectos Arduino utilizando todas las funciones y herramientas disponibles en Visual Studio.

- Admite varios idiomas: Visual Studio es compatible con varios lenguajes de programación, incluidos C++ y C#, lo que le brinda la flexibilidad para desarrollar proyectos Arduino con diferentes enfoques y requisitos.

- Herramientas avanzadas: Visual Studio proporciona herramientas avanzadas de depuración, finalización de código, refactorización y administración de proyectos que pueden optimizar y mejorar el proceso de desarrollo de proyectos Arduino.

- Capacidad de extensión: Visual Studio permite la instalación de extensiones y complementos adicionales, lo que brinda personalización adicional y funcionalidad extendida a los proyectos de Arduino.

Ventajas de Visual Studio con Arduino:

- Ambiente familiar y poderoso: Visual Studio es ampliamente utilizado y conocido por su potencia y funcionalidad, lo que le permite aprovechar su interfaz fácil de usar y sus herramientas avanzadas en proyectos Arduino.

- Depuración avanzada: Visual Studio proporciona potentes herramientas de depuración que facilitan la identificación y corrección de errores en el código de Arduino, lo que mejora la calidad y la confiabilidad del proyecto.

- Gran comunidad de desarrolladores: Visual Studio tiene una gran comunidad de desarrolladores que comparten recursos, tutoriales y ejemplos de código, lo que facilita el acceso a la ayuda y el soporte en línea.

- Escalabilidad y flexibilidad: Visual Studio permite la integración de diversas herramientas y servicios, como control de versiones, pruebas automatizadas y análisis de código estático, lo que proporciona una mayor flexibilidad y personalización en los proyectos de Arduino.

Desventajas de Visual Studio con Arduino:

- Curva de aprendizaje inicial: Visual Studio puede tener una curva de aprendizaje más pronunciada que el IDE estándar de Arduino, especialmente para aquellos que no están familiarizados con el entorno de Visual Studio.

- Requerimientos de recursos: Visual Studio es una aplicación más pesada en recursos de hardware y espacio de almacenamiento, lo que puede ser una consideración para aquellos con sistemas más antiguos o con recursos limitados.

- Dependencias de la plataforma Windows: Visual Studio es un IDE desarrollado por Microsoft y proporciona una mejor integración y soporte en el sistema operativo Windows. Esto puede ser una limitación para aquellos que prefieren trabajar en otros sistemas operativos.

En resumen, Visual Studio con Arduino combina el poder y la funcionalidad de Visual Studio con la plataforma de desarrollo Arduino, ofreciendo beneficios como un entorno potente y familiar, herramientas de depuración avanzadas y extensibilidad. Sin embargo, puede tener una curva de aprendizaje inicial y puede

2.4 Tecnologías utilizadas

Las tecnologías utilizadas para la parte mecánica del coche son la placa ESP32, 2 motores y una placa extra que sirve de controladora para los motores, esta placa es la L298N. Con estas piezas podemos montar la parte mecánica de nuestro vehículo, acompañado del software necesario para su correcto funcionamiento. También utiliza 2 módulos sensor *sigue-líneas* IR TCRT5000. Estas se explicarán más adelante.

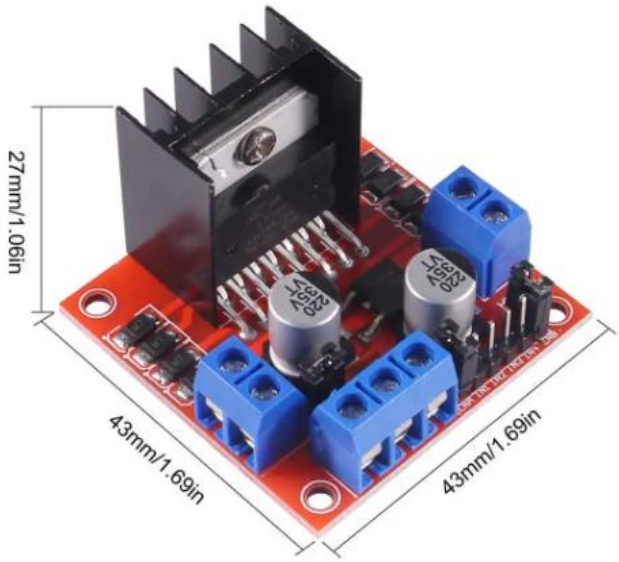


Ilustración 2.7: Controladora L298N(Fuente:[33])

Ilustración 2.8: Motores y ruedas(Fuente:[33])

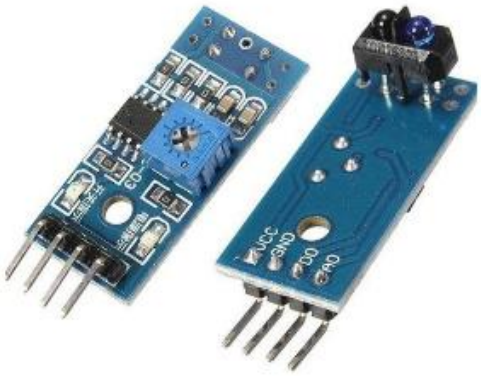


Ilustración 2.9: Sensores sigue-líneas(Fuente:[33])

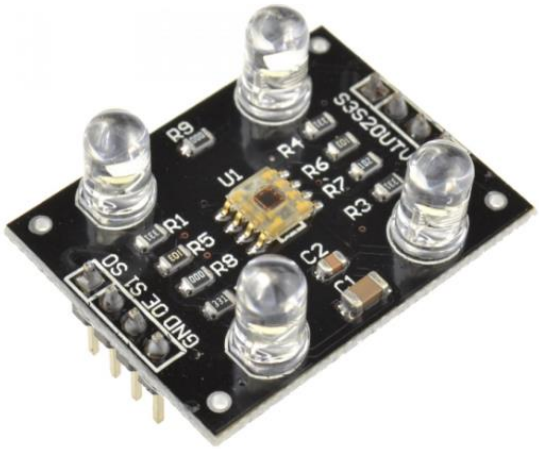


Ilustración 2.10: Sensor de color(Fuente:[33])

Para la parte del software vamos a utilizar los siguientes programas:

- Arduino IDE versión 2.0.2 con el lenguaje de programación C para la realización del software.
- Librería BluetoothSerial.h junto con un módulo externo para la placa ESP32, ya que la instalación inicial no viene preparada para el uso de esta placa.
- Android Studio Chipmunk | 2021.2.1 Patch 1 con lenguaje de programación en java es el software e idioma usado para programar la aplicación móvil de control de nuestro vehículo. Se usan librerías internas para la gestión del bluetooth por donde conectarnos con el coche.

2.5 Análisis y diseño del sistema

En este apartado vamos a ver el análisis y el diseño de todos los apartados que se han tenido en cuenta a la hora de realizar el prototipo, poniendo especial atención en todas sus funcionalidades y qué medidas se han tomado para poder satisfacer todas las especificaciones adoptadas en el diseño.

2.5.1 Análisis y diseño del vehículo motorizado

Para el diseño de nuestro coche vamos a necesitar realizar un análisis y diseño de las 2 partes fundamentales del mismo:

- la parte hardware, que va a ser todo el sistema físico y eléctrico que necesitamos, y
- la parte software, que va a ser el código que tenemos que programar para que todos nuestros sensores y motores funcionen en armonía para hacer funcionar nuestro juguete.

2.5.1.1 Diseño Hardware

El diseño de la circuitería propuesto es el siguiente:

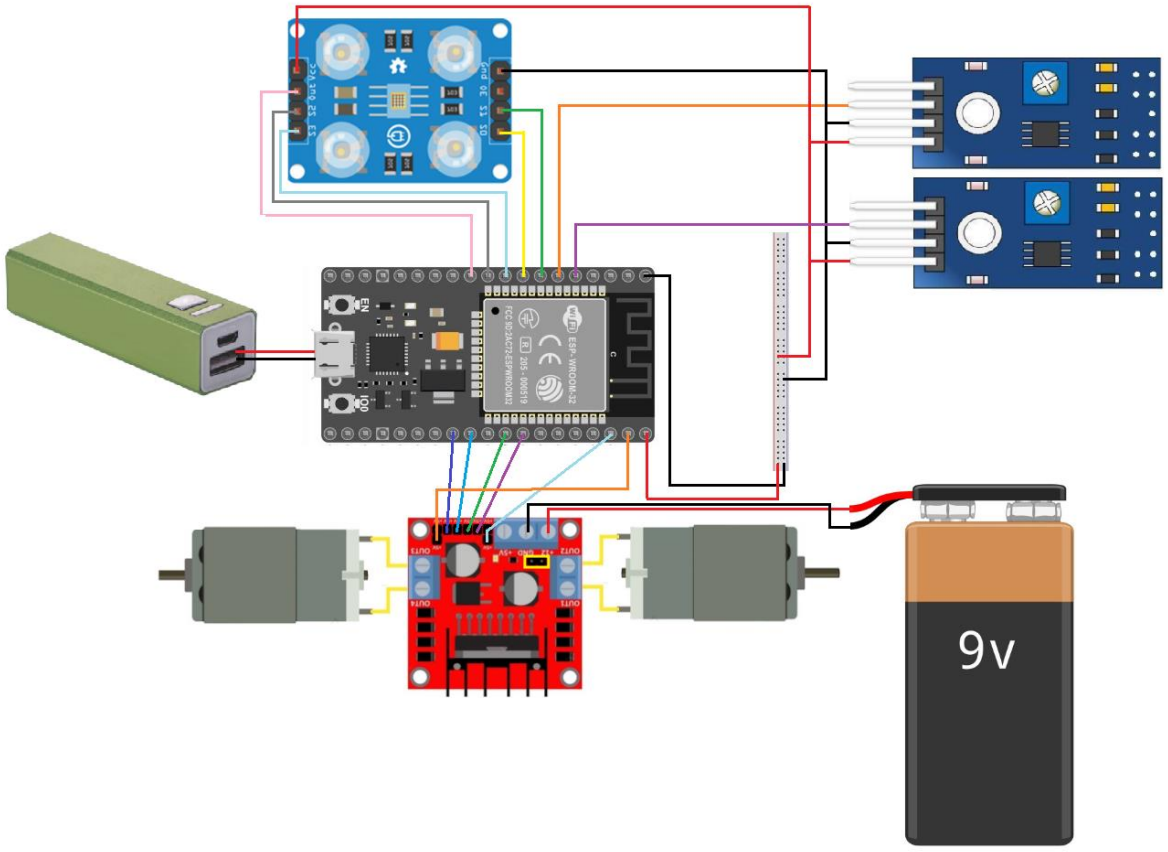


Ilustración 2.11: Diseño electrónico del coche

Para nuestro diseño se han tenido en cuenta varios factores que se dispondrán a continuación:

- La necesidad de utilizar dos fuentes de energía alternativas: Como se puede observar en el diseño actual, se ha utilizado varias pilas distintas, una recargable encargada de nutrir de energía el circuito principal, y otra auxiliar con más potencia para los motores. Esto es debido a que no son suficientes los 5V que nos proporciona la pila recargable para suministrarle toda la energía a nuestro circuito, ya que 2 motores externos necesitan una mayor potencia para poder moverse y que nuestro sistema funcione.

- Se van a utilizar dos sensores *sigue-líneas*, que serán los encargados de mantener el coche recto, esto es así debido a que los 2 motores que se usan en el circuito, al ser elementos mecánicos, no son completamente similares y pueden desviarse y no ir recto, así que utilizamos unas líneas en el carril que detectamos con los sensores *sigue-línea*s para saber que nuestro vehículo va por buen camino.

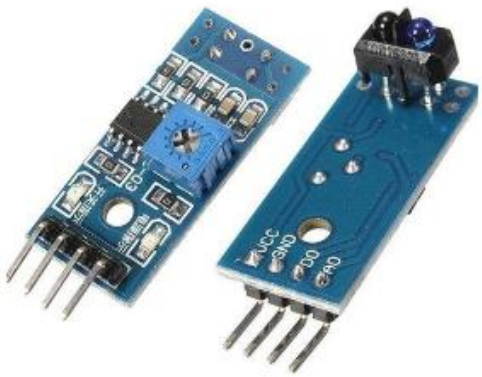


Ilustración 2.12.2: Sensores sigue-líneas(Fuente:[33])

- Motores, los cuales se van a encargar de controlar los movimientos de nuestro coche, como ir hacia delante o girar donde se le indique. Los motores vienen

con una reductora integrada en ellos y unas ruedas en sus extremos. También vamos a colocar una rueda auxiliar con lo que tener un tercer punto de apoyo y se mantenga firme en el suelo. Para conectar los motores vamos a usar la controladora L298N, la cual se va a usar para controlar la corriente que le llega a los motores, la velocidad y el sentido de la rotación de ambos.

Ilustración 2.12.3: Motores y ruedas(Fuente:[33])

Ilustración 2.12.4: Rueda loca(Fuente:[33])

- Un sensor de colores que se utiliza para que el coche pueda revisar en el entorno, este es crucial para que el vehículo se comunique con la aplicación y

sepa en todo momento si tiene que realizar un siguiente movimiento, si se ha salido del circuito o si ha llegado a la meta.

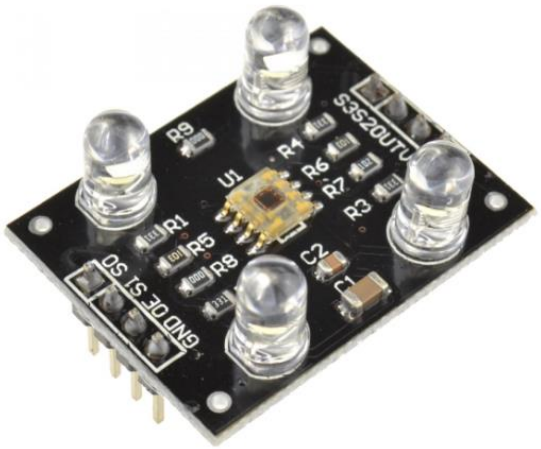


Ilustración 2.12.5: Sensor de color(Fuente:[33])

- Por último, tenemos la placa ESP32. La placa se compone de pines donde por cables conectaremos todos estos componentes y será el corazón de nuestro sistema, que albergará el código correspondiente para que todo funcione y del que hablaremos más adelante. Esta placa es clave ya que tiene integrado un sensor bluetooth con el que se hacen todas las conexiones con nuestra app móvil, donde podremos controlar todos los movimientos del coche y nos comunicaremos con la placa.

2.5.1.2 Diseño Software

Teniendo en cuenta toda la parte hardware que tenemos, se ha creado una solución software que satisfaga las necesidades de nuestro sistema. Para ello nuestro pseudocódigo debe seguir las siguientes pautas para que todo funcione tal y como se ha pensado.

1. Antes que nada, la conexión por bluetooth tiene que ser correcta para que nuestro sistema funcione correctamente.
2. El sistema se quedará a la espera de que llegue por bluetooth alguno de los comandos configurados que le indican el movimiento que tiene que realizar.

3. Si llega un movimiento de rotación a derecha o izquierda, el vehículo se saldrá del carril en el que esté y se posará en el carril que tenga 90 grados a la dirección que se le indique.
4. Si se le indica que debe ir hacia delante, se tienen que usar todos los sensores, tanto los *sigue-líneas* para evitar que se salga del carril como el sensor de color, para que el sistema sepa cuando tiene que parar el movimiento y sepa cuando se ha salido o ha llegado a la meta. Esto ejecutará un comando por bluetooth que le enviará la información a la aplicación móvil si se ha salido o ha llegado a la meta para que pare el movimiento.
5. También debe de tener funciones auxiliares para detectar los colores del entorno, para poder reconocer a posteriori cuáles son los colores y que tonalidades tiene que reconocer.

2.5.2 Análisis y diseño de la aplicación móvil

Para la aplicación hemos pensado en realizar un diseño simple e intuitivo, donde todos puedan entender los distintos movimientos y opciones que tenemos disponibles, a continuación, se muestran los distintos mockups con el diseño de la misma:

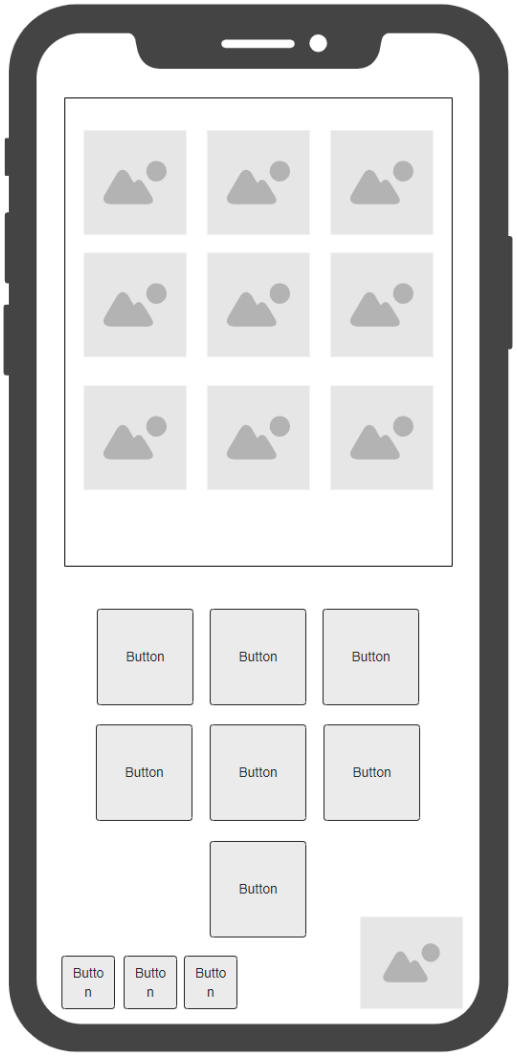


Ilustración 2.12.6: Mockup app móvil

Como se puede observar en la imagen, la interfaz de la aplicación móvil muestra una pantalla con siete botones principales y 3 secundarios, donde se encontrarán los distintos movimientos y algunos de los distintos modos que tenemos. La parte superior de nuestra aplicación es dominada por una lista de imágenes, que se irá actualizando con la imagen de los movimientos conforme vayan siendo pulsados por el usuario e irán desapareciendo conforme nuestro vehículo vaya realizando los movimientos.

A continuación, se va a presentar el diseño final que tendrá la aplicación con las imágenes que se utilizarán para cada opción y movimiento:

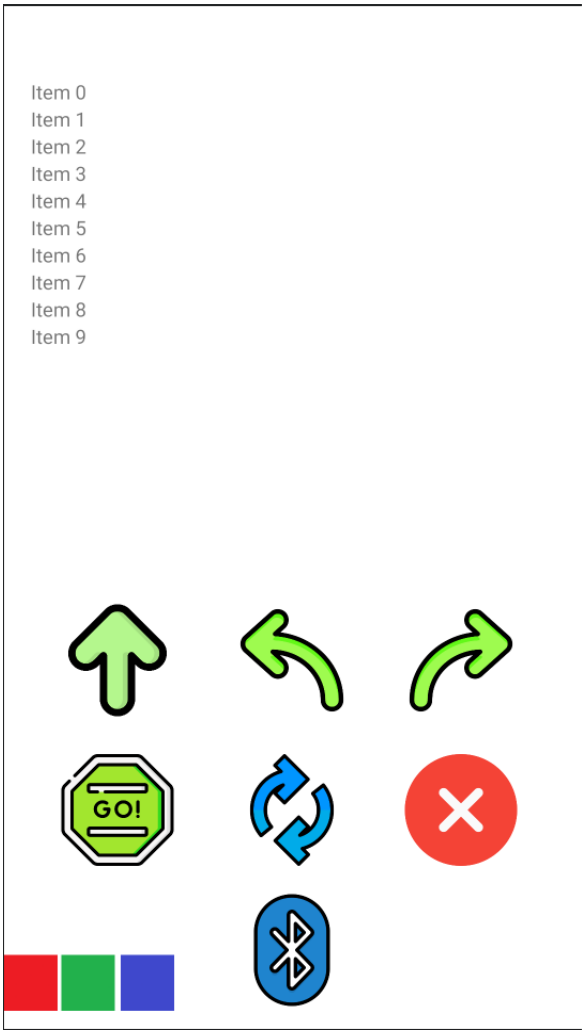


Ilustración 2.12.7: Diseño pantalla principal

Los movimientos son representados con forma de flechas que indican el movimiento que va a realizar nuestro sistema. Como se puede observar hay también un botón verde que indica que se va a realizar la secuencia de movimientos indicada y otro rojo que elimina la secuencia que se ha insertado y, en el caso de que ya esté en movimiento, para el movimiento de los motores de nuestro coche.

También se pueden ver 2 botones centrales, uno que activa el modo bucle, que indica a la aplicación que va a reenviar la secuencia hasta que nuestro sistema llegue

a su destino o se salga del camino asignado y un símbolo de bluetooth que sirve para desconectarse del sistema en caso de que hayamos terminado y dejar que este pueda realizar una conexión distinta.

Por último, tenemos 3 botones en la parte inferior derecha que se utilizarán para calibrar los colores que nuestro sistema tendrá que reconocer a la hora de realizar el circuito, estos colores pertenecen a los mismos colores usados en nuestro circuito y cada uno corresponde al color que se va a analizar del entorno.

2.5.3 Análisis y diseño del tablero de control

Viendo los demás apartados y sabiendo cuales son los y que necesita para que nuestro sistema detecte el entorno y pueda reaccionar ante él, se ha considerado realizar un tablero interactivo y modular, para ello se han realizado los siguientes diseños de las piezas que se montarán en el tablero:

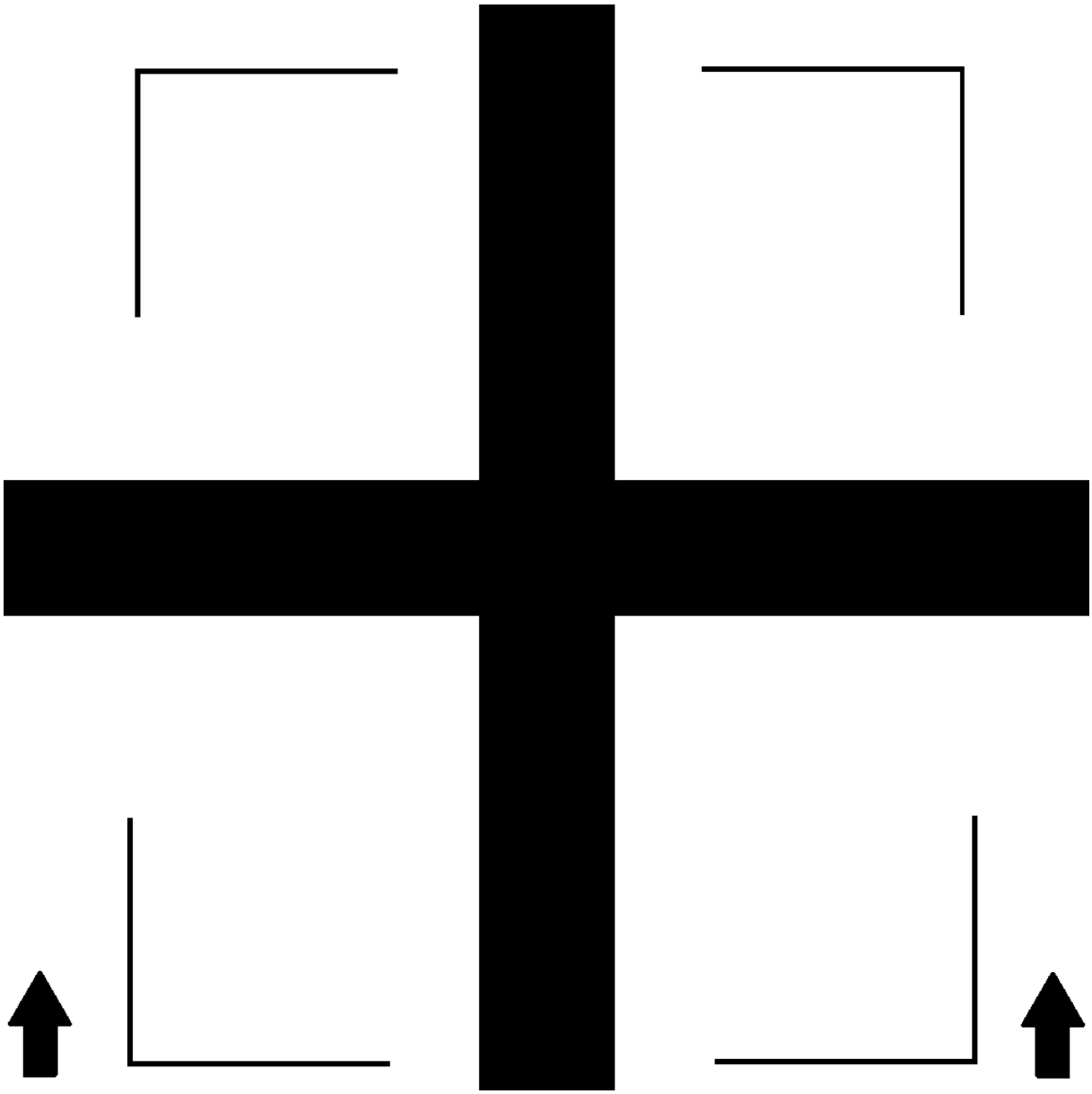


Ilustración 2.12.8: Diseño casilla de salida

Esta será la casilla de salida, en donde se posicionará el coche a la hora de empezar el circuito, es por eso que se han realizado unas marcas para saber la posición inicial en la que se va a comenzar.

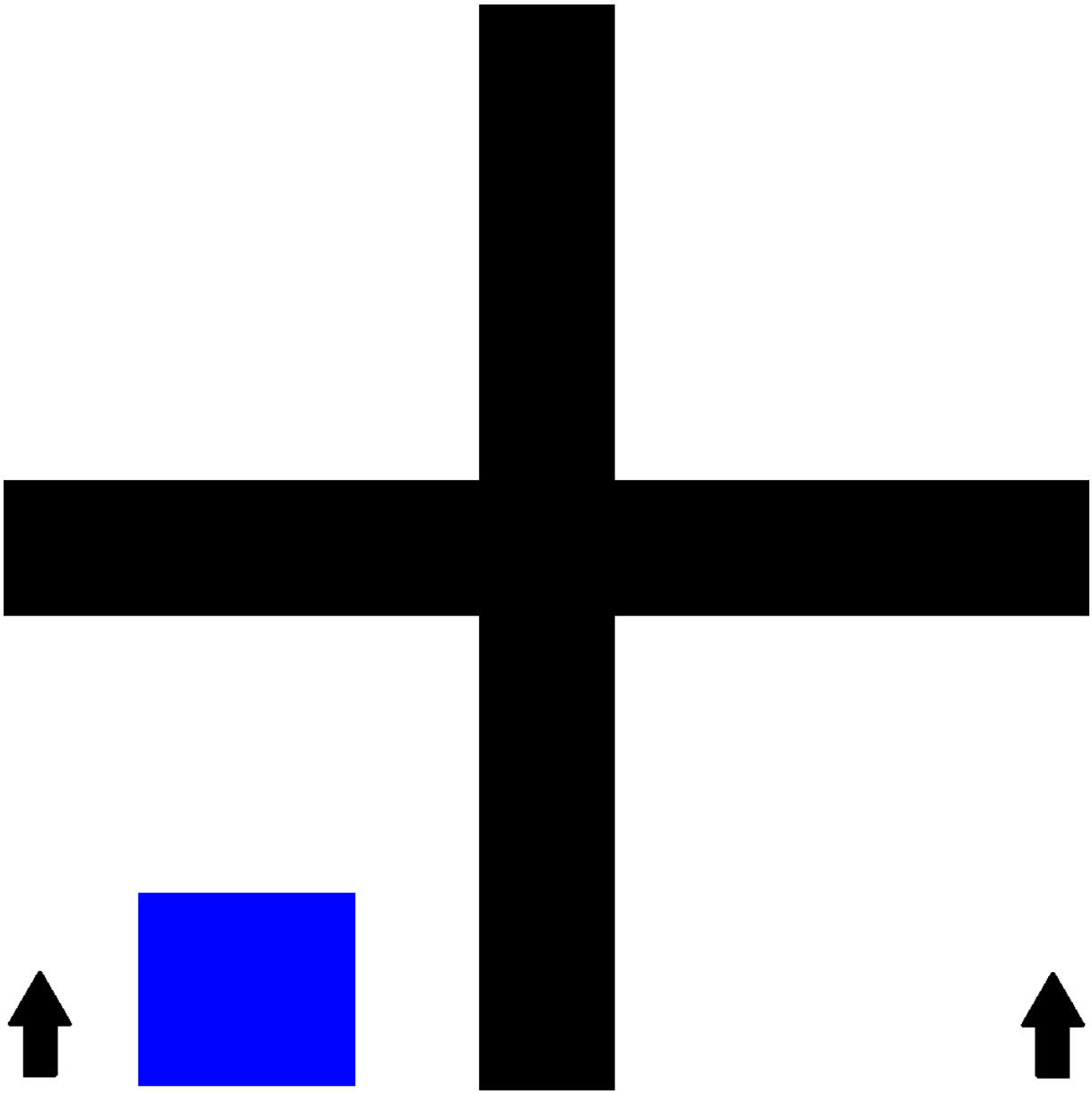


Ilustración 2.12.9: Diseño casilla Movimiento

Esta es una casilla de movimiento, esta casilla interconecta los carriles entre sí y es por donde nuestro coche tendrá que desplazarse hasta llegar a la meta. Como se puede ver en la imagen, hay una marca azul a la izquierda que indicará al vehículo los puntos estratégicos en los que tiene que pararse para realizar el siguiente movimiento.

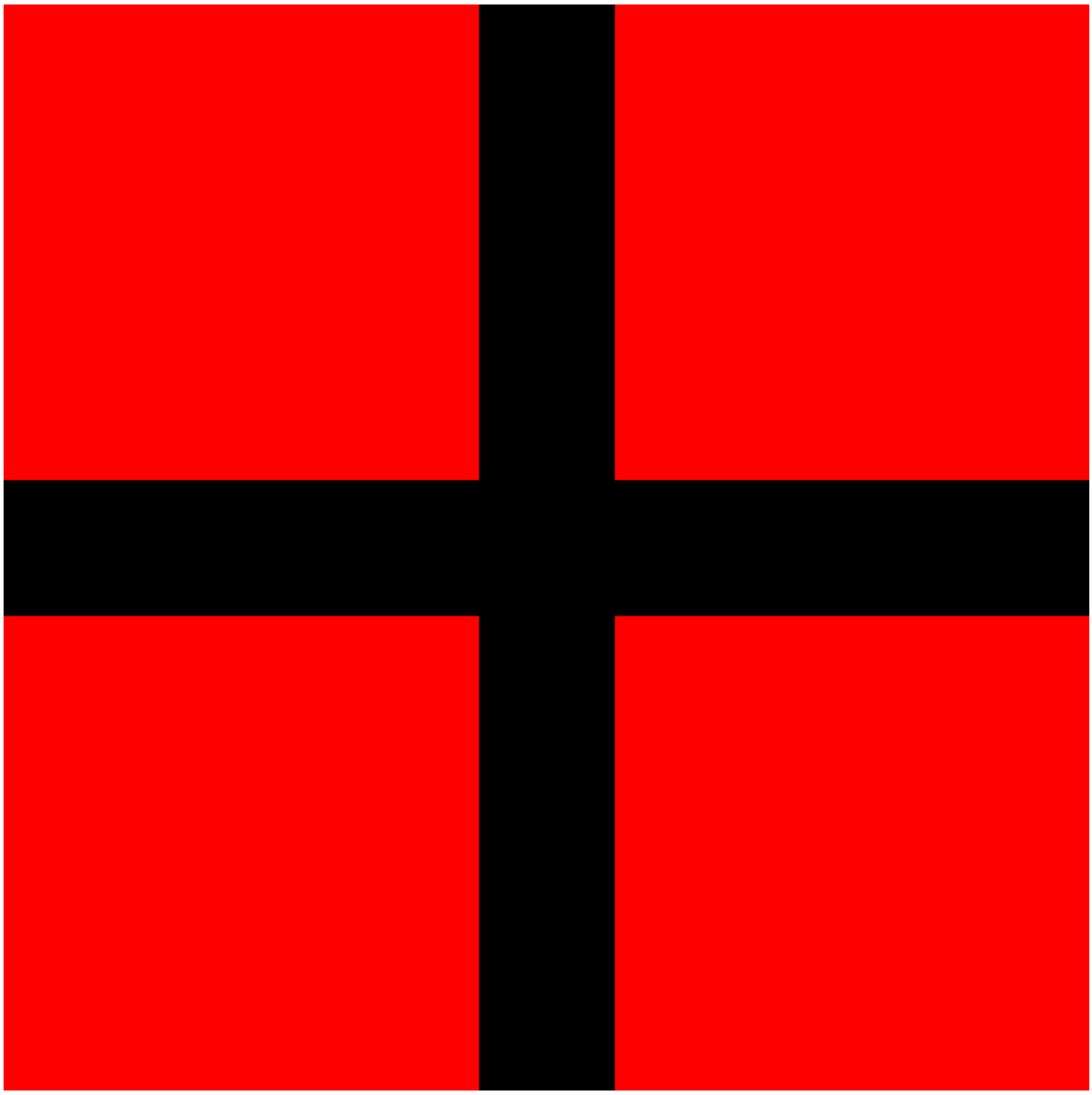


Ilustración 2.10: Diseño casilla Fuera

Estas casillas son la parte externa al camino, y gracias a su color rojo característico el sistema será capaz de reconocer que está posicionado fuera de pista y le enviará una señal a nuestra aplicación para parar el recorrido restante.

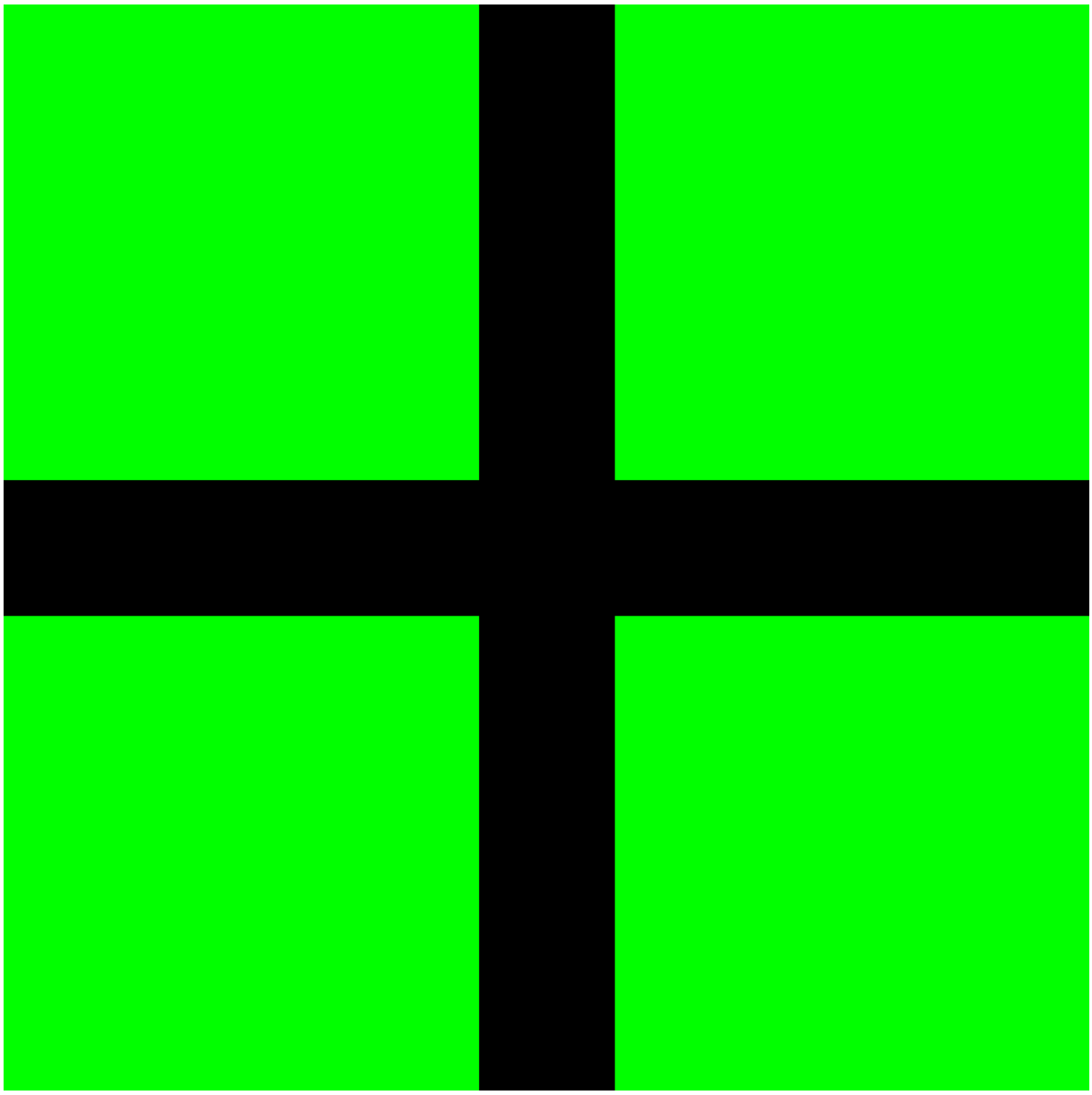


Ilustración 2.11: Diseño casilla Fin

Por último, tenemos la casilla que indica que el recorrido se ha completado con éxito, gracias a su color verde, nuestro sistema será capaz de reconocerlo y enviar una señal a la aplicación para indicarle que el circuito ha sido completado con éxito.

3 DESARROLLO

Para el desarrollo del proyecto hemos seguido una metodología de desarrollo ágil de 4 iteraciones, dividiendo los desarrollos en el montaje del vehículo, creación del software, montaje del entorno y, por último, una mejora de la interfaz en la aplicación móvil para que los niños puedan entender mejor las instrucciones que utilizan y sea un entorno muy visual con el que aprender a programar jugando.

3.1 Primera Iteración

En esta primera iteración del proyecto, nos enfocaremos en el montaje del vehículo controlado por nuestra placa ESP32. Seleccionaremos los componentes necesarios para el chasis, las ruedas, motores y sensores. Luego, procederemos con el ensamblaje físico del vehículo. Utilizaremos un microcontrolador L298N para controlar los motores y capturar datos de los sensores. El objetivo principal es lograr que el vehículo se mueva según las instrucciones básicas de control.

3.1.1 Tareas

Las tareas a realizar para esta iteración son las siguientes:

1. Seleccionar y adquirir los componentes necesarios: chasis, ruedas, motores, sensores y todo lo necesario. Será necesario el uso del cableado para poder conectar todos los componentes del sistema entre sí.
2. Ensamblar el chasis y montar las ruedas y motores de acuerdo con el diseño planificado.
3. Conectar los sensores y motores al microcontrolador ESP32.
4. Programar el ESP32 para recibir comandos de control básicos (adelante, atrás, izquierda, derecha) y para responder a los datos de los sensores.

5. Realizar pruebas para asegurarse de que el vehículo se mueva según lo esperado y responda a los obstáculos.

3.1.2 Desarrollo de la iteración

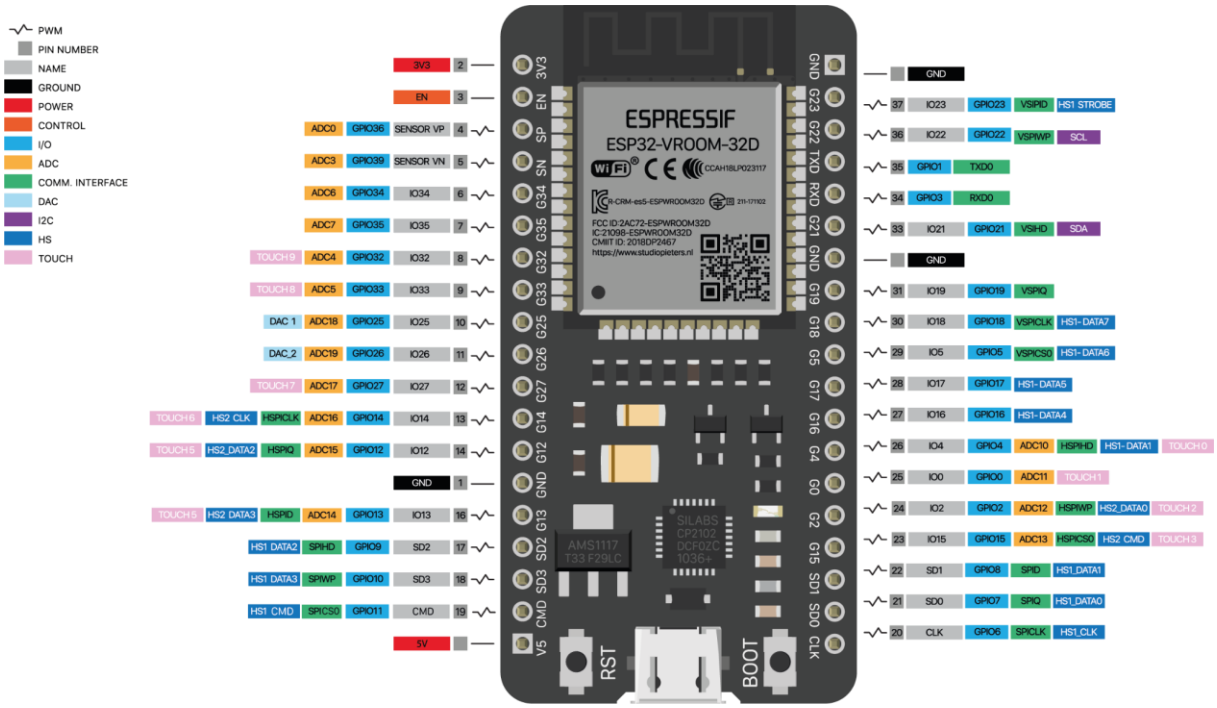


Ilustración 3.1: Pinout ESP32(Fuente:[27])

En el siguiente diagrama se puede observar los distintos tipos de pines que son necesario conocer para poder realizar nuestro proyecto, ya que tenemos que saber cómo conectar y donde los distintos sensores del sistema.

A continuación, se va a mostrar un esquema de todos los pines que se han utilizado para realizar el proyecto, así como el destino de los sensores en los que van conectados.

Para los controlar el giro de los motores vamos a utilizar los pines 16, 17, 18 y 19. Para su velocidad se utilizarán los pines 22 y 23 de forma analógica para poder

tener un mayor control, ya que, a diferencia del giro, la velocidad tiene que poder ser regulada, ya que si no el coche podría ir lo suficientemente rápido como para salirse del carril o no detectar los colores del suelo. La velocidad utilizada será de un valor de 80 frente a los 255 de máximo que tenemos, lo que implica aproximadamente un 30% de su velocidad.

Para los sensores *sigue-líneas* se van a necesitar dos pines de entrada únicamente, uno por sensor. Se usan el 34 y 35 y cada uno indicará un 0 si está posicionado encima de la línea o un 1 si no lo está.

Para el sensor de color se necesitan más pines, en este caso vamos a usar los pines siguientes:

s0 = 27

s1 = 26

s2 = 25

s3 = 33

out = 32

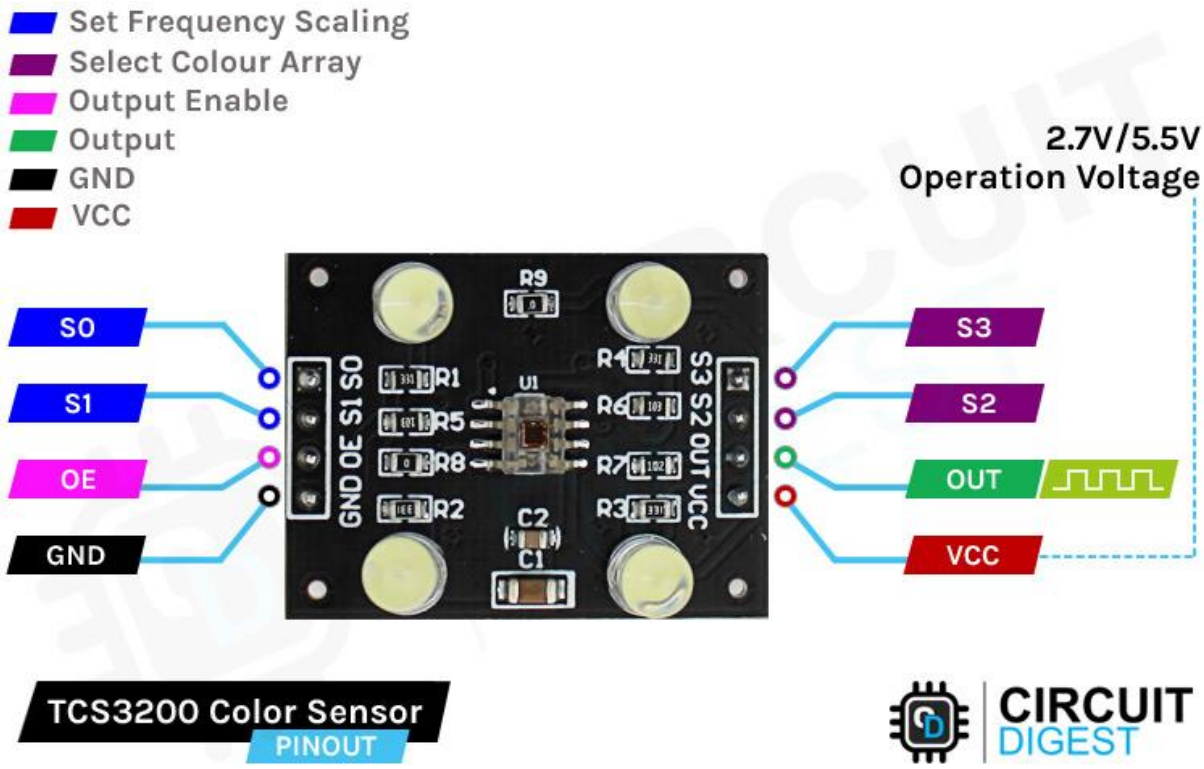


Ilustración 3.2: Pinout sensor de colores(Fuente:[28])

Los pines S0 y S1 son de salida y se utilizan para indicar una frecuencia de muestreo al sensor, en este caso vamos a usar un escalado del 100%, que es indicado cuando tanto s0 como s1 están en valor 1 o también conocido como Alto.

Los pines S2 y S3 se utilizan para indicarle qué filtro queremos utilizar a la hora de detectar la intensidad de la luz que entra. De este modo y usando los 3 tipos de filtro es posible detectar el color que tiene delante y es lo que se usará para detectar las distintas marcas de colores en la pista. La intensidad de la luz se marcará desde el pin de entrada analógico Out, indicando 0 en el caso de que llegue una cantidad de luminosidad total o 255 en el caso de que no se registre ningún tipo de luz.

S2 = 0, S3 = 0 → Filter Red
S2 = 0, S3 = 1 → Filter Blue
S2 = 1, S3= 0 → No Filter
S2 = 1, S3= 1 → Filter Green

Por último, nos queda conocer los pines que utilizaremos para dar alimentación a nuestros componentes, estos son los contactos de los extremos 3V3 y GND, que hacen referencia a una salida de 3.3 voltios y otra a tierra respectivamente.

Todos los componentes que utilizamos necesitan estar conectados a la alimentación a través de estos pines, y debido a la falta de estos en la placa se ha optado por usar la parte de una placa de pruebas para realizar la conexión de todos ellos:

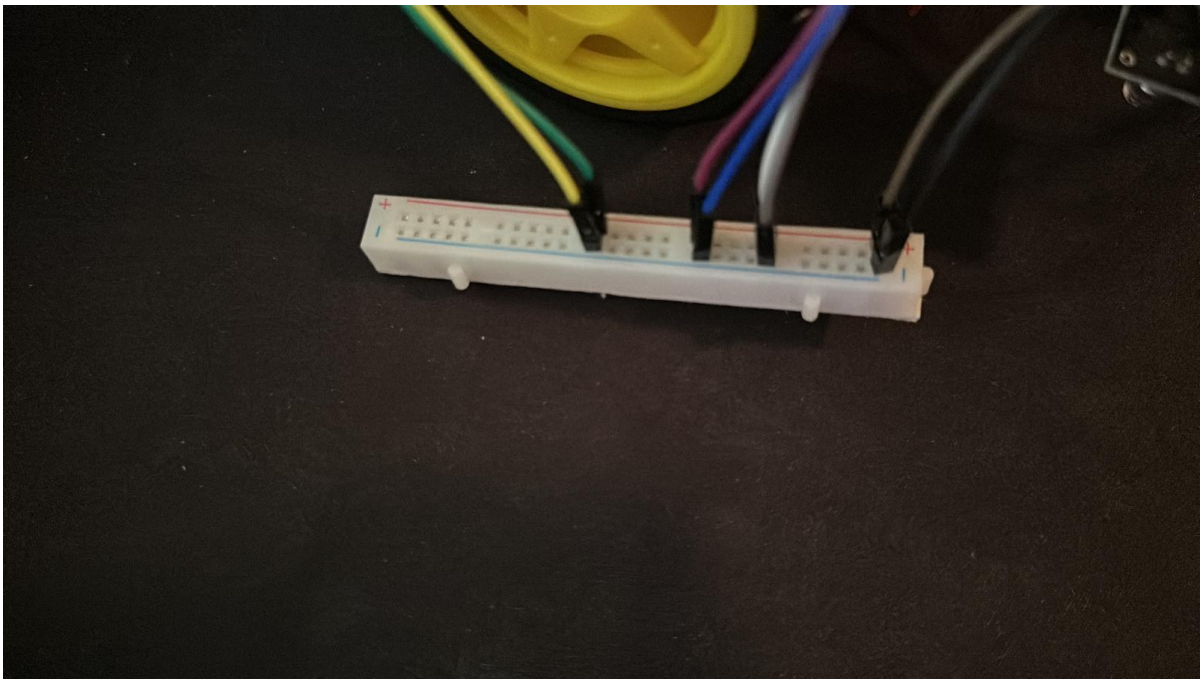


Ilustración 3.3: Protoboard de alimentación

Una vez sabemos cómo va a ir cada pin conectado, se va a proceder a realizar el montaje. Para ello vamos a recurrir al diseño previo que se hizo y, se va a montar.

Luego se ha implementado un código simple para ver si funciona bien y es capaz de realizar los movimientos.

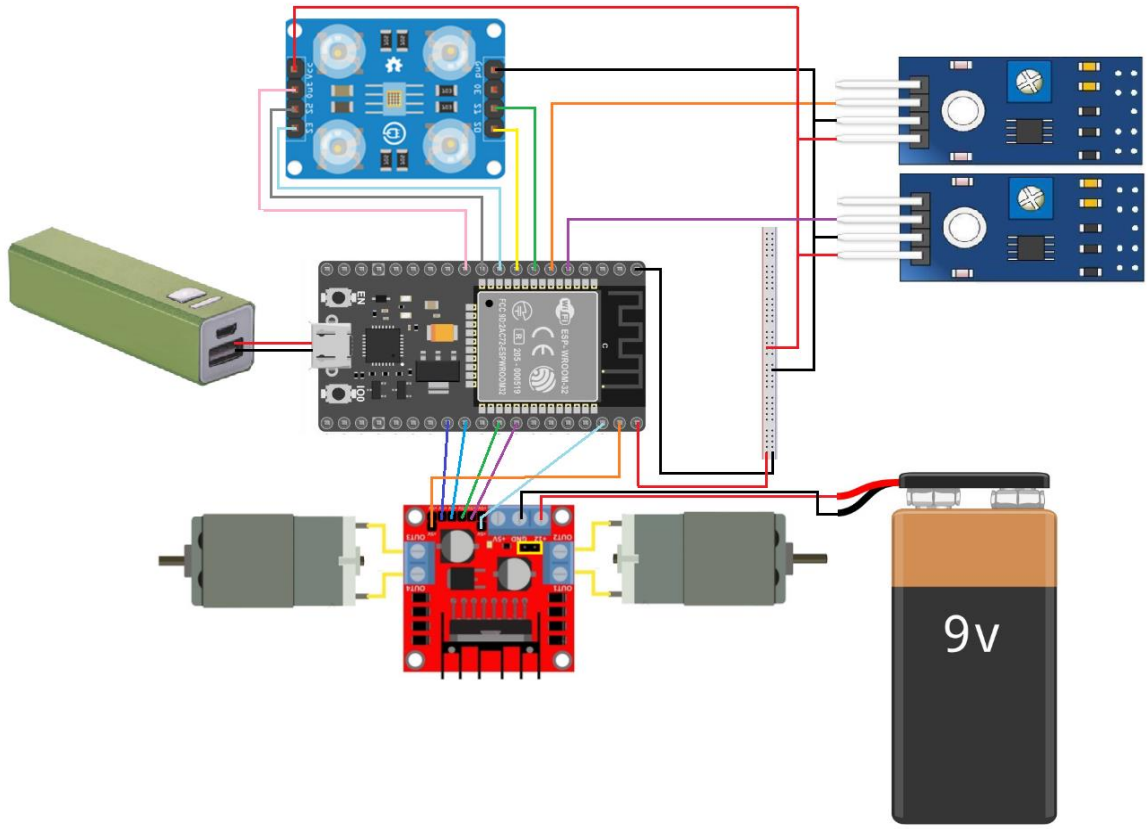


Ilustración 3.4: Diseño electrónico coche

Una vez todo montado el resultado es el siguiente:

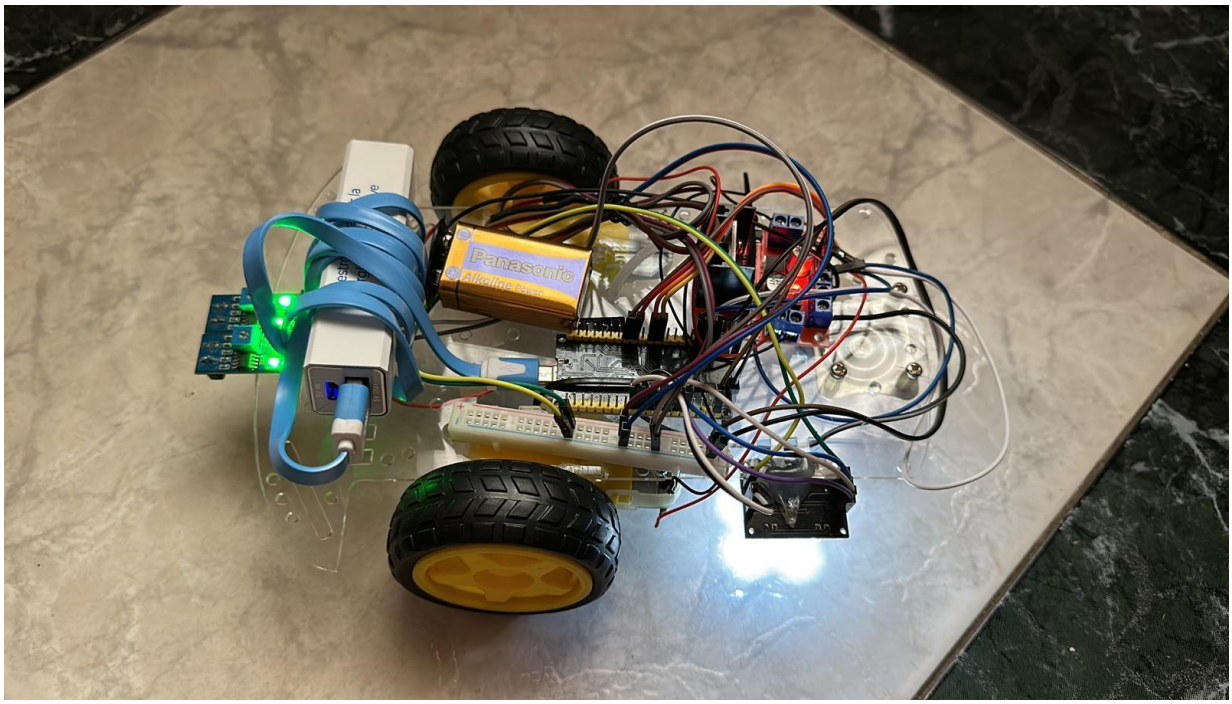


Ilustración 3.5: Prototipo construido

3.1.3 Pruebas

Para realizar las pruebas finales se ha utilizado el siguiente código de prueba y la siguiente aplicación del Store de Google. La aplicación permite comunicarse con dispositivos bluetooth y mandarle los caracteres necesarios, es una app genérica que se ha utilizado solo para las pruebas iniciales.

```ino
#include <SoftwareSerial.h>
SoftwareSerial BT(4,2); // RX, TX recorder que se cruzan
// Motor 1
int ENA = 10;
int IN1 = 9;
int IN2 = 8;
// Motor 2
int ENB = 5;
int IN3 = 7;
int IN4 = 6;
void setup() {
 Serial.begin(9600);
 BT.begin(9600);
 pinMode (ENA, OUTPUT);
 pinMode (ENB, OUTPUT);
 pinMode (IN1, OUTPUT);
 pinMode (IN2, OUTPUT);
 pinMode (IN3, OUTPUT);
 pinMode (IN4, OUTPUT);
}
void loop() {
  if (BT.available()){
    char dato=BT.read();
    Serial.println(dato);
    switch(dato){
      case 'a':
        Adelante();
        break;
      case 'r':
        Atras();
        break;
      case 'd':
        Derecha();
        break;
      case 'i':
        Izquierda();
        break;
      case 'p':
        Parar();
        break;
    }
  }
}
void Adelante (){
 //Dirección motor A
 Serial.println("Entra a perfe");
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, HIGH);
 analogWrite (ENA, 255); //Velocidad motor A
 //Dirección motor B
 digitalWrite (IN3, HIGH);
 digitalWrite (IN4, LOW);
 analogWrite (ENB, 255); //Velocidad motor B
}
void Derecha (){
 //Dirección motor A
 Serial.println("Entra b perfe");
 digitalWrite (IN1, HIGH);
 digitalWrite (IN2, LOW);
 analogWrite (ENA, 255); //Velocidad motor A
 //Dirección motor B
 digitalWrite (IN3, HIGH);
 digitalWrite (IN4, LOW);
 analogWrite (ENB, 255); //Velocidad motor A
}
void Izquierda (){
 //Dirección motor A
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, HIGH);
 analogWrite (ENA, 255); //Velocidad motor A
 //Dirección motor B
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, HIGH);
 analogWrite (ENB, 255); //Velocidad motor A
}
void Atras (){
 //Dirección motor A
 digitalWrite (IN1, HIGH);
 digitalWrite (IN2, LOW);
 analogWrite (ENA, 255); //Velocidad motor A
 //Dirección motor B
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, HIGH);
 analogWrite (ENB, 255); //Velocidad motor B
}
void Parar (){
 //Dirección motor A
 digitalWrite (IN1, LOW);
 digitalWrite (IN2, LOW);
 analogWrite (ENA, 0); //Velocidad motor A
 //Dirección motor B
 digitalWrite (IN3, LOW);
 digitalWrite (IN4, LOW);
 analogWrite (ENB, 0); //Velocidad motor A
}
```

Ilustración 3.6: Código de pruebas

Ilustración 3.7: App de pruebas

Todas las pruebas han salido de forma satisfactoria, por tanto, podemos proceder a comenzar con la siguiente iteración.

3.2 Segunda Iteración

En esta segunda iteración, nos centraremos en desarrollar el software necesario para el vehículo y una aplicación móvil que permite controlarlo de manera remota. El software del vehículo se comunicará con el microcontrolador Arduino para ejecutar comandos de movimiento y recopilar datos de los sensores. La aplicación móvil permitirá a los usuarios controlar el vehículo a través de una interfaz intuitiva.

3.2.1 Tareas

Las tareas que vamos a realizar en este apartado serán las siguientes:

1. Desarrollar el software embebido para el microcontrolador ESP32. Esto incluirá la lógica para interpretar comandos de movimiento y procesar datos de los sensores.
2. Establecer la comunicación entre la aplicación móvil y el vehículo. Esto será a través de Bluetooth.
3. Diseñar la interfaz de la aplicación móvil para controlar el vehículo. Deberá incluir controles para el movimiento, modo bucle y botones para reconocer los colores del terreno.
4. Implementar la funcionalidad del modo bucle por si se requiere el uso de ciertos movimientos repetitivos, esto mejora el pensamiento lógico
5. Realizar pruebas exhaustivas para asegurarse de que la comunicación entre el vehículo y la aplicación móvil sea estable y que el control sea preciso.

3.2.2 Desarrollo de la iteración

Para comenzar a desarrollar el software del microcontrolador vamos a partir de alguno de los códigos que utilizamos para realizar las pruebas de la iteración anterior. Vamos a seguir el siguiente pseudocódigo para el desarrollo:

Inicialización de variables y pines del ESP32

```
cont = 0; //Contador del número de bucles
void setup() {
        // Se inicializa todos los sensores del coche y bluetooth
}

Bucle() {
  if (Bluetooth disponible) {
    char dato = leemovimiento();
    switch(dato){
      case 'a':
        cont = 0; //reinicia el contador
        accion = true; //inicia el movimiento hacia delante
        while(accion){
          //se lee los valores del siguelineas
          Si los valores son correctos{
            //sigue recto
            Adelante();
```

```
    }
    else Si se sale por la izquierda{
      //gira derecha
      Derecha();
    }
    else Si se sale por la derecha{
      //gira izquierda
      Izquierda();
    }
    else no detecta linea{
      //Se salió del carril
      Parar();
      accion = false;
    }

    LeerColores()

    Si contador>50{
      si Detecta azul{
        Parar();
        accion = false;
      }
      else Detecta rojo{
        Parar();
        accion = false;
        //Envía un mensaje a la app para que muestre que se salió
      }
      else Detecta verde{
        Parar();
        accion = false;
        //Envía un mensaje a la app para que muestre que terminó bien
      }
    }
    cont++;
  }
  Parar();
  break;
case 'd':
  Derecha();
  accion = true;
  delay(100); //delay para que se salga del carril
  while(accion){
   si Vuelve a entrar en un carril{
      Parar();
      accion = false;
```

```
        }
       }
       break;
     case 'i':
       Izquierda();
       accion = true;
       delay(100); //delay para que se salga del carril
       while(accion){
         si vuelve a entrar en un carril{
           Parar();
           accion = false;
         }
       }
       break;
     case 'w':
       //Escanea el color Blanco
       break;
     case 'r':
       //Escanea el color Rojo
       break;
     case 'g':
       //Escanea el color Verde
       break;
     case 'b':
       //Escanea el color Azul
       break;
     case 'p':
       Parar();
       break;
    }
  }
  delay(20);
}

void Adelante (){
//Modifica los motores para que giren ambos hacia adelante
}
void Derecha (){
//Modifica los motores para que se realice un giro a la derecha
}
void Izquierda (){
//Modifica los motores para que se realice un giro a la izquierda
}
void Parar (){
//Para los motores
```

```
}

int getRojo(){
  //Escanea el color rojo
}

int getAzul(){
 //Escanea el color azul
}

int getVerde(){
 //Escanea el color verde
}
```

Simultáneamente se ha ido desarrollando el software de la aplicación móvil en Android Studio para poder hacer las pruebas de control con la app y el software del controlador del coche.

Aquí se puede observar el diagrama UML de la app.

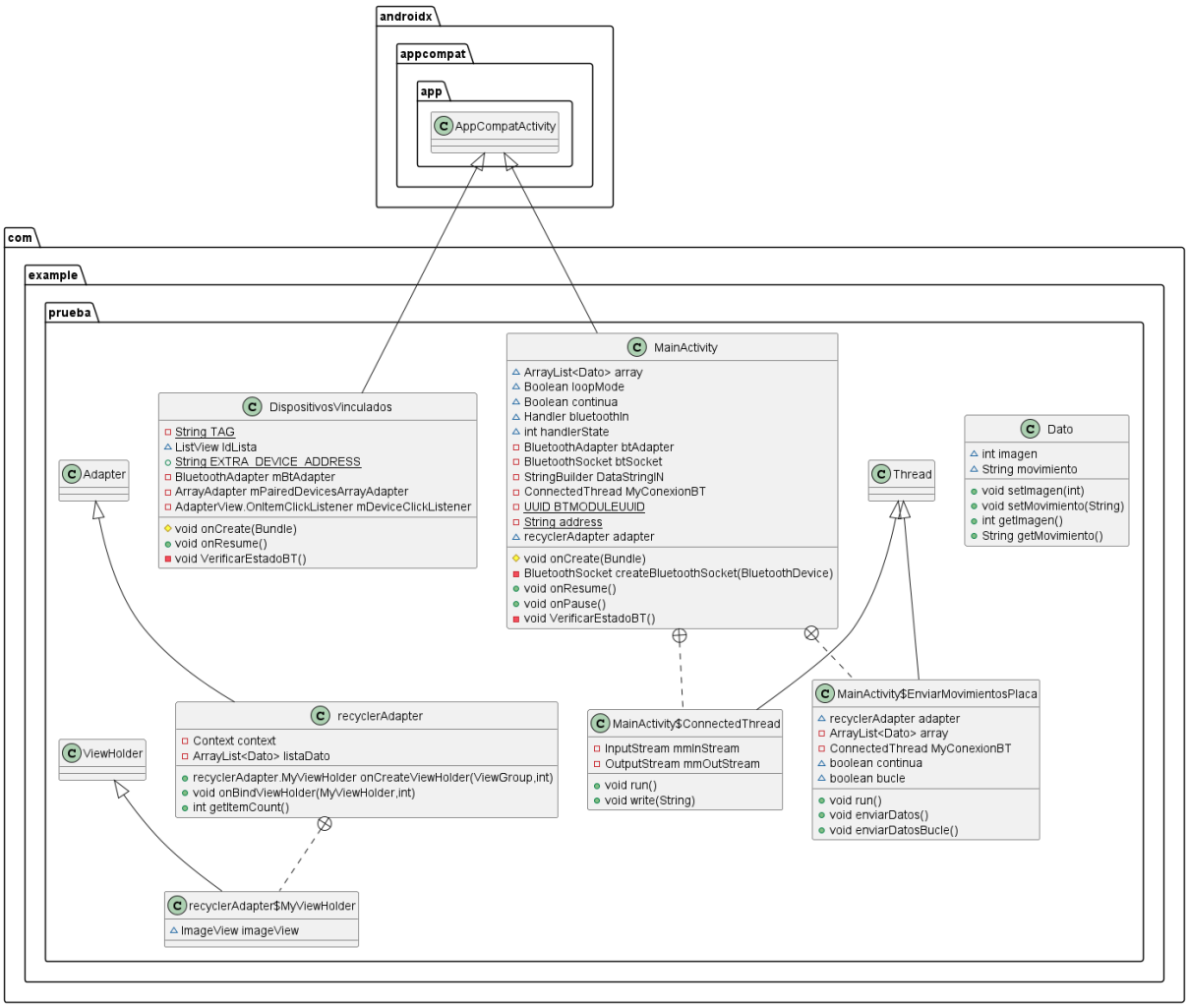


Ilustración 3.8: UML de la aplicación

La aplicación se basa en una pantalla inicial en donde se encuentran todos los dispositivos que se han vinculado a nuestro dispositivo móvil, en la cual habrá que elegir el dispositivo ESP32 que da vida a nuestro sistema.

Una vez pasada la pantalla inicial, tenemos la base de nuestra aplicación, el "*MainActivity*", aquí se encuentra todo lo relevante al movimiento del vehículo. Como podemos observar en el UML, esta clase es capaz de lanzar hilos dentro de ella, esto es debido a un error que se presentó a la hora del desarrollo, que provocaba que cada vez que se enviaba una secuencia de movimientos, al tener tiempos de espera y estar dentro de un bucle, se nos quedase la pantalla principal bloqueada y si intentábamos

cancelar cualquier acción se interrumpía el hilo principal y nos sacaba forzosamente de la app. Esto se ha solucionado creando un hilo cada vez que se crea una secuencia de movimientos, y si se cancela, simplemente se interrumpe el hilo y no se queda bloqueada la aplicación.

Nuestra lista de movimientos está creada por objetos de tipo dato, que se componen por el movimiento que se va a mandar por bluetooth al ESP32 y una imagen representativa del movimiento, que es la que podemos observar en los botones de movimientos y que, cuando estos son pulsados, aparecen en la lista de la parte superior de la app.

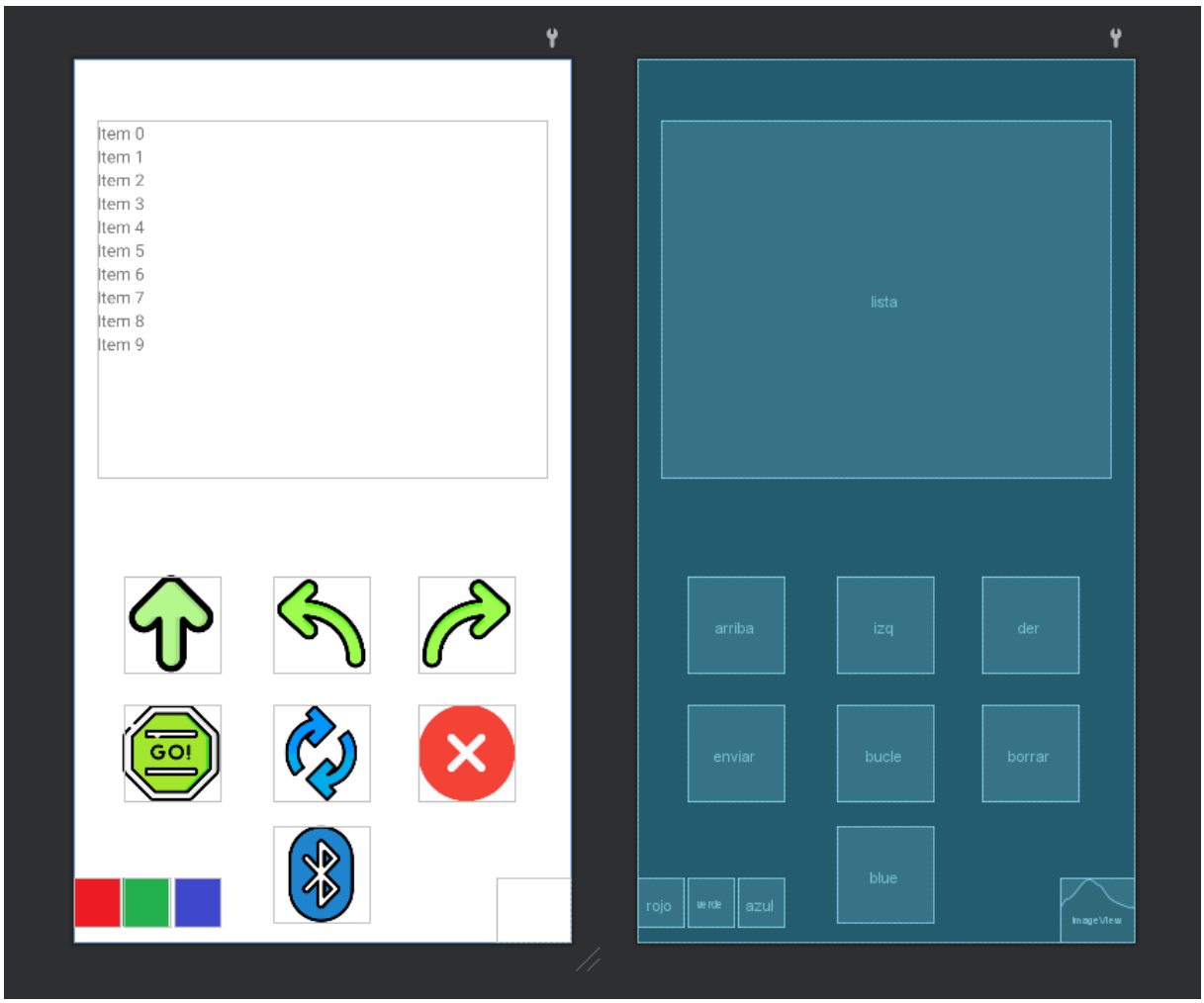


Ilustración 3.9: Diseño en Android Studio

El diseño final es completamente responsive, y, por tanto, se adapta a cualquier teléfono móvil sea cual sea su resolución. También se le han añadido mensajes cada vez que se pulsan ciertas teclas para informar al usuario que es lo que está sucediendo en todo momento. A continuación, en el apartado de pruebas vamos a ver cómo se comporta la app en un teléfono OnePlus 6 con una resolución 1920x1080p y que tal se adaptan los controles a la pantalla.

3.2.3 Pruebas

Las siguientes imágenes muestran todas las funcionalidades de la app y cómo se han probado.

Ilustración 3.10: Pantalla dispositivos vinculados

Este pantallazo pertenece a la pantalla inicial para seleccionar el dispositivo vinculado al que queremos conectarnos.

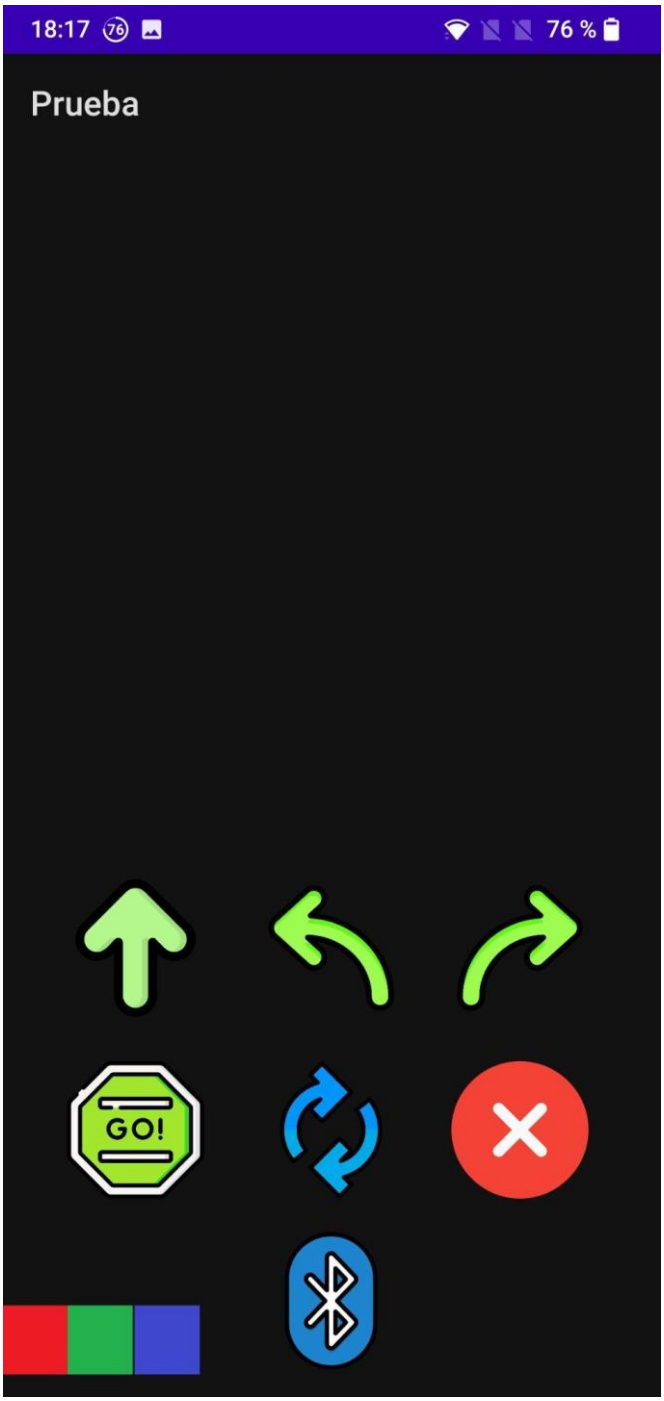


Ilustración 3.11: Pantalla principal de la app

En esta imagen se puede ver como la pantalla principal en la que se muestran los distintos botones que se pueden pulsar para realizar una secuencia de

movimientos. Cabe destacar que, al tener el sistema del móvil en modo oscuro, el fondo se oscurece automáticamente adaptándose al tema del teléfono.

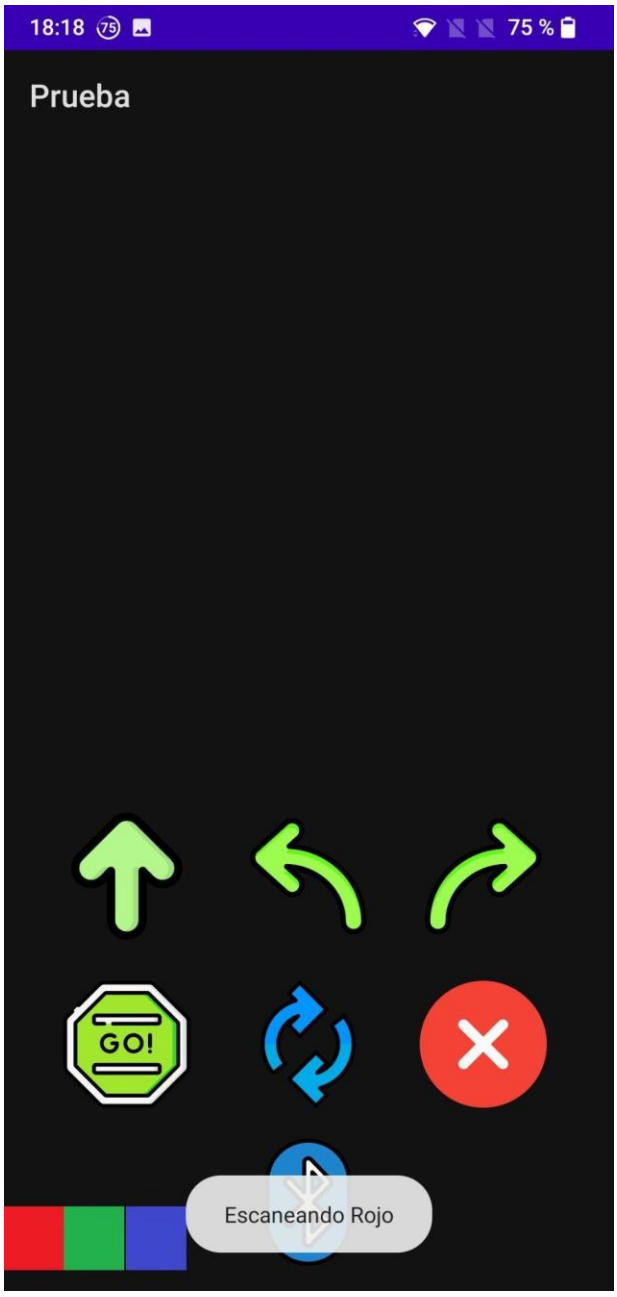


Ilustración 3.12: Pantalla escaneo de color

Los botones utilizados para escanear los colores funcionan perfectamente. Como se demuestra en la imagen, se acaba de escanear el color rojo y está preparado para que nuestro sistema pueda interactuar con él.

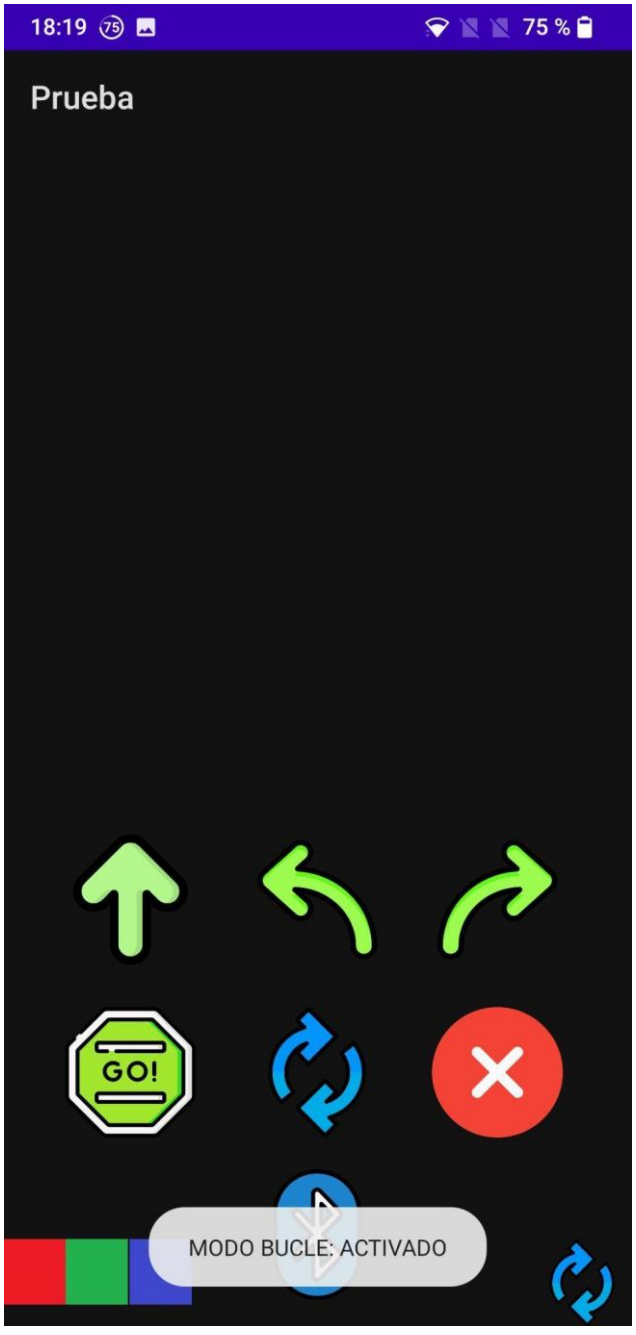


Ilustración 3.13: Pantalla modo bucle

Se ha probado también a activar el modo bucle, que funciona correctamente y no solo nos enseña un mensaje en el que podemos ver que está activo, si no que en la esquina inferior derecha podemos observar una imagen que nos indica también que la secuencia de movimientos que tenemos se va a procesar en modo bucle. Esto nos da un feedback visual a la hora en la que desaparezca el mensaje en pantalla y no sepamos si realmente está o no el modo activado.

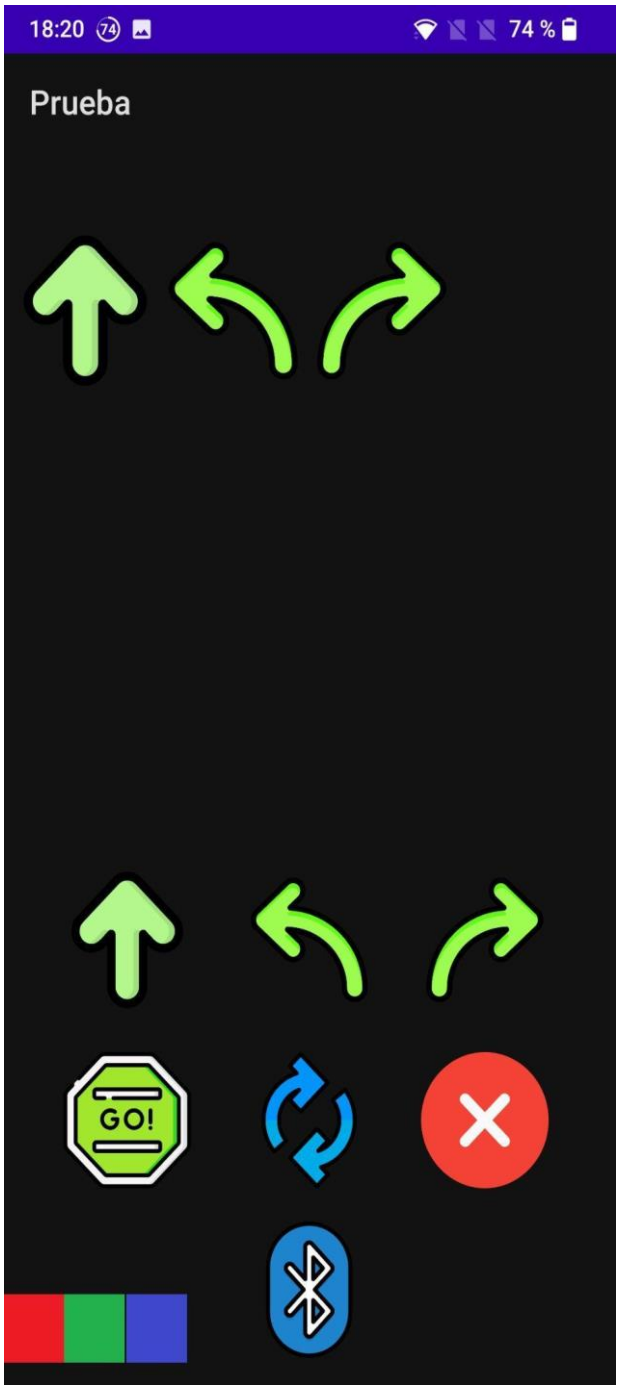


Ilustración 3.14: Pantalla lista de movimientos

Se puede observar también que los botones de movimiento funcionan correctamente, conforme vamos pulsando estos botones, van apareciendo en la lista de arriba las secuencia que acabamos de pulsar, también el botón de borrado funciona perfectamente, ya que si lo pulsamos la secuencia de arriba se borra.

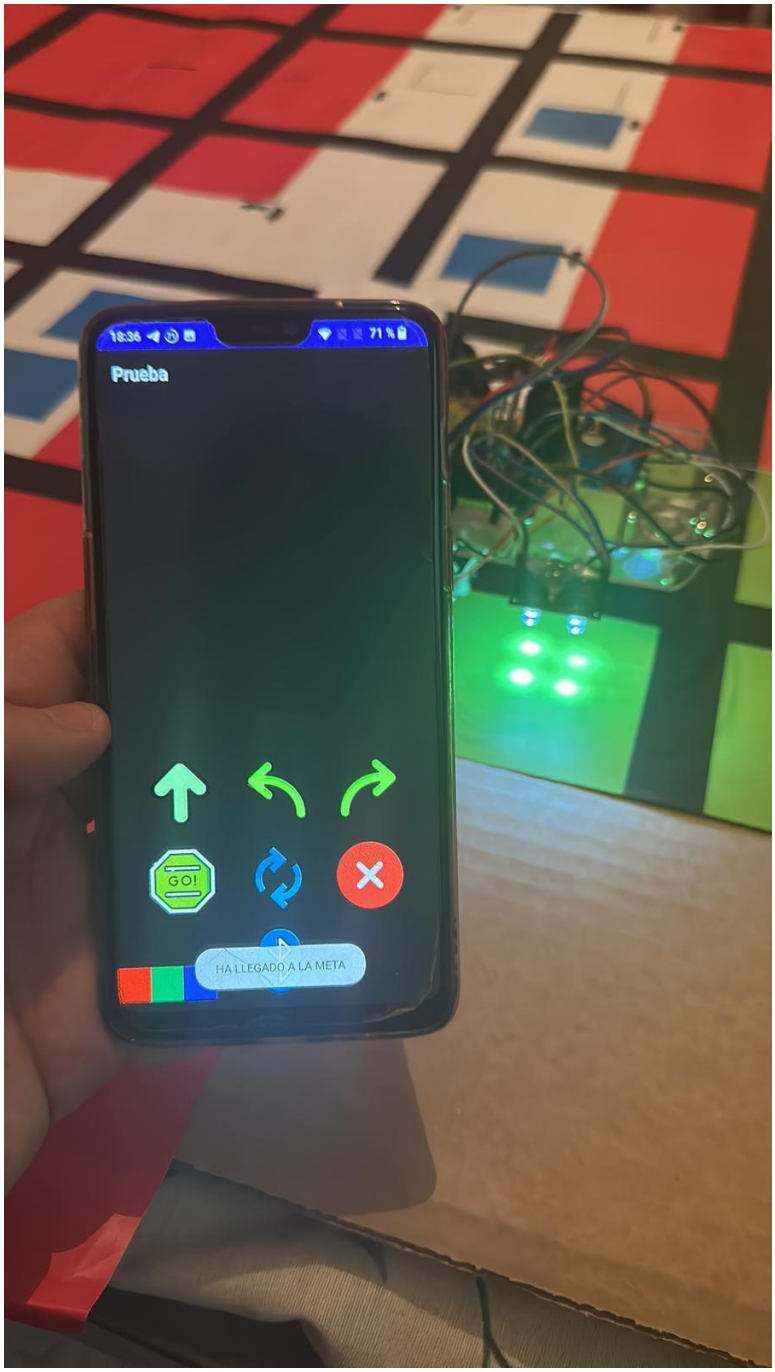


Ilustración 3.15: Prueba de conectividad

Por último, se ha probado que la conectividad con el coche y el sensor de color se realiza correctamente. Si se detecta el color verde previamente escaneado, este manda una señal a nuestra aplicación que nos muestra un mensaje en pantalla que

indica que el vehículo ha llegado a la meta. Si el color que detecta es el rojo, el mensaje que sale es que se ha salido del circuito. Hay que tener en cuenta que el sensor no es perfecto, y si hay un cambio en la luminosidad del ambiente o hay excesiva luminosidad en la estancia es posible que no pueda reconocer el color y tengamos que volver a escanearlo de nuevo o cambiar de instancia de pruebas. Pero generalmente si no se modifica el entorno y la luminosidad es la normal funciona satisfactoriamente.

Como hemos podido observar en las imágenes y en la explicación, las pruebas se han completado satisfactoriamente y la aplicación se comunica perfectamente con los componentes del coche. Por último, vamos a pasar a la tercera iteración, en donde montaremos un circuito completo en el que será el entorno donde probaremos más a fondo el coche y todas las características del mismo.

3.3 Tercera Iteración

En esta tercera iteración, nos enfocaremos en crear un entorno controlado en el cual el vehículo pueda operar y realizar su circuito al completo. Este entorno se creará a partir de los diseños realizados previamente. El diseño mediante piezas separadas permite realizar diferentes circuitos. Para las pruebas, se fotocopian las piezas y han pegado en una superficie de cartón robusta.

3.3.1 Tareas

Las tareas que se van a realizar en esta iteración son las siguientes:

1. Diseñar y crear el entorno físico en el cual el vehículo realizará los movimientos que se indiquen en la aplicación.

2. Integrar marcas o referencias visuales en el entorno que permitan al vehículo realizar un seguimiento preciso de su ubicación, en este caso se utilizarán colores.
3. Fotocopiar y pegar los diseños de las casillas de las que estará compuesto nuestro circuito.
4. Realizar pruebas en el entorno para asegurarse de que el vehículo pueda operar de manera confiable y realizar las tareas previstas de manera efectiva.

3.3.2 Desarrollo de la iteración

En primer lugar, partimos de la necesidad de utilizar un entorno firme donde nuestro coche pueda realizar los movimientos sin tener que preocuparse por ningún tipo de obstáculo ajeno al proyecto. Para ello hemos considerado utilizar un cartón amplio donde se pegarán las casillas de las que estará compuesto nuestro circuito. En este proyecto la idea a futuro sería de utilizar casillas impresas en 3D que puedan ser colocadas al gusto del usuario y retiradas con facilidad una vez se quiera modificar el circuito, pero como estamos realizando un prototipo, se va a realizar como anteriormente se ha mencionado.

Ilustración 3.16: Preparación del entorno

Lo primero que hemos hecho ha sido preparar un cartón firme donde realizar las pruebas y fotocopiar los diseños previamente mostrados, acto seguido se realizará un camino en donde iremos pegando cada casilla de movimiento.

Ilustración 3.17: Diseño del camino

Como muestra la imagen, se ha pegado un camino cualquiera que va a ser el que vamos a utilizar para las pruebas, también se ha pegado la casilla de meta al final del mismo para tenerlo todo listo. Acto seguido vamos a proceder a pegar las casillas externas que indicarán a nuestro coche que se ha salido de la pista en el caso de que se posicione encima de estas.

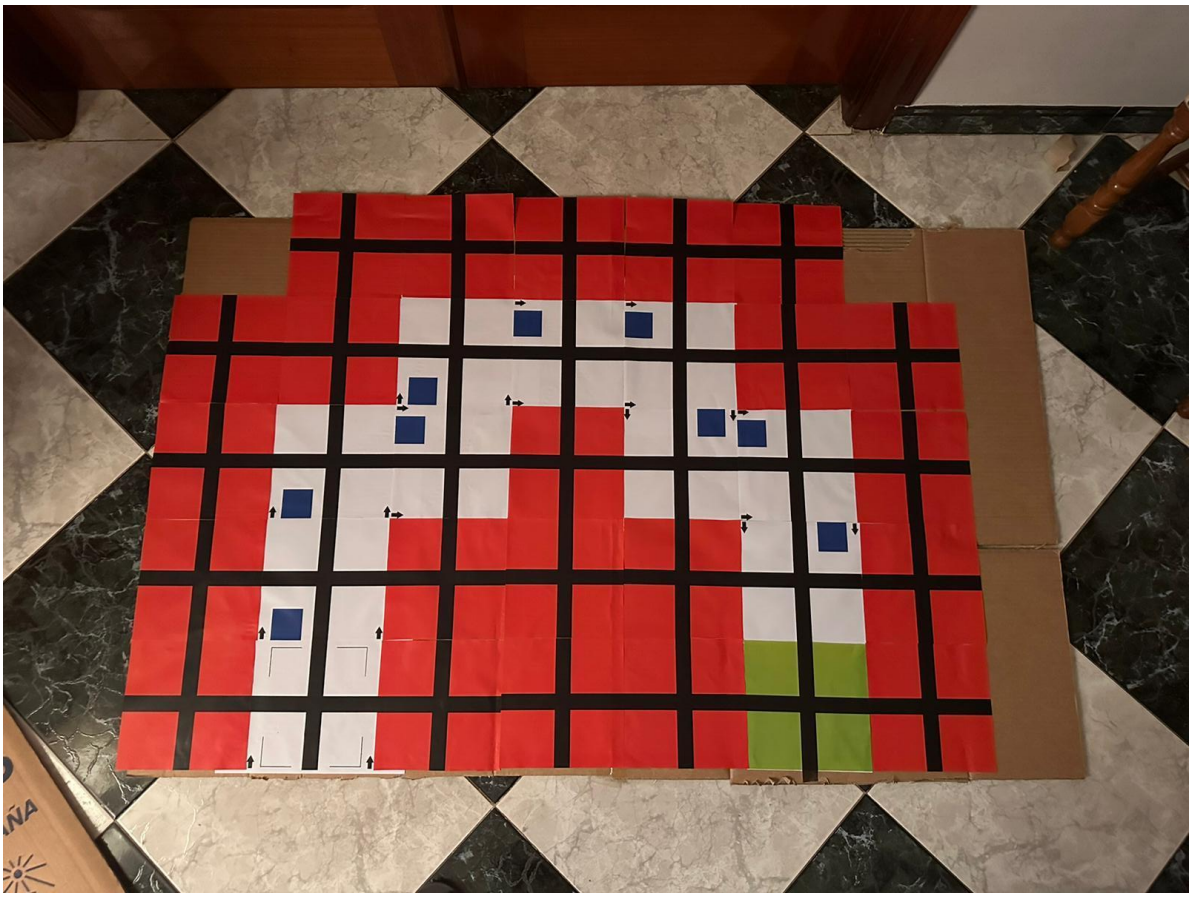


Ilustración 3.18: Circuito creado

Como se puede ver en la imagen, ya está todo listo para que nuestro coche realice el circuito. Por tanto, vamos a proceder a las pruebas finales del sistema y ver si es capaz de pasar el circuito sin muchos problemas

3.4 Pruebas finales

Por último, se ha realizado como prueba final del circuito, usando el entorno creado en la iteración 3. A continuación se va a explicar con todo detalle cómo se han realizado las pruebas y si han resultado satisfactorias o no.

En primer lugar, se ha conectado la aplicación móvil con el sistema se ha procedido a escanear los 3 colores necesarios para continuar con las pruebas, el rojo

para detectar cuando se sale fuera de pista, el verde para detectar si ha llegado a su destino, y por último el azul, para detectar cuando se ha de pasar de un movimiento al siguiente.

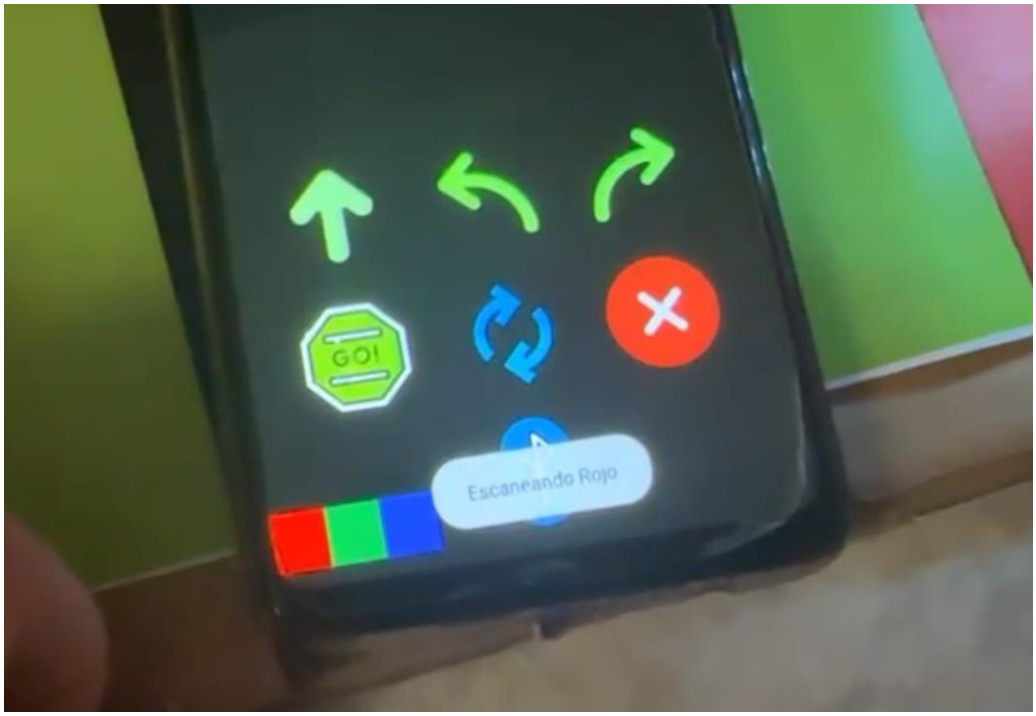


Ilustración 3.19: Escaneo inicial de prueba

Acto seguido se ha realizado el primer circuito, en el que se ha descrito la secuencia apropiada para que el coche llegue a la meta.

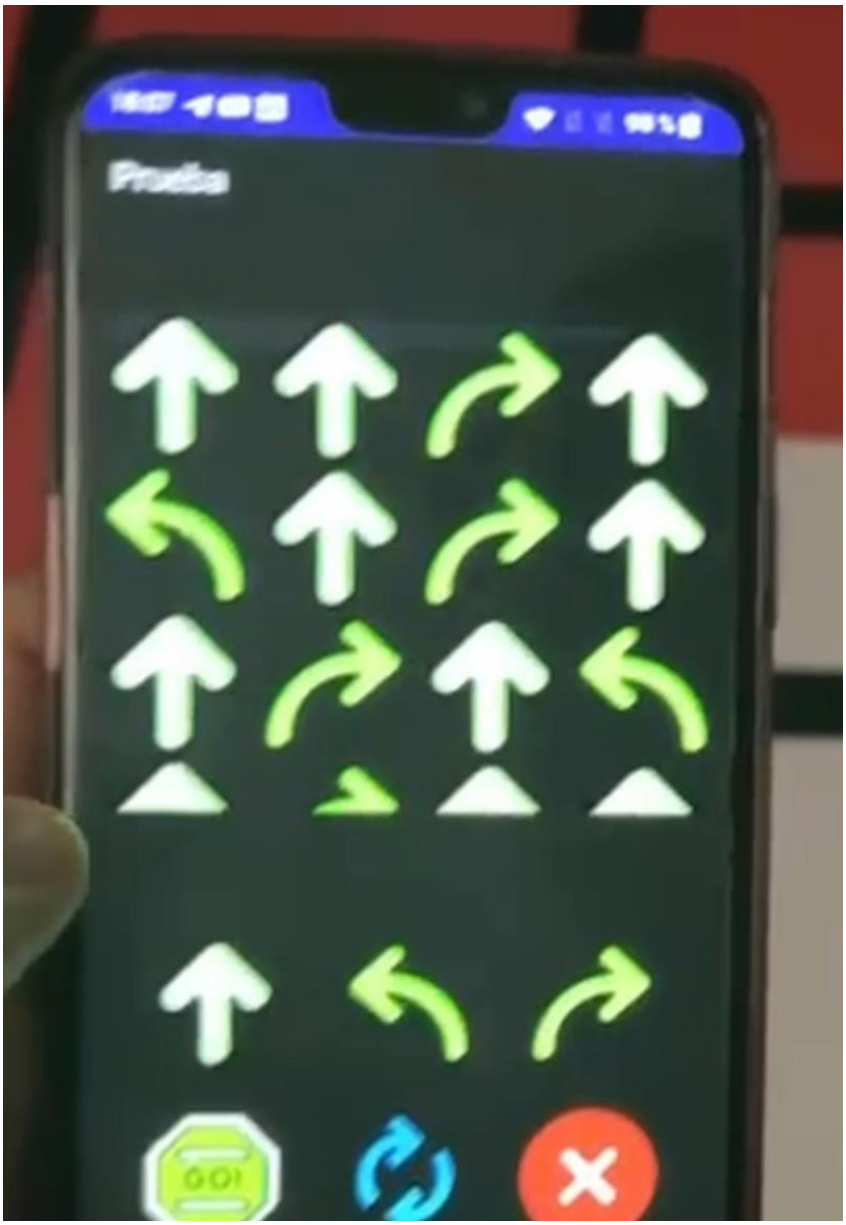


Ilustración 3.20: Listado de movimientos

Una vez que se han añadido las instrucciones, vamos a ver los movimientos que ha ido realizando el vehículo hasta su llegada a la meta.

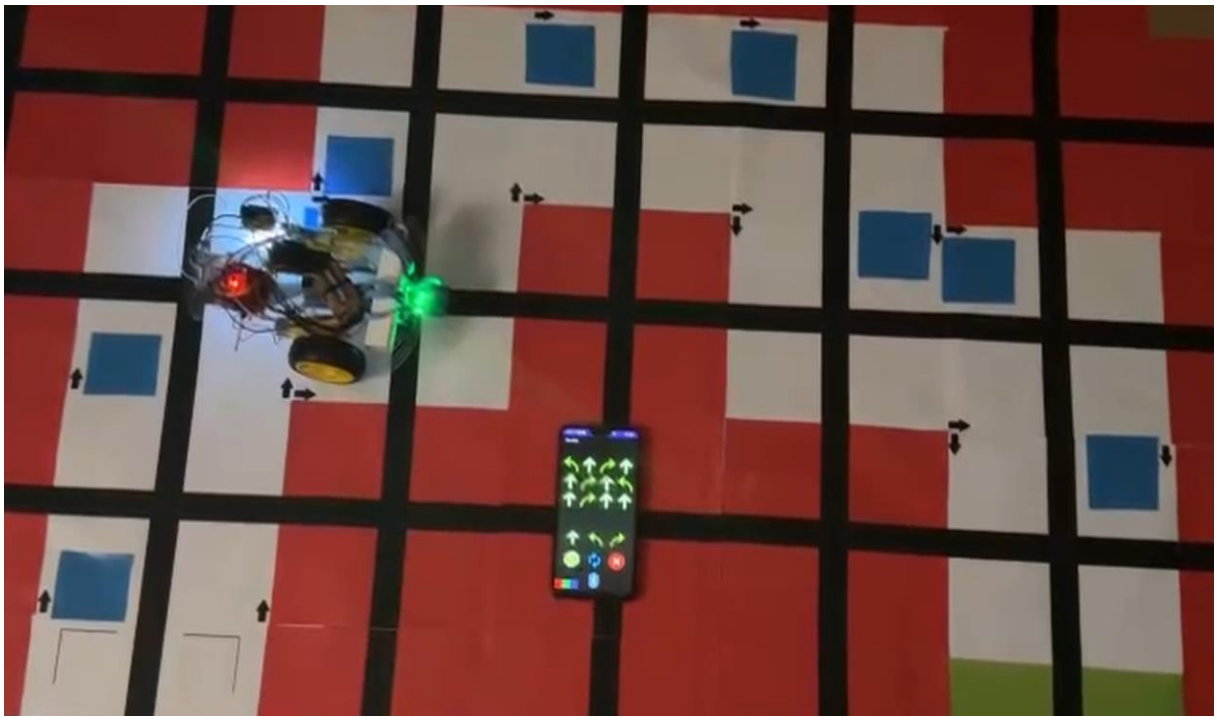


Ilustración 3.21: Secuencia prueba final parte 1

En la primera parte del circuito, vemos como el coche ha ido recto 3 casillas y ha realizado correctamente el giro a la derecha, en la pantalla del móvil se puede ver los movimientos que aún faltan por realizar.

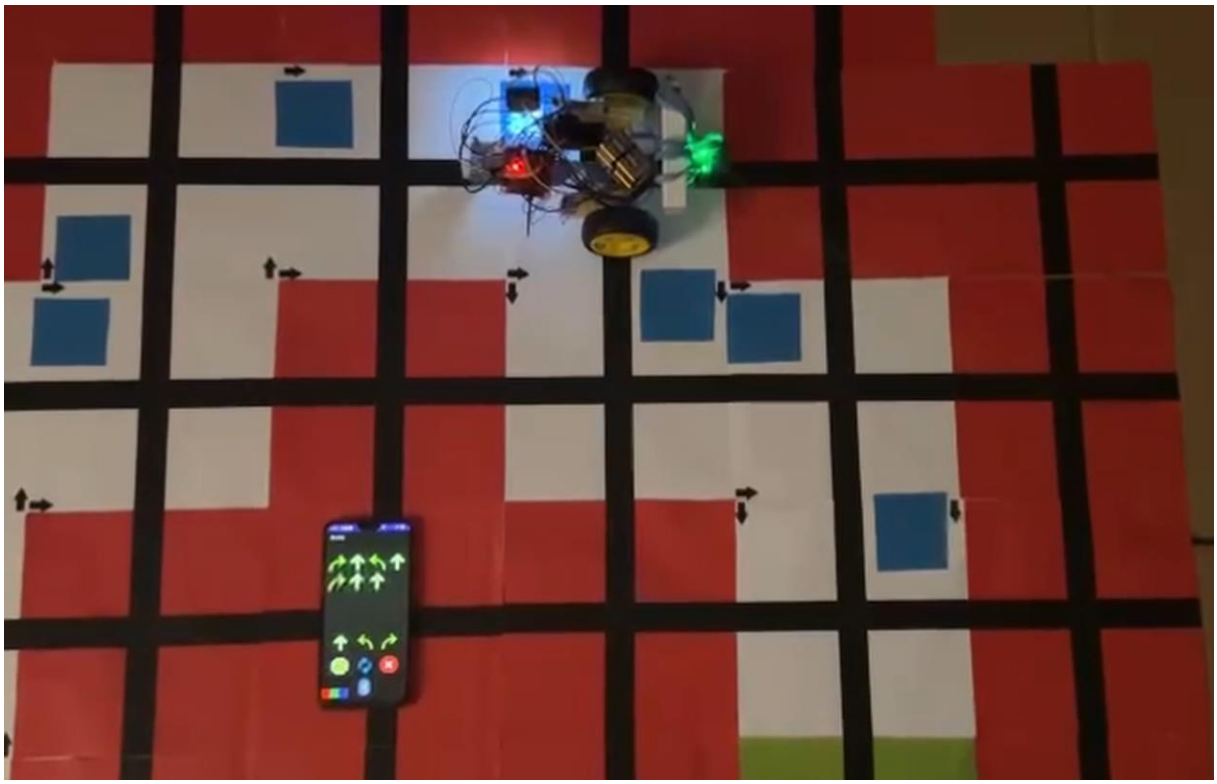


Ilustración 3.22: Secuencia prueba final parte 2

En la segunda imagen vemos como ha realizado el 50% del circuito, realizando varios giros de izquierda y derecha correctamente, vamos a continuar viendo si es capaz de llegar al final sin salirse del circuito y consigue detectar que ha llegado al final del trayecto.

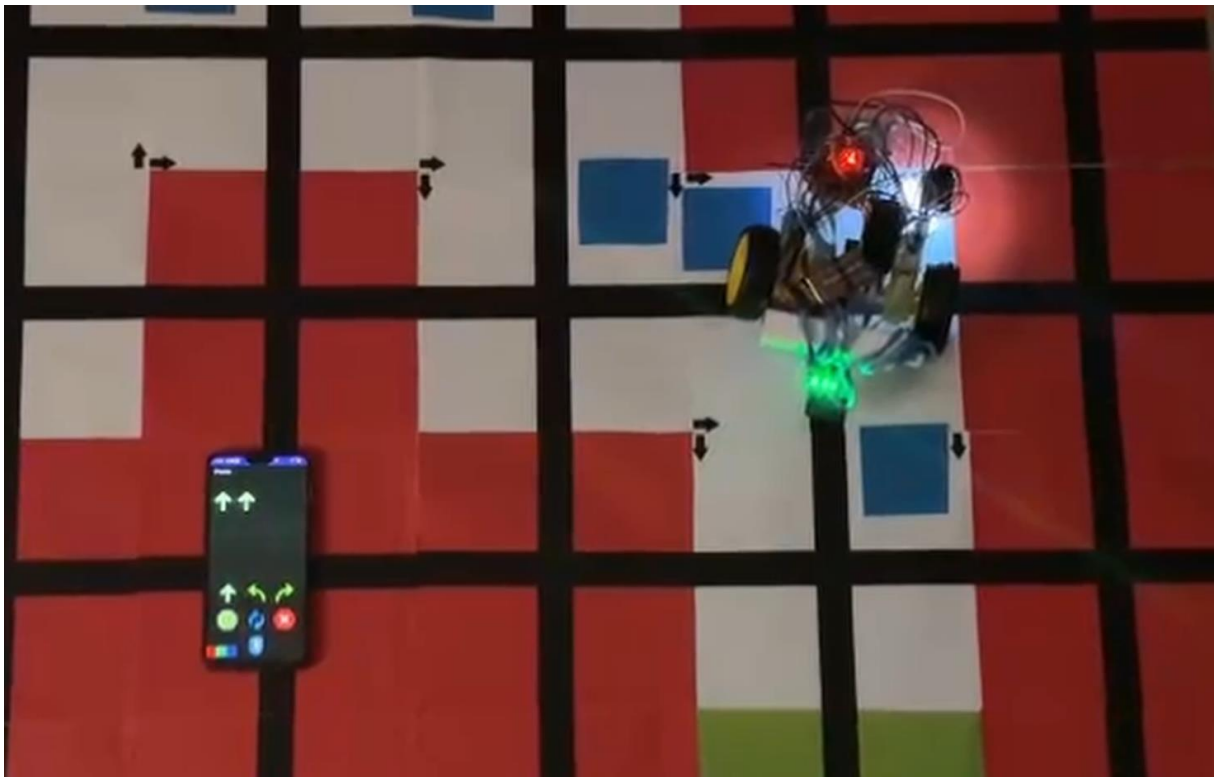


Ilustración 3.23: Secuencia prueba final parte 3

Una vez pasado todos los tramos de curvas correctamente, ya solo le queda recorrer un último tramo recto hasta la meta, que detecte esta y se pare en ella, indicando en el programa que ha llegado a la meta correctamente.

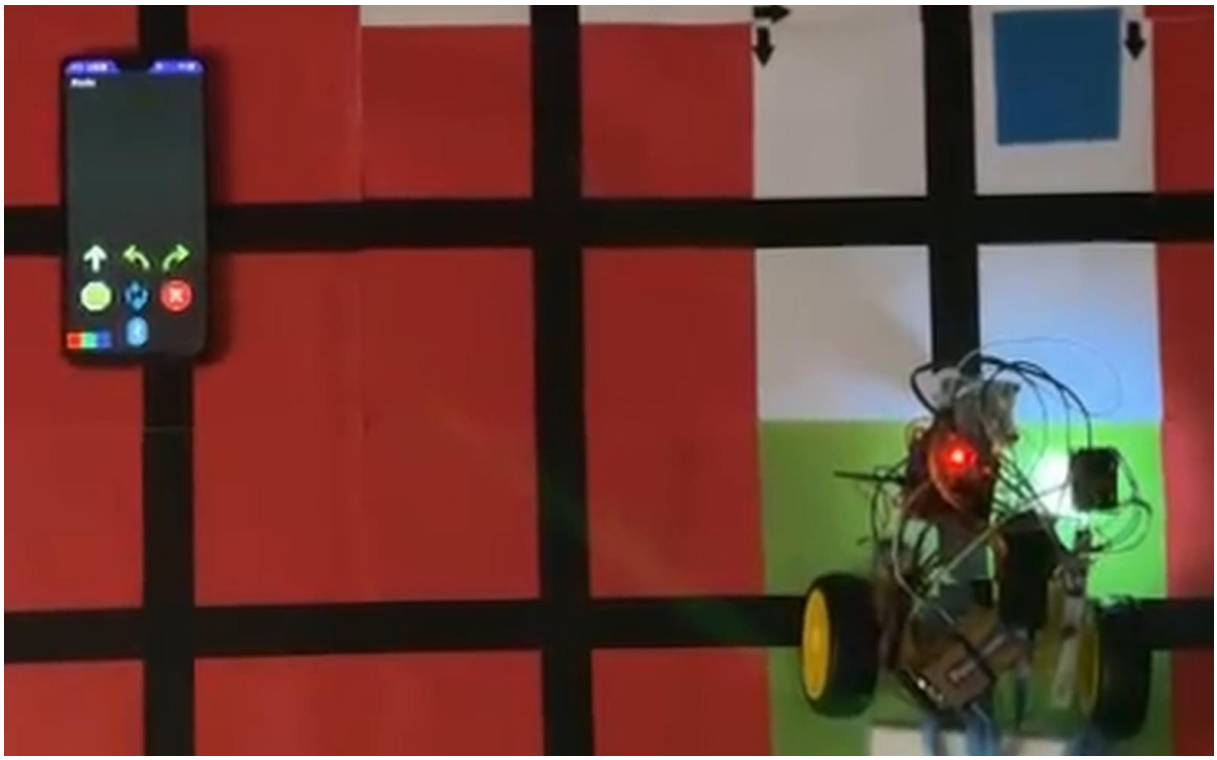


Ilustración 3.24: Secuencia prueba final parte 4

El vehículo se para correctamente al final del tramo y en la siguiente imagen se ve como la señal de la llegada a meta es detectada por nuestro coche, que envía un mensaje a nuestra aplicación y esta muestra por pantalla que se ha llegado correctamente al destino.

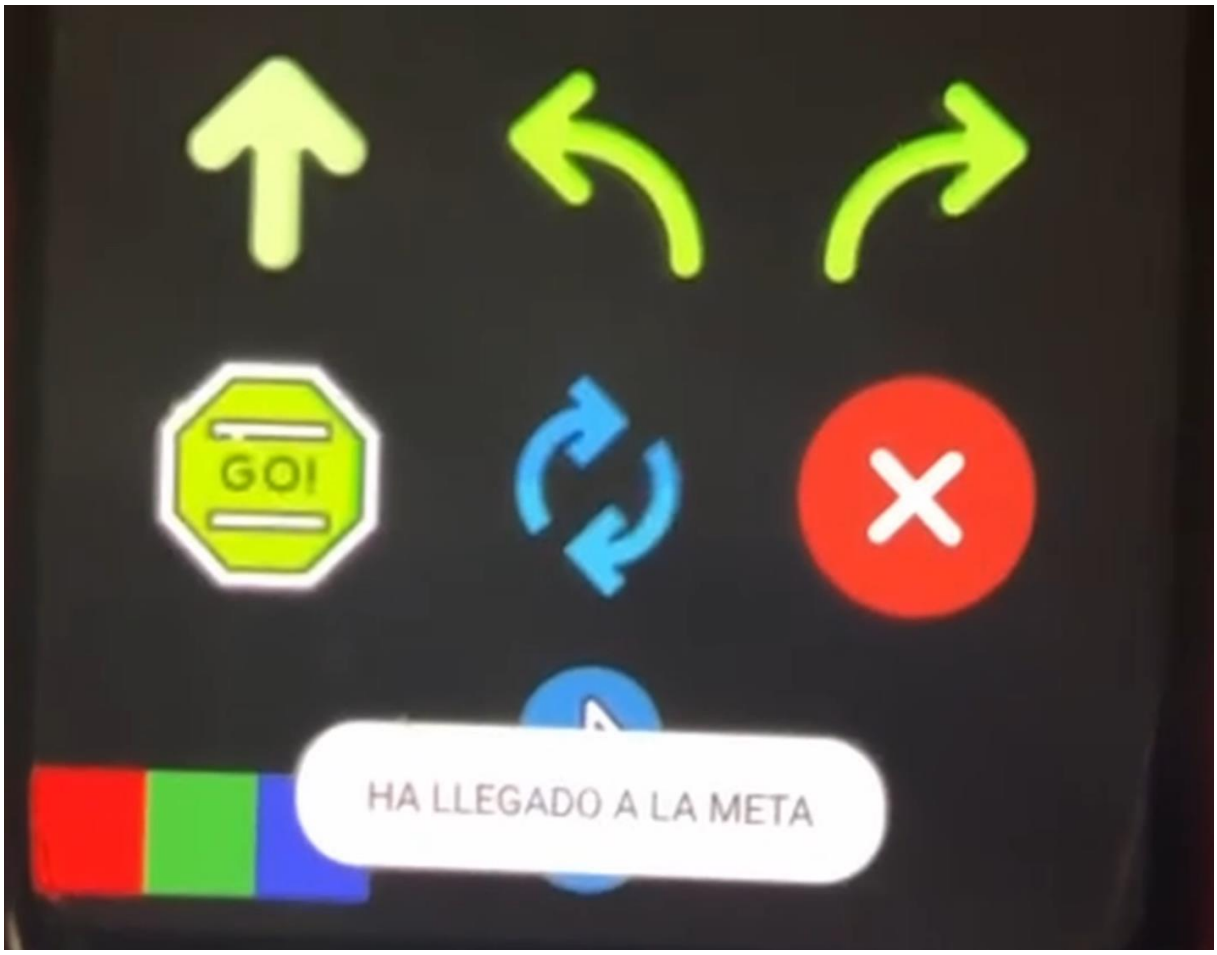


Ilustración 3.25: Secuencia prueba final, llegada a meta

Como se han visto en las imágenes, el coche ha conseguido seguir la secuencia y llegar a la meta correctamente. Cuando detecta el color de la meta ha mostrado por pantalla el mensaje informativo de llegada satisfactoria, por tanto, la prueba del circuito ha salido bien.

Ahora vamos a realizar la prueba en el modo bucle y realizando un recorrido erróneo procurando que se salga de pista para ver si funciona como se espera.

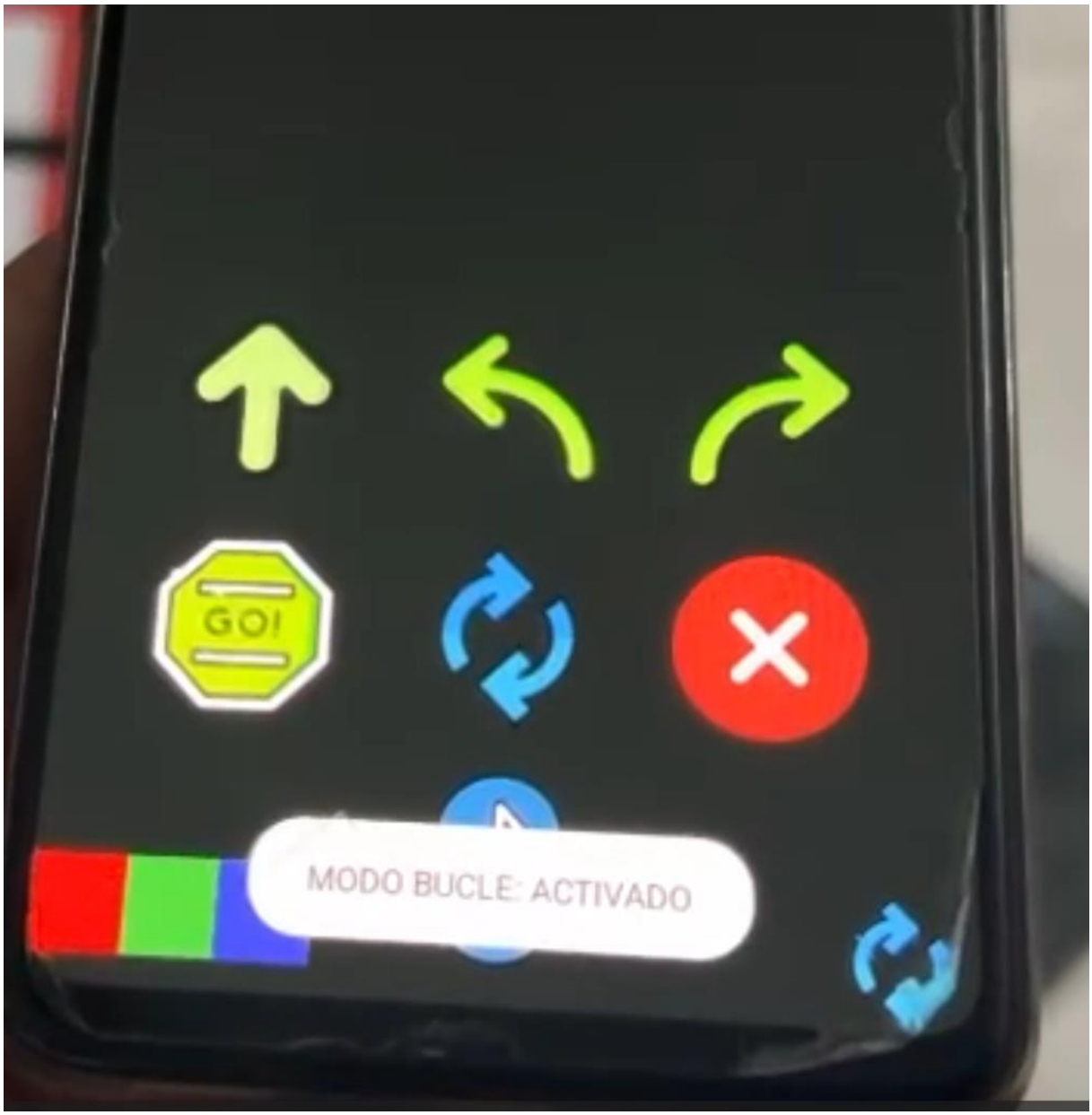


Ilustración 3.26: Activación del modo bucle en pruebas

Pulsando el botón del modo bucle, se activa el envío de la secuencia repetidas veces, hasta que se consiga llegar a meta o se salga del circuito, como se indica en la imagen. Cuando se activa este modo, se puede ver en pantalla un icono abajo a la derecha que nos indica que vamos a activar esa secuencia repetidas veces, también se indica con un mensaje al principio de que el modo ha cambiado.

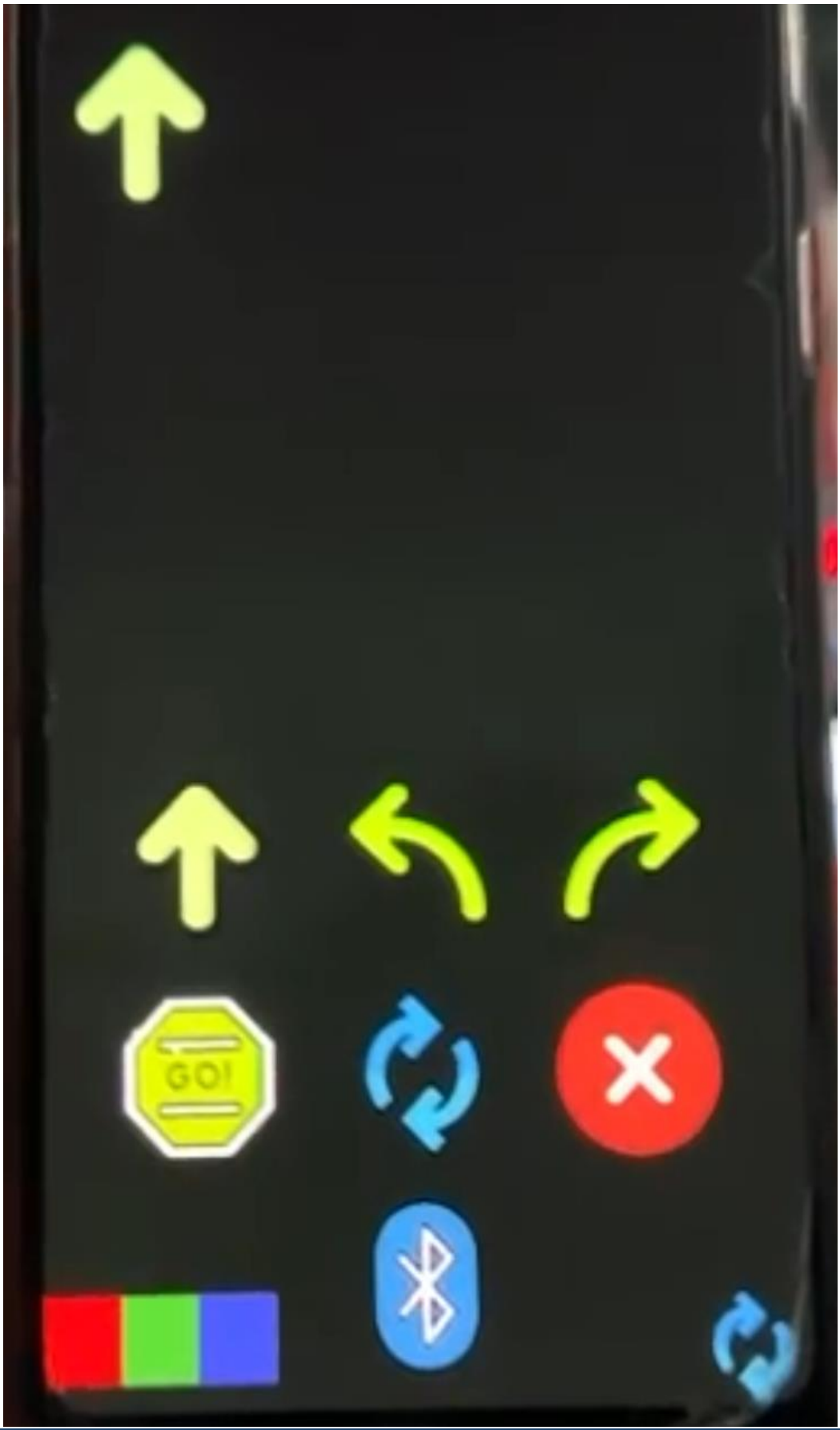

GO!

Ilustración 3.27: Movimientos insertados en modo bucle

Activamos el modo bucle y procedemos a ejecutar la instrucción para que el coche vaya únicamente hacia delante y procurar que se salga de pista.

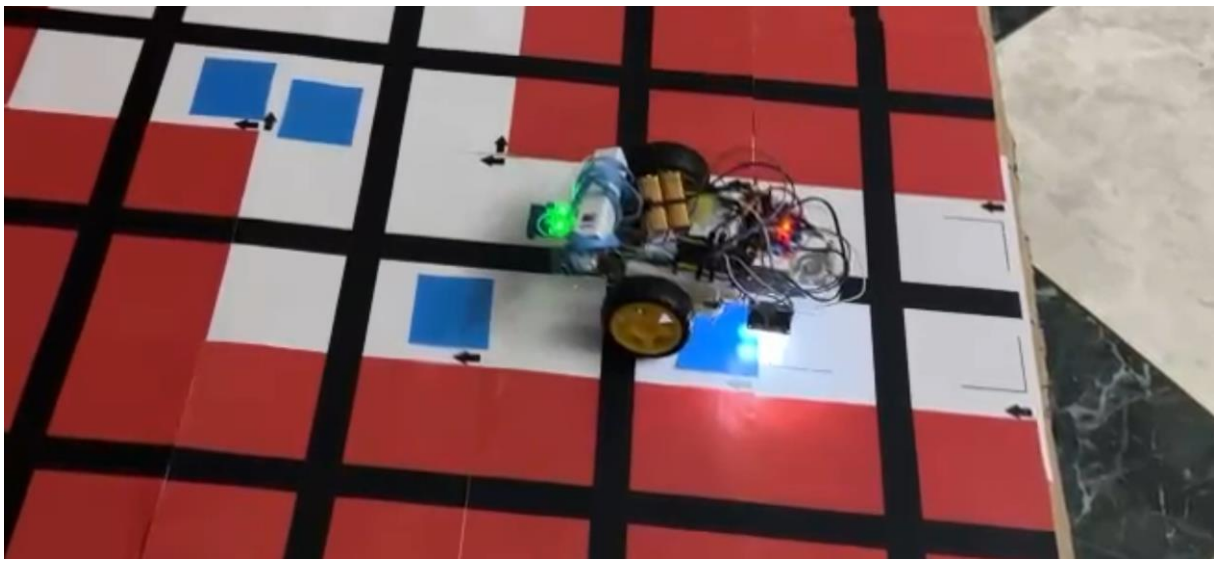


Ilustración 3.28: Secuencia modo bucle parte 1

Comienza el recorrido y nuestro vehículo se para en la primera casilla, si no estuviese el modo bucle, con la secuencia normal, el coche se quedaría en esa casilla parado, pero como vamos a ver a continuación, vuelve a ejecutarse el mismo movimiento.

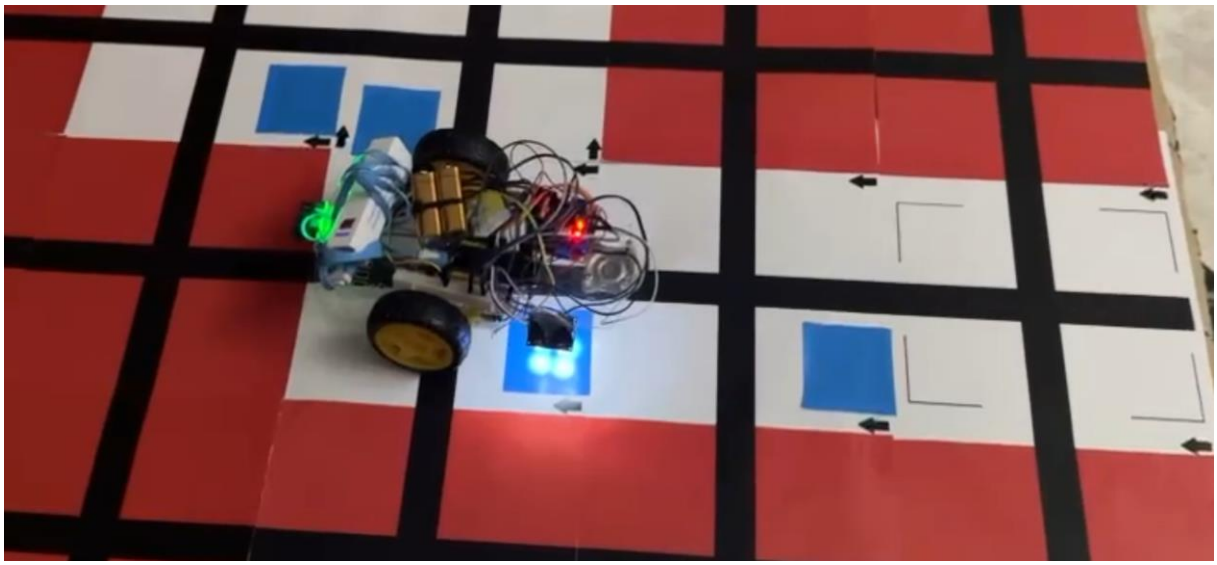


Ilustración 3.29: Secuencia modo bucle parte 2

Aquí podemos ver como se ha ejecutado correctamente nuestro bucle de movimientos. Vamos a ver cómo actúa ahora el coche al salirse del circuito y si se para al detectar esta situación.

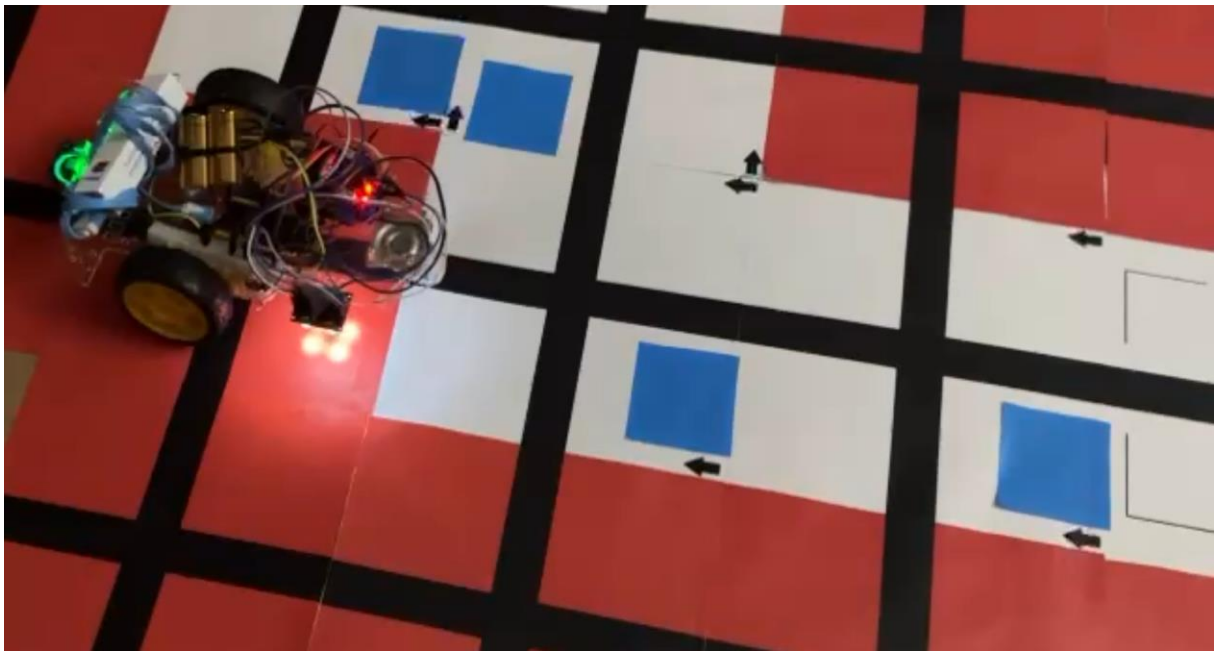


Ilustración 3.30: Secuencia modo bucle parte 3

El vehículo se para fuera del circuito, como se muestra en la imagen, y acto seguido envía un mensaje de error a la app donde le indica que se ha salido del circuito y se para todo el proceso de secuencia.

Ilustración 3.31: Secuencia modo bucle parte 4

Finalmente, como estaba previsto, el coche se sale de pista. Al detectar el color rojo de la casilla de fuera, deja de ejecutar la secuencia de movimientos y saca por pantalla el mensaje informativo de que el sistema se ha salido del circuito. Esto quiere decir que tanto la conectividad entre la app y el coche, como el mecanismo del propio vehículo funcionan perfectamente en condiciones favorables.

3.5 Cuarta Iteración

En esta cuarta y última iteración, vamos a enfocarnos en la interfaz de nuestra aplicación móvil.

Una vez conseguido que todo funcione correctamente y completada la funcionalidad de sistema, se han realizado mejoras a la interfaz de usuario. El objetivo

es proporcionar una interacción que sea intuitiva y que al tiempo contribuya a facilitar el aprendizaje de la programación a los menores proporcionando elementos de retroalimentación más visuales.

También, vamos a aprovechar esta iteración para mejorar las funcionalidades que ya teníamos y añadir nuevas que puedan ayudar a un mejor entendimiento en el caso de que el niño cometa un error al jugar. A su vez, mejoraremos aquellas que ya teníamos y que a lo largo del periodo de pruebas nos ha parecido que necesitaban un remodelado o se han encontrado fallas de implementación.

3.5.1 Tareas

En este apartado se resumirá alguna de las tareas de mejora en la interfaz que se van a realizar en esta iteración, así como añadir nuevas funcionalidades a la misma:

1. Remodelado completo de la interfaz, en donde se diferencie bien cuales son los botones de movimiento y cuáles los elementos de secuencia.
2. Mensaje al iniciar la app que nos recuerde que tenemos que hacer un escaneo previo de los colores para un correcto funcionamiento de la app.
3. Añadir un botón de ajustes que nos lleve a una nueva pantalla para el escaneo de los colores.
4. Marcar el movimiento que está ejecutando el vehículo en cada momento para que sea mucho más visual en la app.
5. Evitar que la secuencia de movimientos que previamente se ha realizado se borre al completarse. De este modo, se puede ejecutar más de una vez, reforzando positivamente el logro alcanzado por el menor.
6. Implementar un botón que borre el último elemento añadido, y no la secuencia completa como estaba anteriormente. De esta forma, el menor puede corregir/modificar la programación realizada si se ha equivocado.

7. Añadir elementos visuales y sonoros que interactúen con el niño cuando el coche llega a su destino o por el contrario se sale del circuito.

3.5.2 Desarrollo de la iteración y pruebas de verificación

Vamos a ver todas las mejoras que se han realizado y el motivo de las mismas.

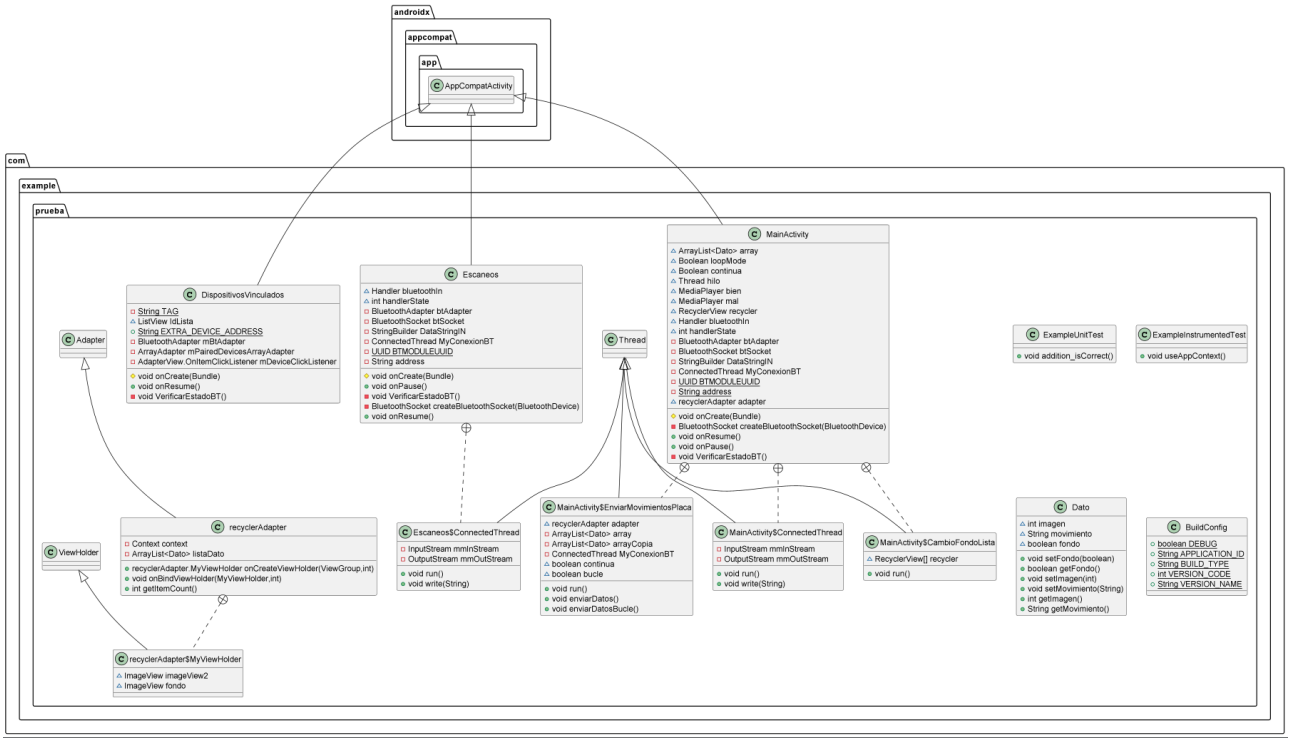


Ilustración 3.32: Uml de las mejoras

Este es el nuevo diagrama UML que se ha planteado para el remodelado de la aplicación, las funcionalidades principales no se han modificado como se puede ver, se han añadido un par de características nuevas que veremos más adelante.

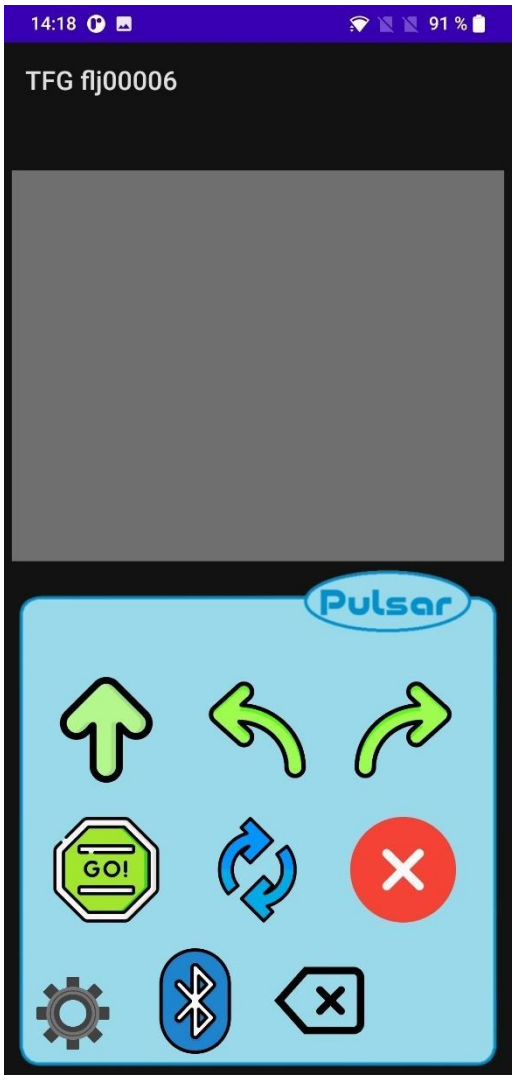


Ilustración 3.33: Pantalla principal mejorada

Uno de los cambios principales y que se pueden apreciar a simple vista es el cambio en la interfaz del usuario de la pantalla principal. Se ha agregado un fondo para los botones de acción y se ha cambiado también el fondo de la lista de movimientos, que más adelante veremos que es interactivo y responderá si el coche llega a la meta o se sale del circuito con colores.

Se ha añadido también un botón de ajustes, sustituyendo a los 3 botones que había en esa misma posición para escanear los colores. Esto se ha realizado así ya que es una funcionalidad que está separada de nuestro programa principal, y que solo

utilizaremos la primera vez que entremos a nuestra aplicación. Además se, se ha considerado que es mejor quitarlo para evitar que el niño sin darse cuenta o por curiosidad toque estos botones y desconfigure el robot sin saberlo, por tanto, se ha separado en una nueva pantalla que luce así:

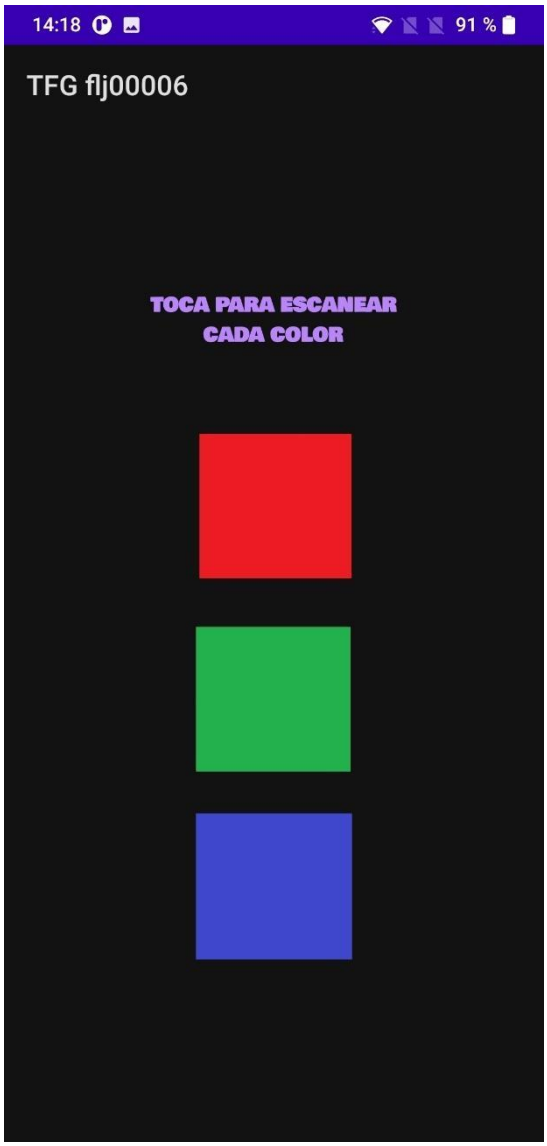


Ilustración 3.34: Ajustes de escaneo independientes

Se han dispuesto de nuevo los 3 botones interactivos con una breve descripción de cómo se deben de usar, estos botones tienen la misma funcionalidad

que los antiguos que estaban en la pantalla principal. Por tanto, ahora el supervisor se encarga de escanear estos colores al principio del programa y, al encontrarse en una pantalla aparte, no tiene el riesgo de que se toquen por error y se rompa nuestro sistema.

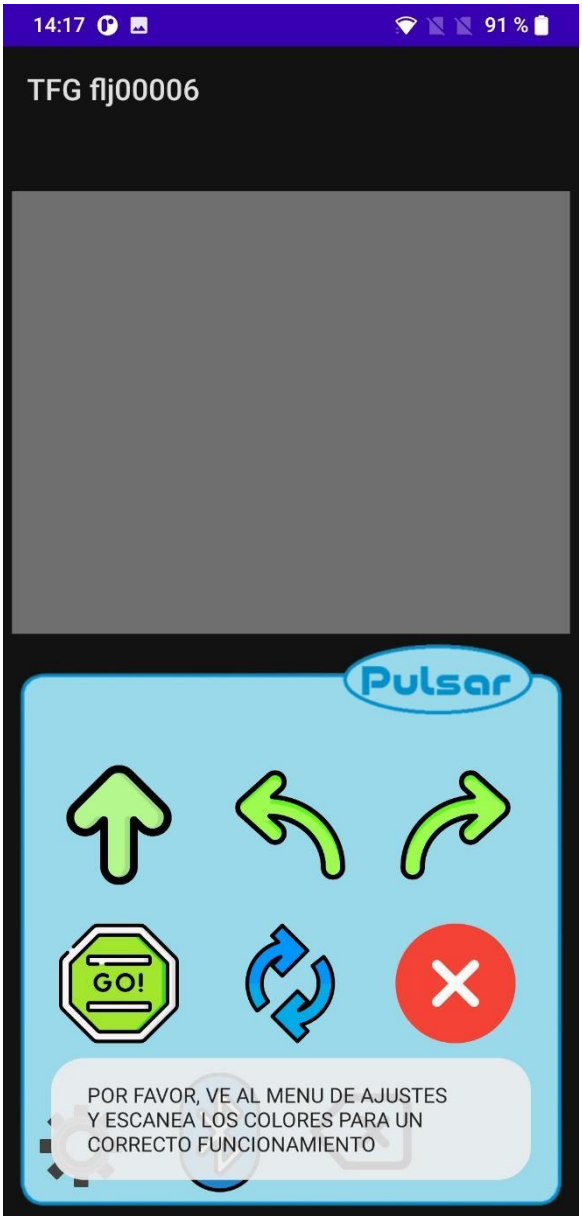


Ilustración 3.35: Mensaje inicial de ajuste

De igual modo, se ha añadido también un mensaje recordatorio cuando iniciamos la app por primera vez que le indica al usuario que tiene que escanear los colores pulsando en este botón de ajustes.

Se ha incorporado una nueva funcionalidad, un botón que nos permite borrar el último movimiento que se ha añadido a nuestra secuencia de movimientos. Anteriormente solo teníamos un botón que nos borraba la secuencia completa. Esto puede ser muy frustrante para el niño si se equivoca varias veces y tiene que escribir nuevamente la secuencia desde cero.

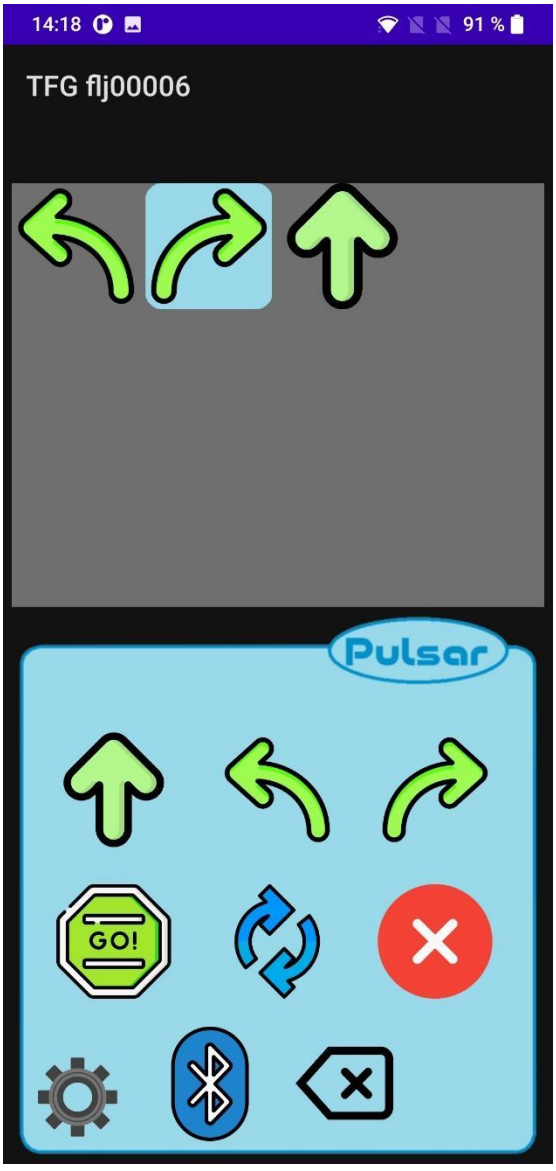


Ilustración 3.36: Nuevo diseño de secuencia en la app

Como se muestra en la imagen, se ha añadido un fondo claro para cada movimiento que indica al usuario cual es la acción que se está realizando en cada

momento por nuestro robot. Gracias a esto, el niño puede reconocer visualmente cual es el movimiento equivalente que se realiza en la aplicación y, en caso de fallo, cual es el último movimiento que se realizó antes de fallar, para poder posteriormente utilizar la herramienta nueva de borrado como veremos más adelante y modificar su secuencia.

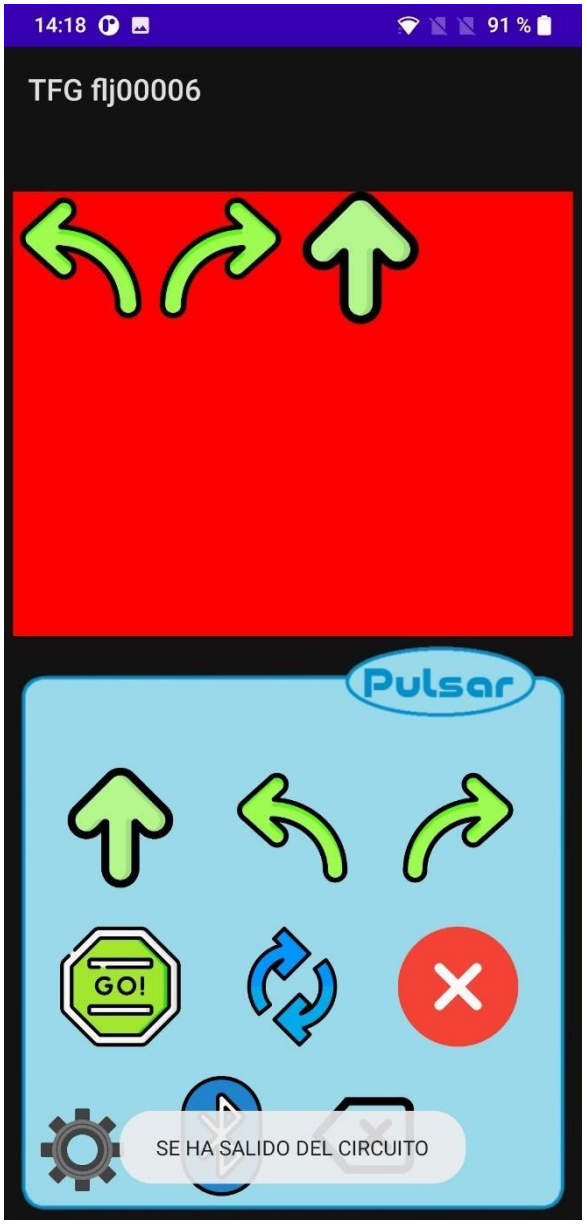


Ilustración 3.37: Nueva interfaz de error

Una vez completada la secuencia de ejemplo que pusimos anteriormente, podemos observar que el coche se ha salido del circuito y la interfaz de la secuencia se ha puesto de color rojo indicando un fallo, también se genera un sonido cada vez que fallamos, que da al niño un feedback auditivo de que ha fallado en su secuencia y que tiene que volver a intentarlo.

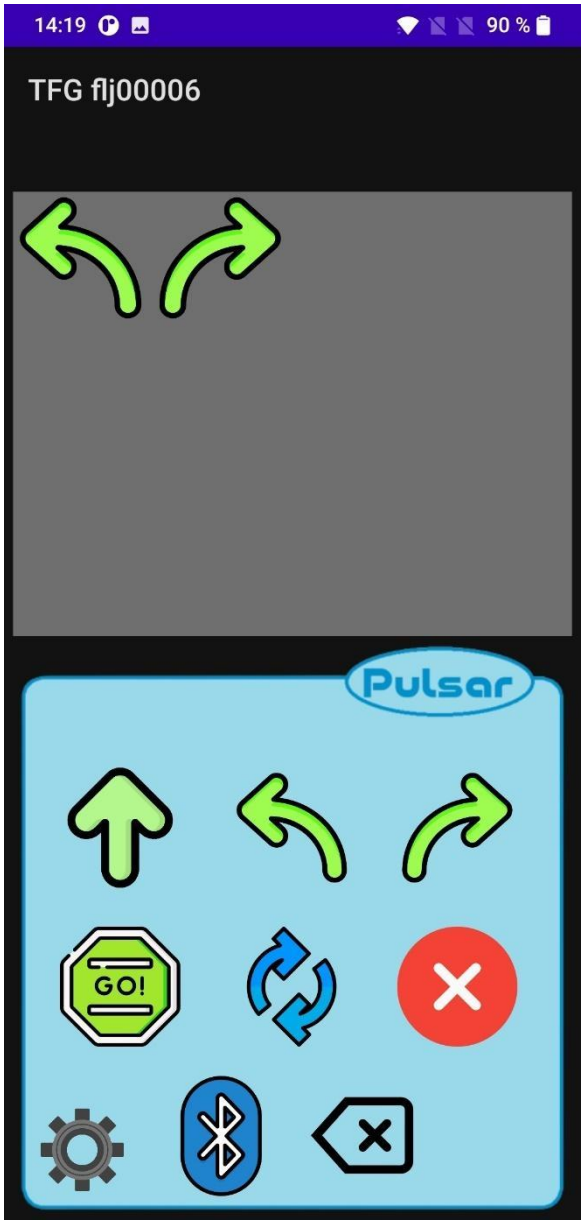


Ilustración 3.38: Deshacer últimos movimientos

Gracias al nuevo botón de deshacer, podemos borrar la última parte de la secuencia que se ejecutó anteriormente para así poder seguir probando con una secuencia distinta desde el punto en el que se salió del circuito. Ahora también se guarda la secuencia anterior, cosa que antes no pasaba ya que se eliminaba automáticamente al fallar o parar nuestro proceso.

El botón de deshacer no solo borra, sino que también provoca que se pare de ejecutar la secuencia de movimientos actuales.

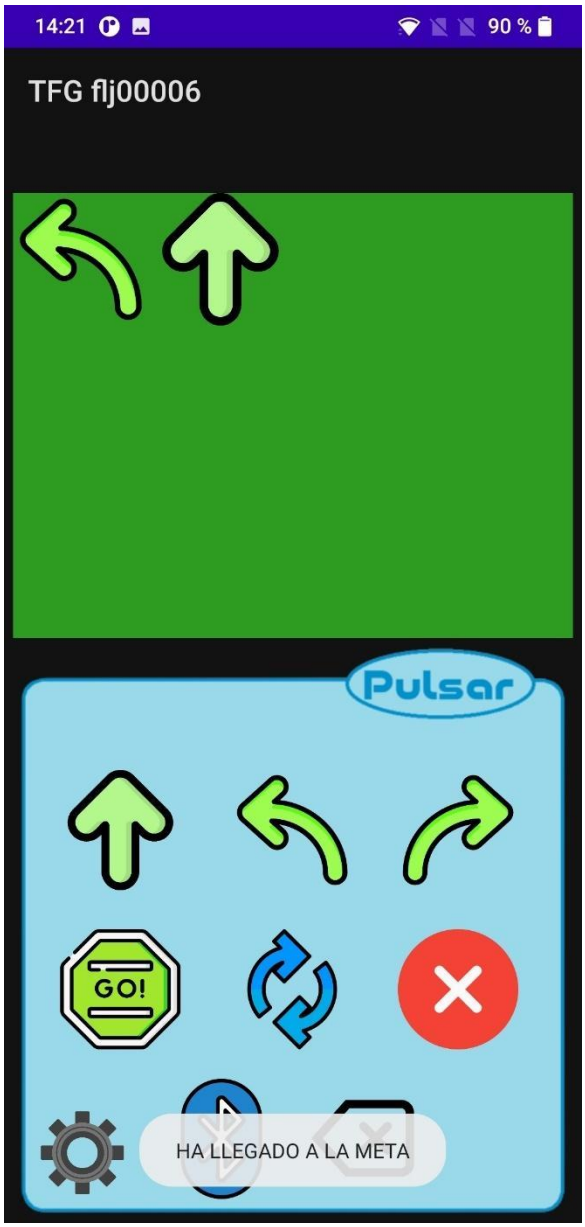


Ilustración 3.39: Nueva interfaz de llegada correcta

Cuando el coche llega al destino y la aplicación lo detecta, ahora el fondo de nuestra lista de movimientos, se vuelve verde. Al igual que pasa con la animación de error, en este caso se envía un sonido que hace referencia a unas palmas indicando que se ha hecho correctamente y así el niño tiene un feedback positivo por parte de la aplicación.

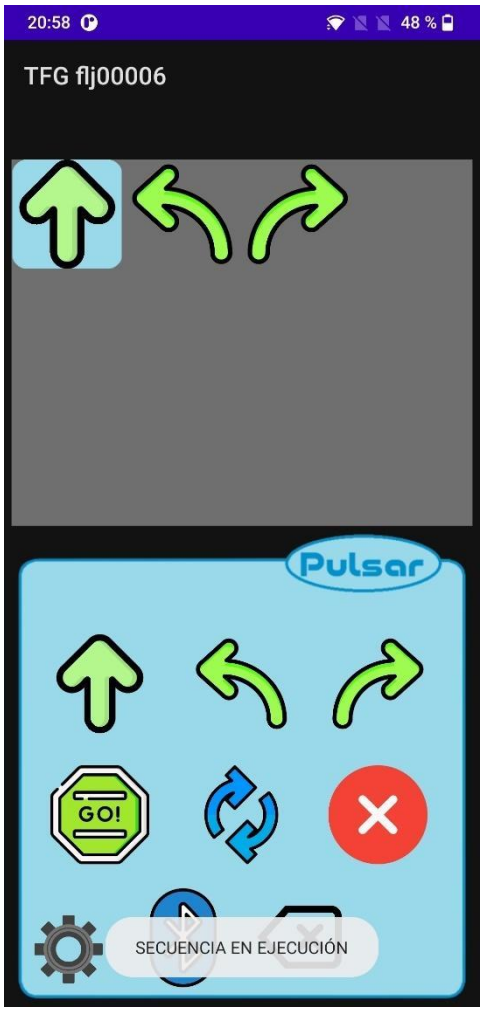


Ilustración 3.40: Mejora de fallas en la app

Para terminar, se ha solucionado un error de implementación que permitía agregar más movimientos a la lista mientras estaba en ejecución, así como enviar otra secuencia de movimientos que se solapaba con la que estaba ejecutando. Ahora es mucho más segura en ese aspecto, ya que cuando se toca una opción que no está permitida mientras hay algo ejecutando nos avisa con un mensaje que nos indica que la secuencia ya está en ejecución. Para poder volver a realizar otra secuencia sería necesario parar la ejecución actual y cambiar la secuencia de movimientos.

Estas son todas las modificaciones y correcciones que se han realizado una vez ya se había comprobado con la anterior versión de la aplicación que todas las

funcionalidades y conectividad con el sistema hardware eran correctas. Ahora la interfaz de la aplicación está libre de errores, es mucho más intuitiva, facilita el aprendizaje de la programación de la secuenciación de movimientos y retroalimenta de forma positiva al menor.

4 ESCENARIOS DE USO Y APLICACIONES EN LA ENSEÑANZA

El propósito inicial de este proyecto es facilitar los principios básicos de la programación a los menores en su etapa infantil o primer ciclo de primaria en el aula o en casa de forma supervisada.

Enseña a los menores a aprender a programar y dar instrucciones de manera motivante. Es muy bueno porque con esto van a captar la atención y despertar su interés de manera sencilla. Funciona para desarrollar la lógica, el pensamiento computacional y la lateralidad, ya que deben diferenciar entre izquierda y derecha. También funciona especialmente bien para que aprendan a dar órdenes sencillas en cadena e incluso se atrevan a dar sus primeros pasos comprendiendo los bucles sencillos que pueden realizar con la opción dada en la app.

No obstante, dada su simplicidad y versatilidad las aplicaciones en la enseñanza pueden ser múltiples y mucho más diversas.

Por ejemplo, los pequeños pueden adquirir mucho vocabulario con el robot porque tú puedes poner diferentes objetos o imágenes por el circuito y darle la orden de a cuál quieren que vaya para que los peques lo programen. Un posible uso de esto se muestra a continuación:

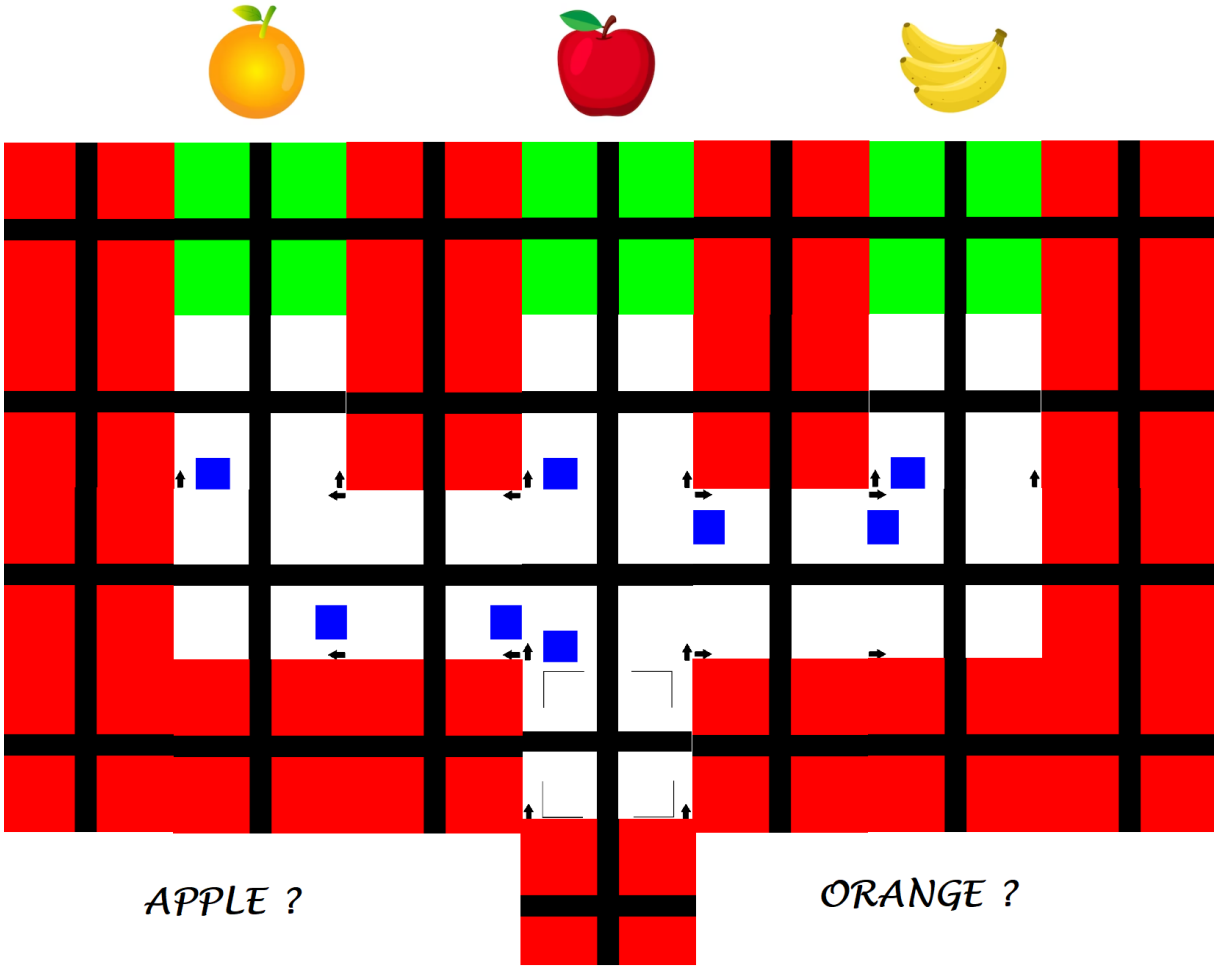


Ilustración 4.1: Ejemplo 1 de diseño para la enseñanza

Como vemos, no solo podemos hacer un camino simple para que el niño aprenda una secuencia lógica de movimientos, sino que también podemos enseñarle muchas cosas donde tendrá que elegir entre una opción u otra. Por ejemplo, el profesor puede indicarle al niño que tiene que utilizar el camino de forma que vaya hacia la fruta cuyo nombre en inglés es "*Apple".*

No solo se puede trabajar con el lenguaje, si no que se podría realizar cualquier posible cuenta sencilla en la que un niño tenga que elegir una solución mediante varios caminos posibles.

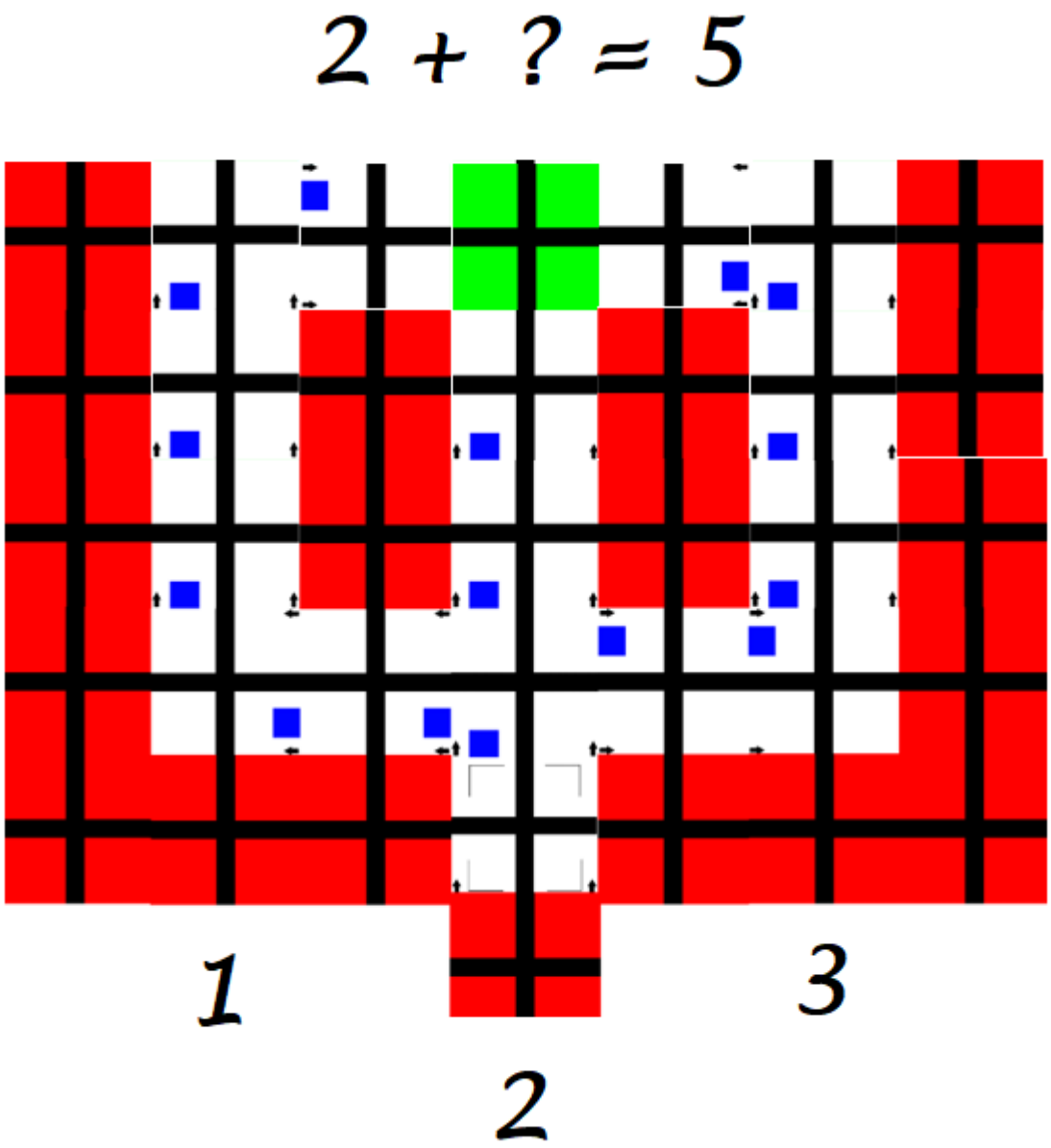


Ilustración 4.2: Ejemplo 2 de diseño para la enseñanza

En este caso, el niño tiene que elegir que camino tomar para que la cuenta sea correcta. Aquí, el aprendizaje tiene que estar supervisado por un adulto que le proponga diferentes problemas al niño adaptados a su edad y conocimiento. En el caso de fallar la cuenta pueda ser el profesor el que le eche un cable al niño para asegurar su enseñanza sobre el tema.

Un ejemplo muy práctico para hacer también con los pequeños sería darles el inicio y el final de un circuito, y que sean ellos los que sean capaces de crear un circuito con láminas que lleve al coche del punto de partida hasta el final y lo programen en la app para ver si su solución es la correcta o no.

Otra posible aplicación también podría ser un camino recto con varias opciones en las que hacer que el coche se pare en la casilla en la que se encuentre la solución.

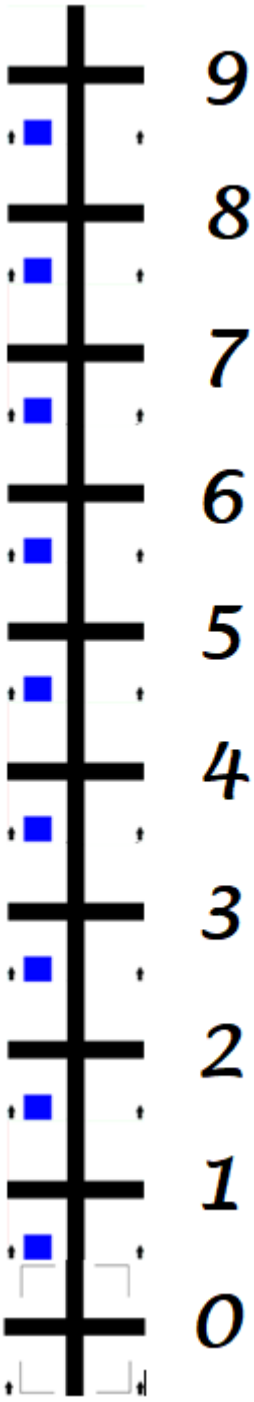


Ilustración 4.3: Ejemplo 3 de diseño para la enseñanza

En este caso, el coche comenzaría desde la casilla de salida y el niño tendría que hacer un movimiento hacia delante tantas veces como sea la solución. Algunos de los problemas que se podrían plantear podrían ser los siguientes:

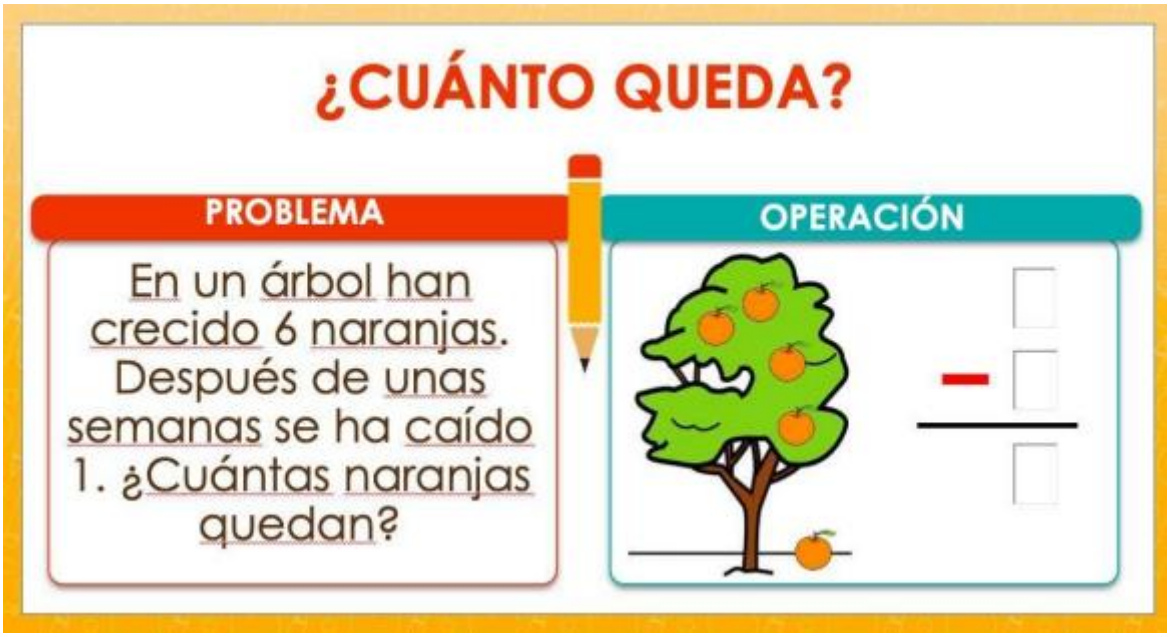


Ilustración 4.4: Problema ejemplo nº1(Fuente:[30])

Ilustración 4.5: Problema ejemplo nº2(Fuente:[30])

Ilustración 4.6: Problema ejemplo nº3(Fuente:[30])

Como se puede ver, con este proyecto hay multitudes de opciones que se pueden pensar para realizar actividades con los más pequeños mientras éstos aprenden jugando. Al final de este proyecto, se dejará en el anexo las plantillas para que cualquiera pueda hacer sus circuitos en casa, ya sea impresos en folios, en papel de pegatina o de la forma que se desee.

5 CONCLUSIONES Y TRABAJOS FUTUROS

Gracias a este proyecto he adquirido grandes conocimientos de programación móvil y programación de sistemas IOT con la placa ESP32, también he aprendido mucho sobre el mundo de la electrónica.

Este proyecto no solo me ha servido para mejorar mis dotes en la programación y electrónica de componentes, si no que he aprendido mucho sobre la enseñanza infantil y de primaria, y cómo esta educación puede mejorar gracias al uso de la tecnología. Se ha optado por hacer algo innovador en donde creo que un niño se

puede sentir mucho más cómodo aprendiendo de forma divertida con un juguete real que puede encontrar en casa y manipular y prácticamente sin darse cuenta, y no con una pantalla delante.

Creo que es un trabajo que tiene mucho potencial y que sería muy recomendable utilizar en la enseñanza actual en edades tempranas, ya que como se ha explicado a lo largo del proyecto, puede ayudar a los niños a un mejor desarrollo, y gracias a la flexibilidad de la creación de distintos entornos, puede tener mayores aplicaciones que las que se ha contado en este trabajo.

El funcionamiento de nuestro sistema al completo es correcto, pero es un prototipo que en un futuro se mejorará para hacerlo mucho más confiable como, por ejemplo:

- El uso de un circuito desmontable sería ideal si queremos llegar a comercializarlo en un futuro.
- Una mejora de los sensores, ya que hay algunos que son poco sensibles a los cambios del entorno como es el sensor de la luz, el cual puede llegar a fallar en distintas ocasiones.
- Una buena carcasa para el diseño de nuestro coche, ya que como prototipo hemos dejado los cables y la circuitería al descubierto, pero lo ideal sería que tenga una carcasa protectora externa para que los usuarios no tengan que estar manipulando con sumo cuidado para no romper nada.

Como conclusión final, ha sido un gran desafío partiendo de la poca base que se enseña en la carrera sobre estos temas, pero me ha resultado muy interesante y todo un reto que quería asumir como ingeniero.

6 APÉNDICES

6.1 Instalación y configuración del sistema

Para instalar la app necesaria en nuestro móvil vamos a seguir los siguientes pasos:

1. Vamos a descargar el archivo .apk y desde el administrador de archivos de nuestro teléfono vamos a abrirlo.

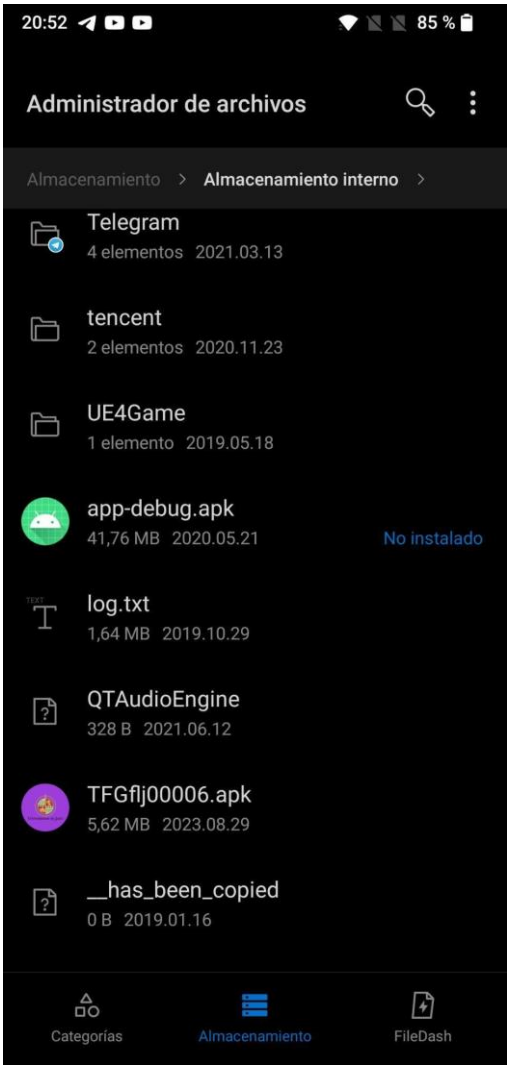


Ilustración 6.1: Instalación apk parte 1

2. Pulsamos en la app y la abrimos con el instalador de paquetes.

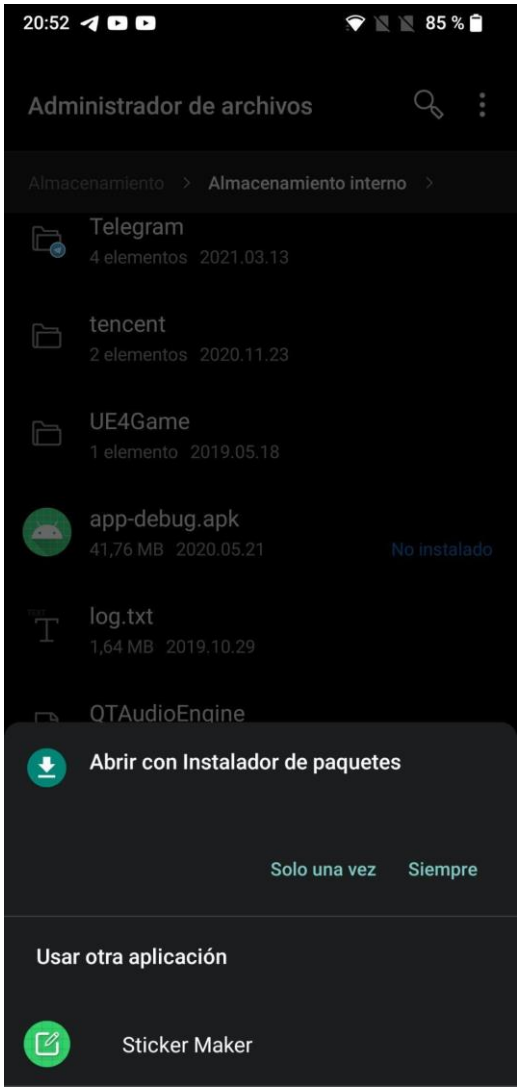


Ilustración 6.2: Instalación apk parte 2

3. Una vez abierto, pulsamos en instalar y esperamos a que se instale la app.

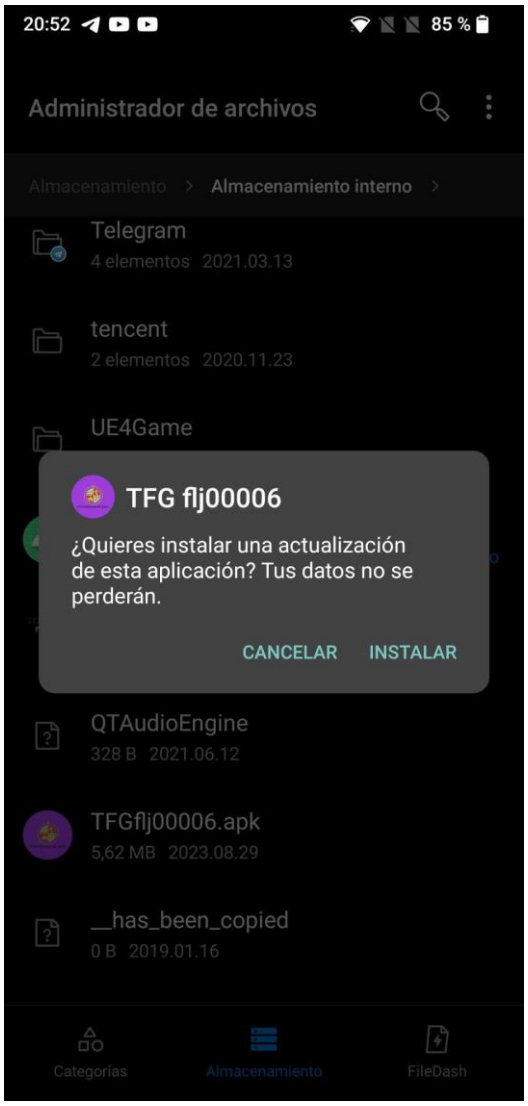


Ilustración 6.3: Instalación apk parte 3

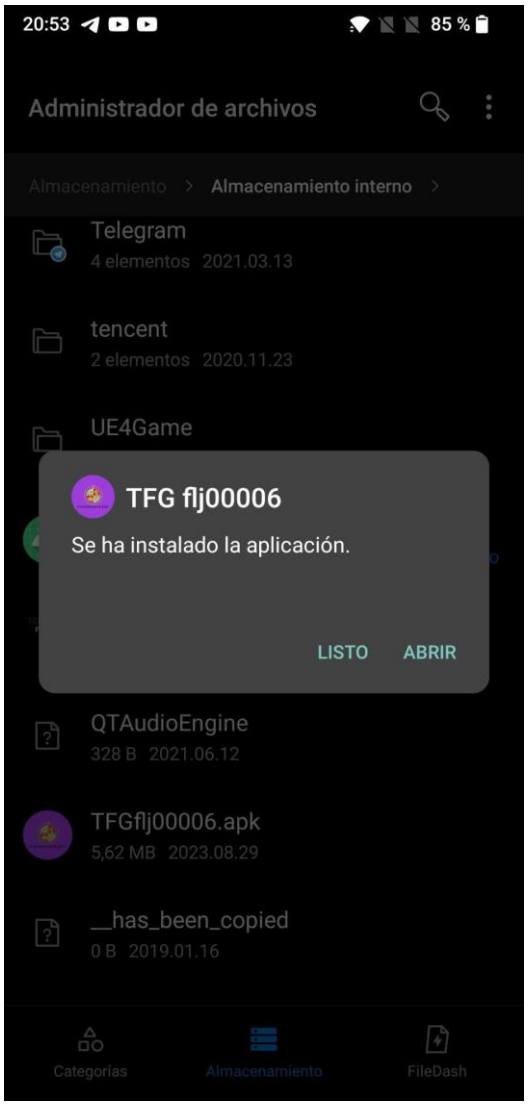


Ilustración 6.4: Instalación apk parte 4

Una vez seguidos los siguientes pasos ya tendremos la app en nuestro dispositivo lista para ser utilizada.

6.2 Manuales de usuario

El uso de la aplicación es muy simple y se van a explicar todas las opciones con imágenes y una leyenda que explica cada parte de la interfaz que vamos a encontrarnos.

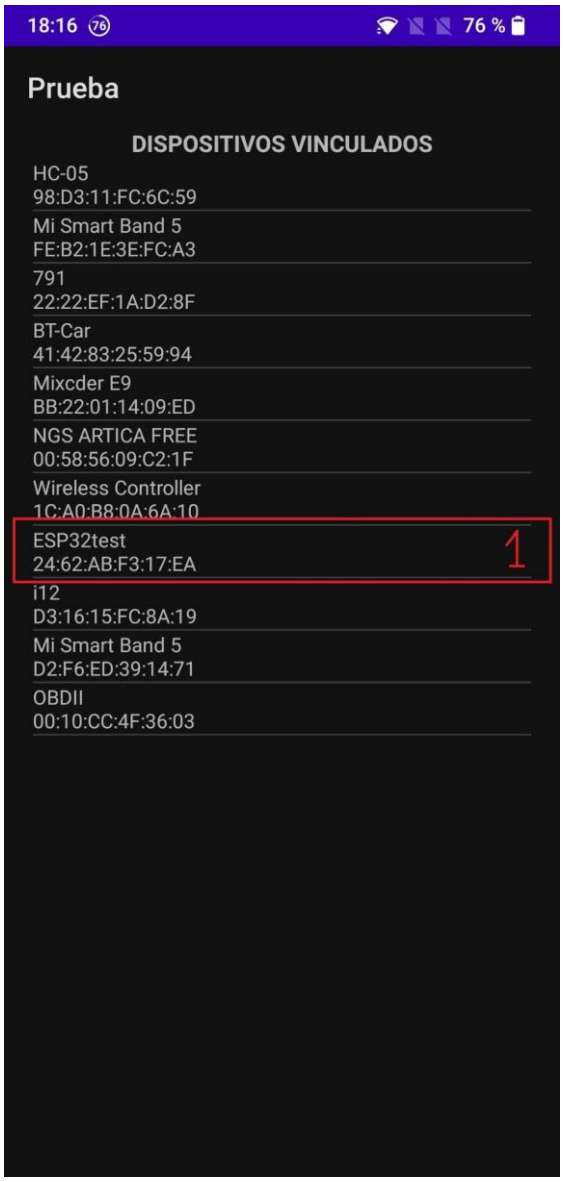


Ilustración 6.5: Manual de usuario parte 1

1. En la pantalla de los dispositivos vinculados vamos a pulsar el dispositivo del ESP32 del coche.

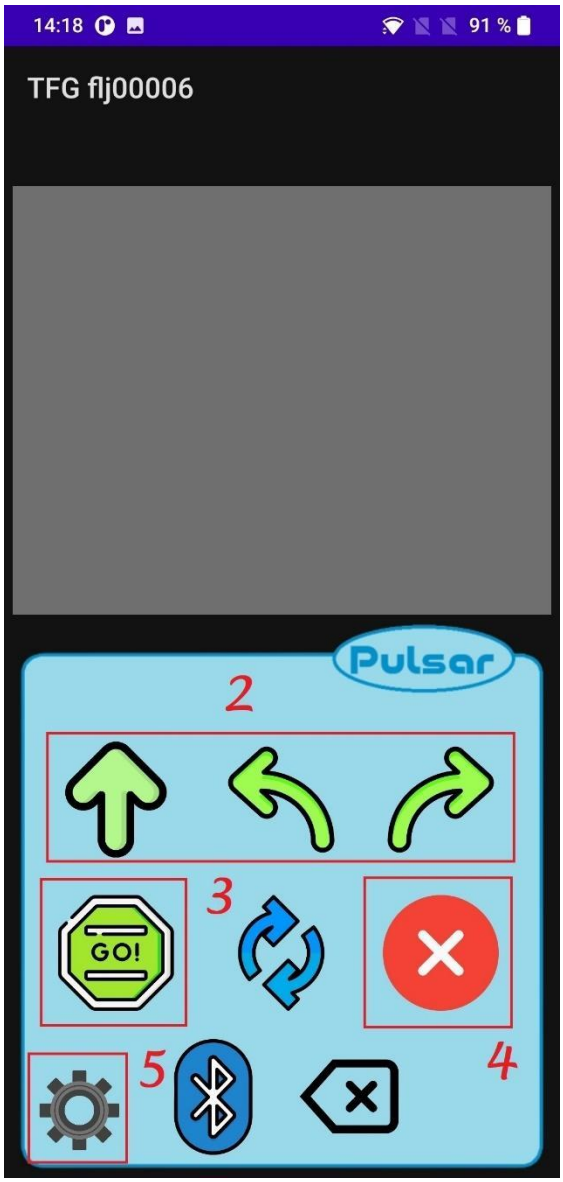


Ilustración 6.6: Manual de usuario parte 2

2. Botonera de los distintos movimientos.
3. Botón para empezar los movimientos almacenados en la lista superior.
4. Botón para eliminar la lista actual de movimientos y parar el coche en caso de que se estén ejecutando los movimientos.
5. Botón para abrir la pantalla de escaneo.

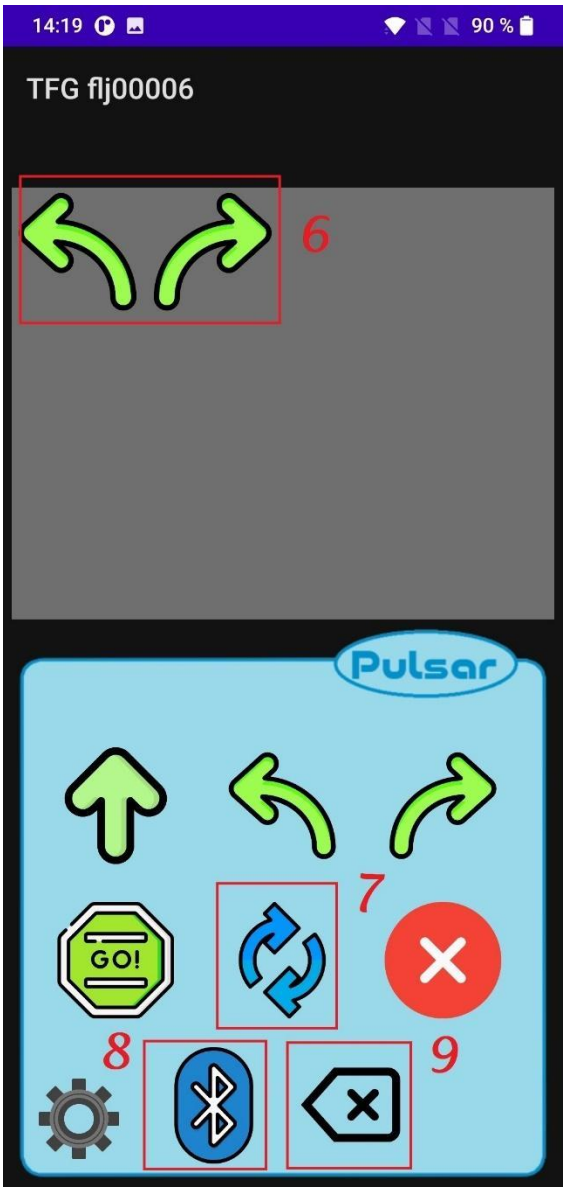


Ilustración 6.7: Manual de usuario parte 3

6. Secuencia de los movimientos seleccionados.
7. Botón para activar el modo bucle de la aplicación
8. Botón para desconectarse del dispositivo bluetooth y salir de la pantalla.
9. Botón para deshacer el último movimiento seleccionado en la app.

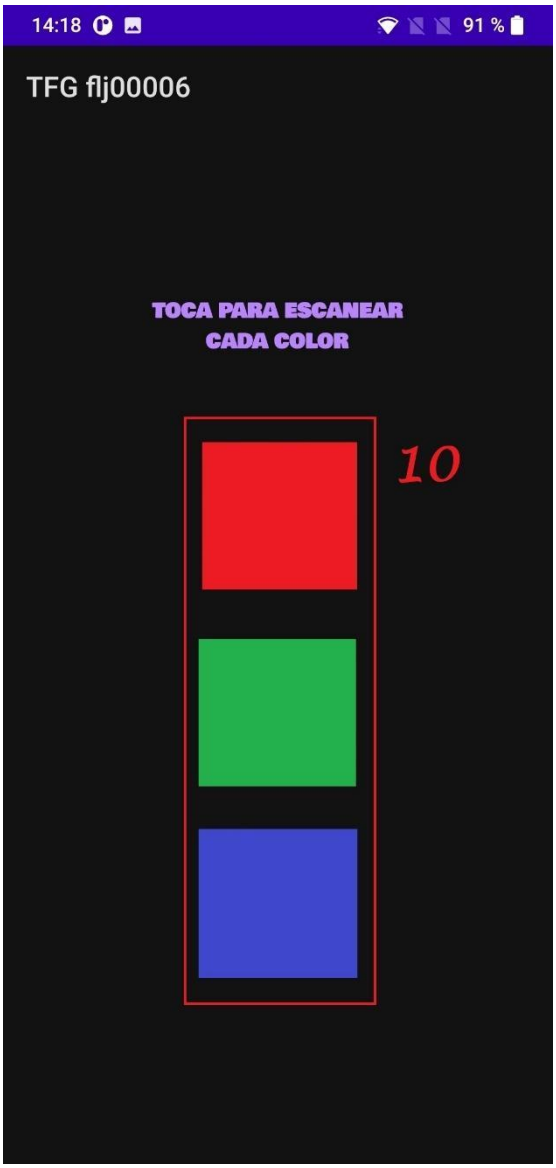


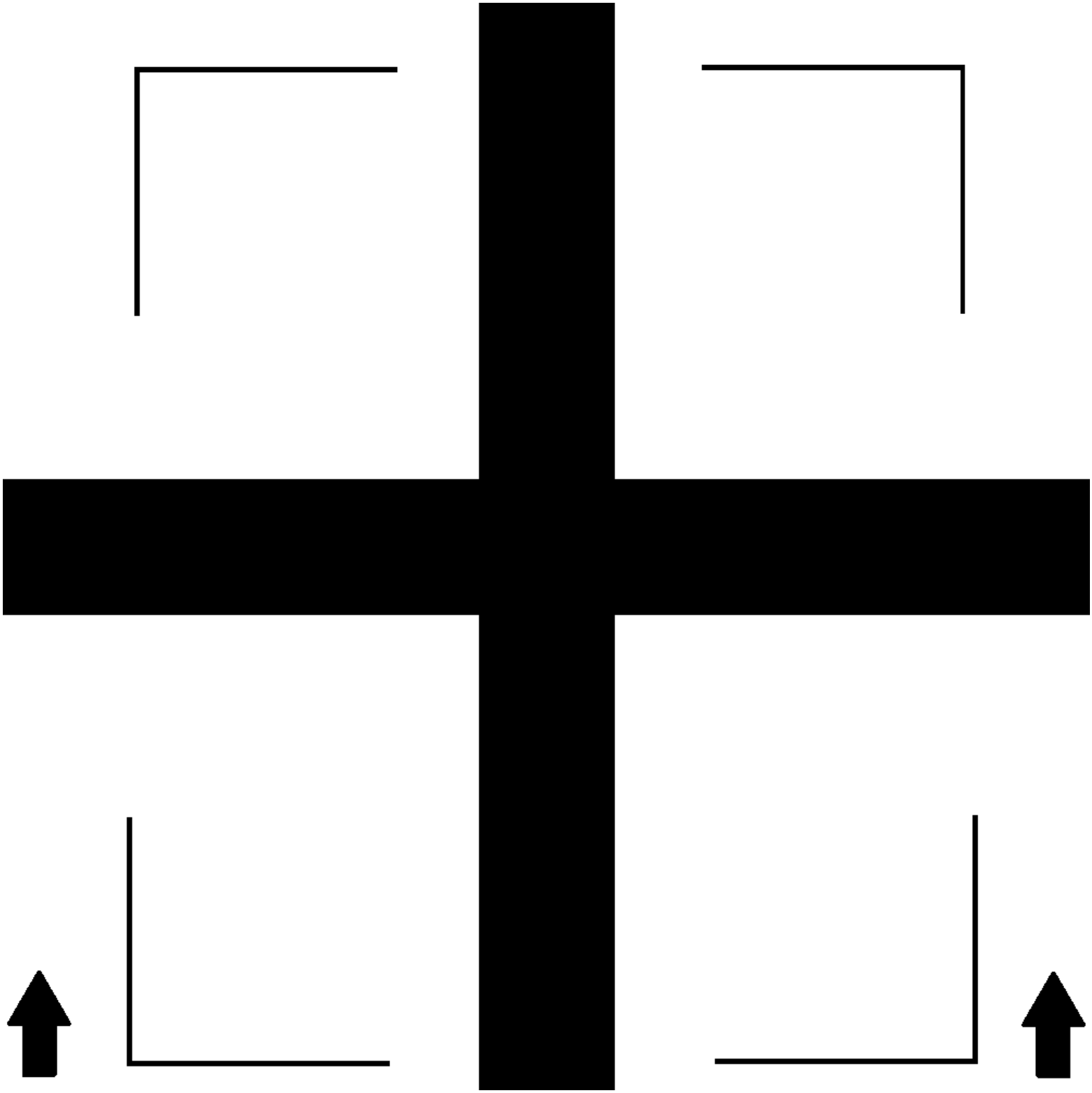
Ilustración 6.8: Manual de usuario parte 4

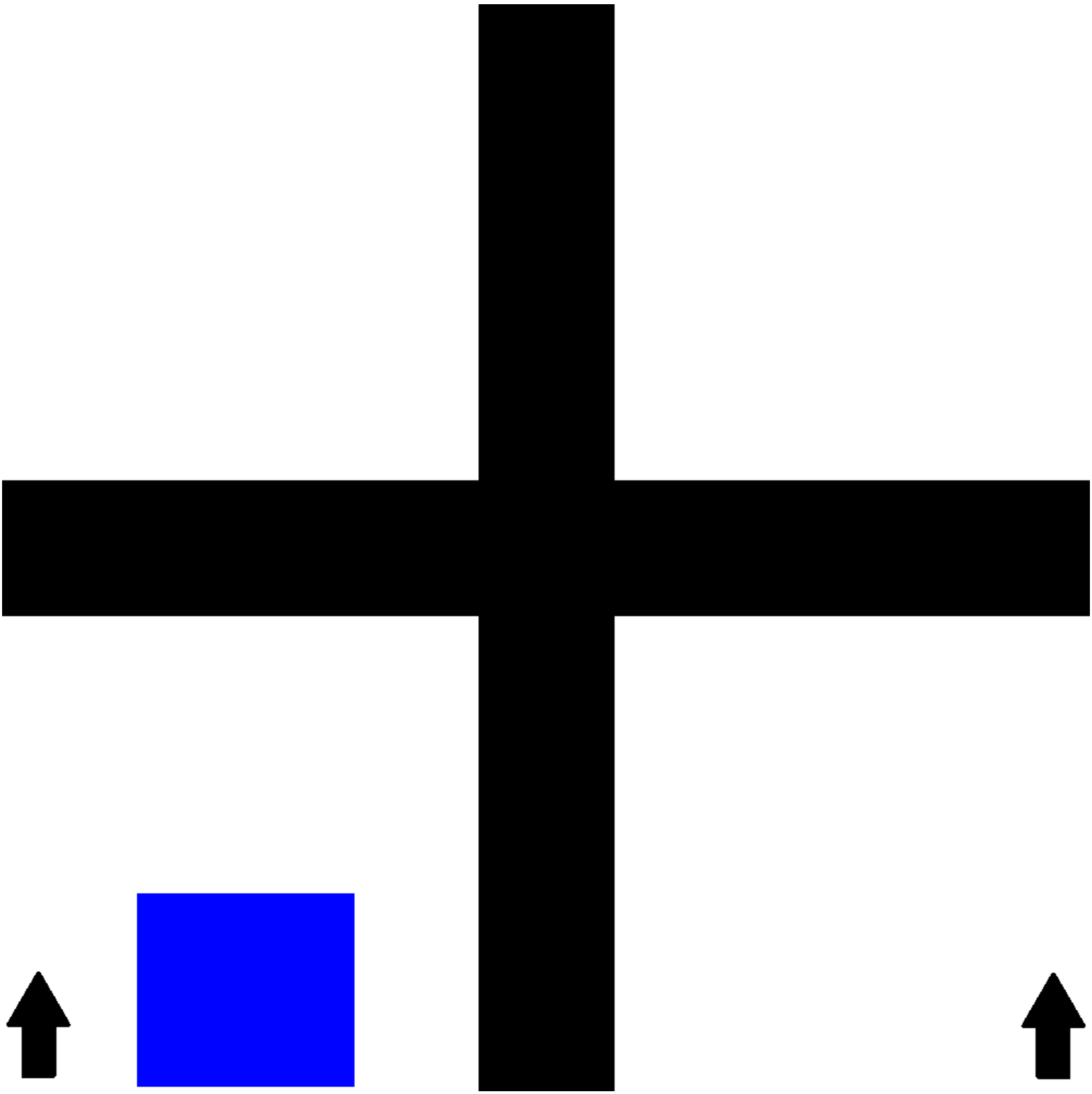
En la pantalla de escaneo tenemos lo siguiente:

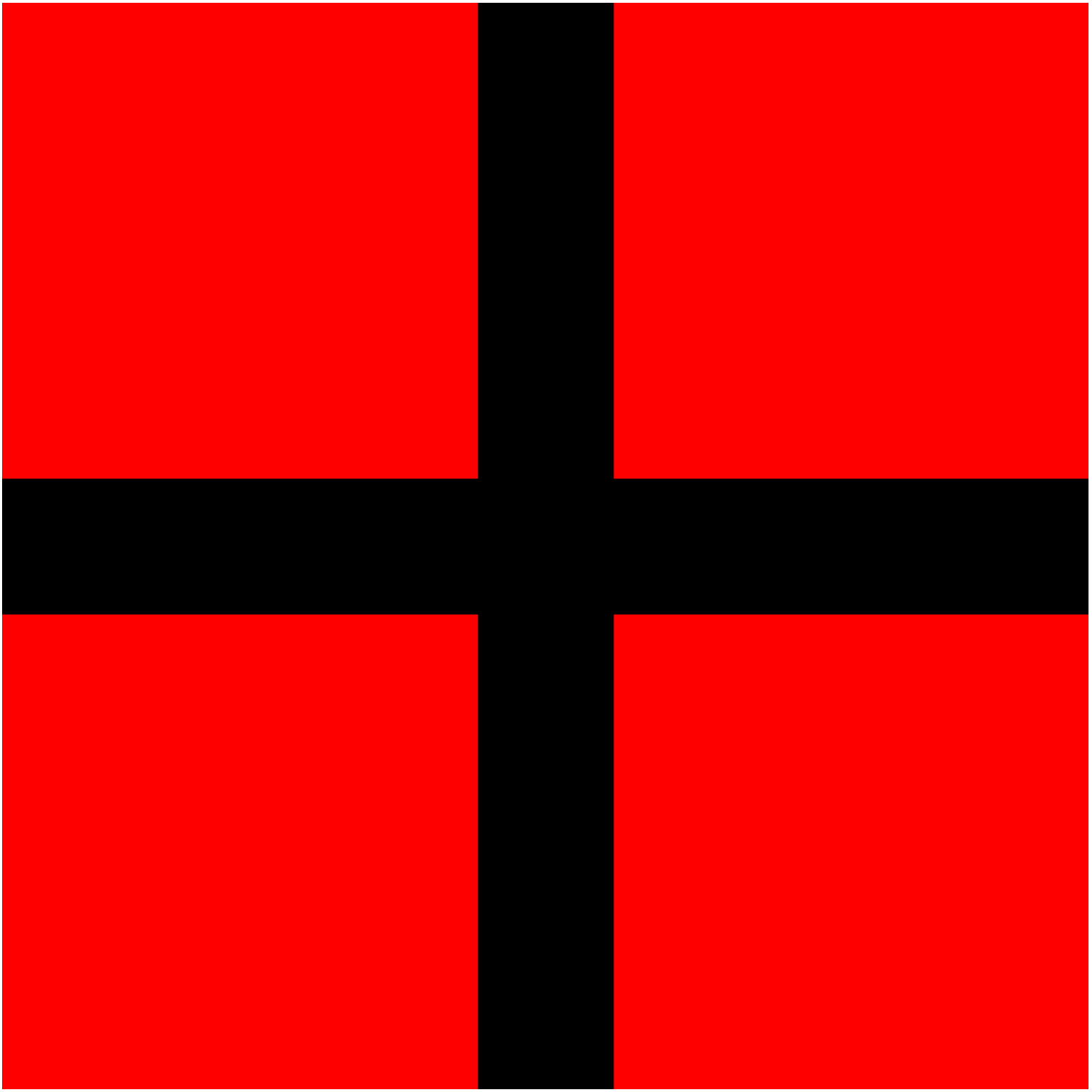
10. Botones para escanear los colores.

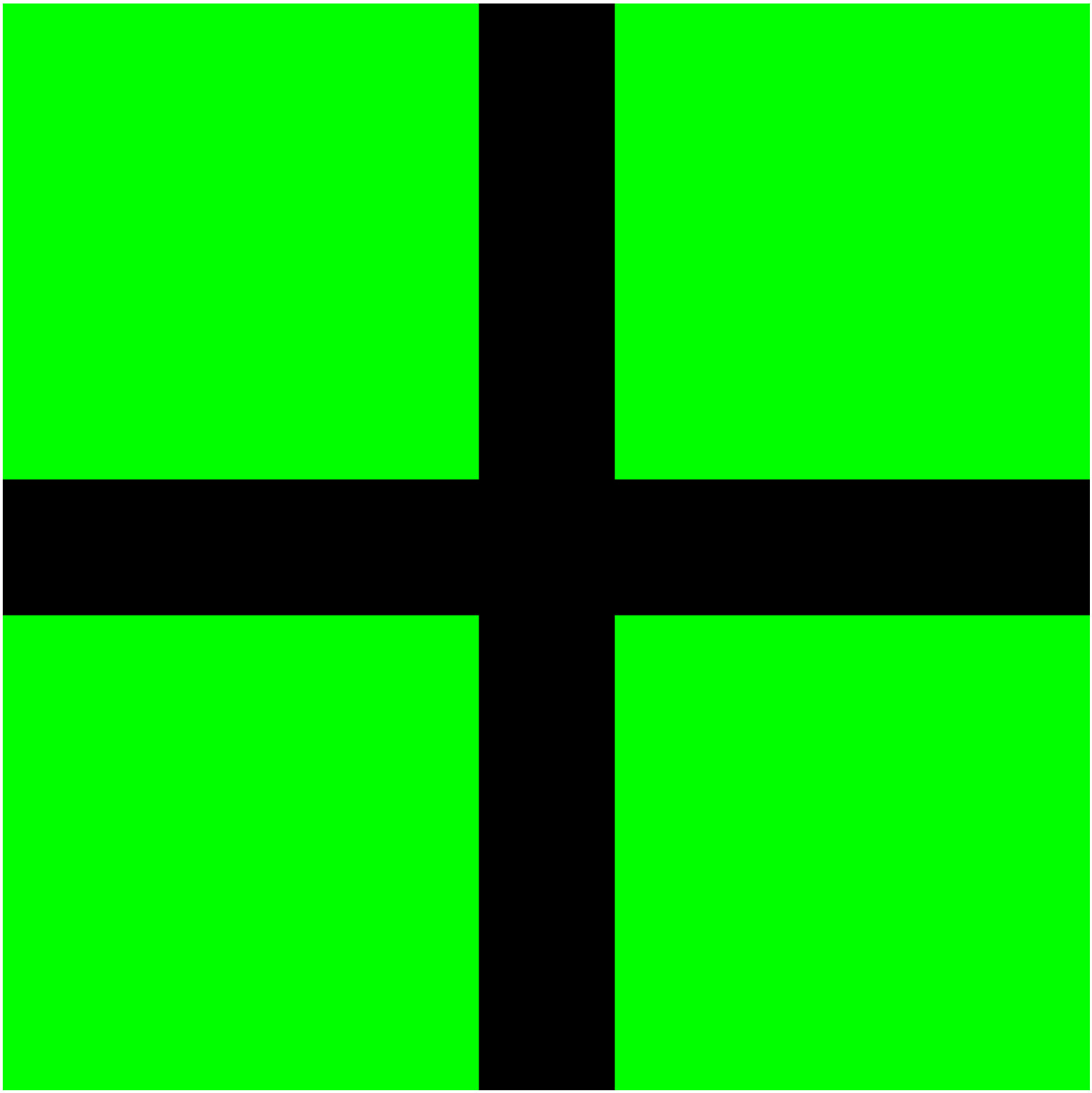
6.3 Plantillas para la creación del entorno

Se proporciona al usuario los diferentes diseños del entorno de nuestro prototipo para que pueda montar los circuitos que estime oportunos. Las plantillas están preparadas para ser utilizadas de manera impresa tal cual, recortadas y puestas sobre una superficie firme y robusta, sin imperfecciones donde el robot pueda sufrir movimientos bruscos









7 BIBLIOGRAFÍA

[1] Estos son los beneficios de aprender programación en la infancia. Educación 3.0. https://www.educaciontrespuntocero.com/noticias/beneficios-aprender-programacion-infancia/

[2] https://espiraleducativa.org/beneficios-de-aprender-programacion-en-la-infancia/

[3] 13 reasons why every parent should encourage their child to learn coding skills. FunTechBlog. https://funtech.co.uk/latest/13-reasons-why-every-parent-should-encourage-their-child-to-learn-coding-skills

[4] Evidencias científicas de los beneficios de aprender a programar desde infantil. Programamos. https://programamos.es/evidencias-cientificas-de-los-beneficios-de-aprender-a-programar-desde-infantil/

[5] EL JUEGO Y LAS NUEVAS TECNOLOGÍAS. Ana García-Valcárcel Muñoz-Repiso. Universidad de Salamanca http://www.quadernsdigitals.net/datos_web/articles/pixel/pixel13/p13juego.html

[6] Programación con LOGO. https://www3.gobiernodecanarias.org/medusa/ecoescuela/recursosdigitales/2015/02/11/programacion-con-logo/

[7] Logo (lenguaje de programación) https://es.wikipedia.org/wiki/Logo_(lenguaje_de_programaci%C3%B3n)

[8] La tortuga que nos enseñó a programar: la historia de Logo, el primer lenguaje de programación diseñado para niños. Enrique Pérez. 2018. https://www.xataka.com/historia-tecnologica/tortuga-que-nos-enseno-a-programar-historia-logo-primer-lenguaje-programacion-disenado-para-ninos

[9] Qué es Ionic: ventajas y desventajas de usarlo para desarrollar apps móviles híbridas. 2021 https://profile.es/blog/que-es-ionic/

[10] Android Studio: ventajas, desventajas y principales características, 2016. https://androidstudiofaqs.com/conceptos/ventajas-desventajas-android-studio

[11] ¿Por qué el desarrollo de aplicaciones móviles con Flutter? AbaMobile. https://abamobile.com/web/desarrollo-aplicaciones-flutter-caracteristicas-ventajas/

[12] Flutter: pros y contras del SDK open source de Google. 2019. https://sumatd.com/blog/flutter-pros-contras/

[13] Ionic Framework, ventajas y desventajas. 2016. https://openwebinars.net/blog/ionic-framework-ventajas-desventajas/

[14] Qué es Ionic y todas sus ventajas. Randy Valera. https://www.randyvarela.es/ionic-definicion-ventajas/

[15] ESP32 Series Datasheet https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[16] Arduino Uno. https://es.wikipedia.org/wiki/Arduino_Uno

[17] 5 alternativas increíbles al IDE de Arduino. https://descubrearduino.com/alternativas-ide-arduino/

[18] Arduino IDE entorno de desarrollo oficial. José Guerra Carmenate. https://programarfacil.com/blog/arduino-blog/arduino-ide/

[19] El mejor IDE para Arduino. Visual Studio con Visual Micro. https://www.luisllamas.es/el-mejor-ide-para-arduino-visual-studio-con-visual-micro/

[20] PROGRAMA DE CONTROL DEL COCHE. Movimiento del coche. Prometec. https://www.prometec.net/coche-programa-control/

[21] Cómo funciona la Metodología Scrum: Qué es y cómo utilizarla. Javier Sáez Hurtado. 2021. https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/

[22] Ventajas y desventajas importantes de la metodología Scrum. 2023. https://es.indeed.com/orientacion-laboral/desarrollo-profesional/ventajas-desventajas-metodologia-scrum

[23] http://www.burndowngenerator.com

[24] Moqups. https://moqups.com/es/

[25] StarUML. https://staruml.io/

[26] Averigua cuánto cobra un Ingeniero Informático. UAX. https://www.uax.com/blog/ingenieria/cuanto-cobra-un-ingeniero-informatico#:~:text=Un%20Ingeniero%20Inform%C3%A1tico%20reci%C3%A9n%20egresado,20.450%20euros%20brutos%20por%20a%C3%B1o

[27] ESP32 – PinOut. 2020. https://www.studiopieters.nl/esp32-pinout/

[28] How Does the TCS3200 Color Sensor Work and how to Interface it with Arduino?. 2020. https://circuitdigest.com/microcontroller-projects/interfacing-color-sensor-with-arduino

[29] Arduino bluetooth controller. Google Play. https://play.google.com/store/apps/details?id=com.giumig.apps.bluetoothserialmonitor&hl=es_VE

[30] ACTIVIDADES PARA TRABAJAR CON EL ROBOT EN EL AULA. curso 2020/2021. http://creecyl.centros.educa.jcyl.es/sitio/upload/PARA_COLGAR_EN_LA_WEB.pdf

[31] Lightbot. https://www.lightbot.lu/

[32] Scratch. https://scratch.mit.edu/

[33] Aliexpress. https://es.aliexpress.com/