JUEGO DE CONSTRUCCIÓN DE CIRCUITOS DE CARRERAS CON COCHE RADIOCONTROL BASADO EN ARDUINO

RACEWAY BUILDING GAME WITH REMOTE-CONTROLLED CAR BASED ON ARDUINO

Grado en Ingeniería del Software

Realizado por
Sergio Fernández Aguilar

Daniel Garrido Márquez
Cristian Martín Fernández

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, (Septiembre, 2021)

Agradecimientos

Antes de entrar en materia, me gustaría dar las gracias a una serie de personas que directa o indirectamente considero que han jugado un papel importante no solo durante la elaboración de este proyecto, sino durante todo el proceso educativo universitario.

Dar las gracias a mis tutores Daniel y Cristian por ofrecerme la oportunidad de realizar mi proyecto con ellos y guiarme en algunos de los puntos clave del trabajo. Sin sus ideas y propuestas estoy seguro de que el resultado de este proyecto habría sido bien distinto.

Gracias a mis amigos por sacarme una sonrisa en los momentos más difíciles, y a mi pareja por darme ánimos, apoyarme y estar siempre dispuesta a echar una mano.

Dar las gracias a mi abuela Antonia, quien sin pretenderlo ha logrado ser una fuente de inspiración y motivación constante que me ha empujado día a día a seguir adelante.

Por último, dar las gracis a mis familiares por todos los ánimos y la confianza depositada en mí. Gracias a mi hermano Rubén, mi tita Susana y mi primo Alberto. Muchas gracias a mi padre y mi madre por proporcionarme todos los recursos que han estado a su alcance, por su fe ciega y por todo el cariño y amor que me han dado.

Resumen

En este proyecto se detalla el desarrollo del prototipo de un juguete constituido por una aplicación Android y un coche radiocontrol hecho con Arduino, donde la aplicación móvil juega el papel de mando a distancia y a la vez de videojuego, proponiendo al jugador la construcción de pistas de carreras con materiales caseros para luego desafiarlo a marcar determinados tiempos por vuelta.

Tras un análisis de los productos con características similares que se encuentran actualmente en el mercado y un estudio de las tecnologías que intervienen en el proyecto, se diseñó e implementó la infraestructura básica del prototipo: circuitería y código del coche radiocontrol, esqueleto del videojuego junto con las configuraciones de control y el establecimiento de la comunicación Bluetooth entre ambos sistemas.

Se desarrolló un sistema de construcción guiada de circuitos que leyendo la información contenida en un fichero JSON es capaz de construir gráficamente un circuito y generar la secuencia de pasos para su construcción. Este sistema junto con los contadores de vueltas implementados usando la tecnología RFID (Identificación por Radiofrecuencia), constituyen el sistema de juego del prototipo.

El proceso de desarrollo, el cual ha ido acompañado por pruebas de usabilidad que han ayudado a identificar y corregir problemas de usabilidad serios, terminó con la construcción del prototipo del coche radiocontrol y la generación de contenidos para la aplicación.

El resultado obtenido ha sido el prototipo de un juguete que, aun estando lejos de ser un producto final, es totalmente funcional y cumple con los objetivos del proyecto.

Palabras clave

Arduino, Android, coche radiocontrol, videojuego para móvil.

Abstract

This project details the development of a toy prototype constituted by an Android application and an Arduino radio-controlled car, where the mobile application plays the role of the remote control and at the same time of a videogame, suggesting the player the construction of raceways with homemade materials to then challenge him to set specific lap times.

After an analysis of similar products that are currently in the market and a study of the technologies involved in the project, the basic infrastructure of the prototype was designed and implemented: circuitry and code of the radio-controlled car, the skeleton of the videogame together with the forms of control and the establishment of Bluetooth communication between both systems.

Then a guided raceway building system which is capable of building a circuit graphically and of generating the sequence of steps for its construction by reading the information contained in a JSON file, was developed. This system together with the lap counters implemented using RFID (Radio-frequency Identification) technology, constitute the game system of the prototype.

The development process, which has been accompanied by usability tests that have helped identify and correct severe usability problems, ended with the construction of the radio-controlled car prototype and the generation of content for the application.

The result obtained has been the prototype of a toy that, even though it is far from being a final product, is fully functional and fulfils the project's objectives.

Keywords

Arduino, Android, remote-controlled car, mobile's videogame.

Índice

Índice de figuras

Índice de abreviaturas

API: Application Programming Interfaces.
Código QR: Código Quick Response.
DIY: Do It Yourself.
EEPROM: Electrically Erasable Programmable Read-Only Memory.
IDE: Integrated Development Environment.
JSON: JavaScript Object Notation.
JVM: Java Virtual Machine.
LCD: Liquid-Crystal Display.
LED: Light Emitting Diode.
NFC: Near Field Communication.
P2P: Peer to Peer.
PWM. Pulse Width Modulation.
RAM: Random Access Memory.
RFID: Radio-frequency identification.
RGB: Red Green Blue.
RX: Recepción.
SPI: Serial Peripheral Interface.
TX: Transmisión.
UID: Unique ID.
XML: Extensible Markup Language.

1. Introducción

1.1 Motivación

En la actualidad existe un amplio abanico de posibilidades en cuanto a juguetes y juegos se refiere. Y es un hecho que la tecnología está cada vez más presente en ellos. Una buena inclusión de la tecnología en los juguetes provoca que sean más entretenidos, didácticos e interactivos para los más pequeños. Esto no significa que sean incapaces de divertirse sin recurrir a ella. Un caso muy específico es el de los cochecitos de juguete. Los niños juegan con ellos de mil formas distintas: juegan a lanzarlos por el pasillo a ver cuál llega más lejos, juegan a chocarlos unos con otros, simulan persecuciones policiales y un largo etcétera.

Con el propósito de estimular la creatividad y divertir a todo aquel que muestre interés por los coches de juguete y/o radiocontrol, se ha ideado un juguete compuesto por un coche radiocontrol y un videojuego para móviles que propone a los jugadores construir circuitos con materiales de la vida cotidiana, para luego correr en ellos manejando el coche desde la propia aplicación.

Actualmente, en el mercado existen productos que ofrecen características similares al prototipo propuesto, de ellos hablaremos en el apartado *2. Estudio del arte*. Aunque la mayoría de ellos tienen algo en común, y es que suelen ser caros y por consiguiente no son accesibles para todo el mundo. Con el objetivo de solventar esta limitación, se ha considerado que el desarrollo de una aplicación móvil en unos tiempos en los que más de la mitad de la población mundial dispone de un teléfono [1] (una estadística que va a continuar creciendo) junto con el empleo de materiales caseros puede resultar en un acierto en lo que oportunidades se refiere.

1.2 Objetivos

El propósito del trabajo es desarrollar el prototipo de un juguete para niños, llamado **Race your Track**, constituido por una aplicación móvil y un coche radiocontrol. Donde la aplicación móvil actuará como controlador del coche y como videojuego. De forma que el videojuego propone la construcción de circuitos o tramos al jugador con materiales caseros, para luego correr por esos trazados controlando el coche radiocontrol desde el propio videojuego.

Para alcanzar este objetivo será necesario cumplir con los siguientes:

- Construcción de un prototipo de coche radiocontrol manejable desde una aplicación móvil mediante Bluetooth.
- Diseñar un sistema de construcción guiada de trazados, acompañado de un pequeño kit de construcción hecho con materiales caseros.
- Implementar un sistema de desafíos sencillo y amistoso que integre las características de construcción y conducción del juguete.
- Alcanzar un grado de usabilidad alto en la aplicación móvil respaldado por pruebas de usabilidad llevadas a cabo durante el desarrollo del producto.

Puesto que el resultado del trabajo será un prototipo, no entran dentro de los objetivos de este proyecto:

- Construir un coche de radiocontrol que satisfaga características imprescindibles en un juguete para niños tales como lo son la resistencia a golpes, durabilidad de la batería, atractivo del diseño o seguridad.
- Realizar un estudio de mercado con el fin de orientar el desarrollo del producto a campañas publicitarias o a satisfacer las necesidades del cliente objetivo.

1.3 Estructura del resto del documento

La estructura de la memoria es la siguiente:

Apartado 1. Introducción. Contextualización del lector sobre cuál es la idea y los objetivos que persigue el proyecto.

Apartado 2. Estudio del arte. Análisis sobre productos del mercado que presentan características similares a las del proyecto, así como descripción de las tecnologías utilizadas en el proyecto.

Apartado 3. Metodología de trabajo. Descripción de la metodología y decisiones de diseño adoptadas durante el desarrollo del trabajo.

Apartado 4. Desarrollo de la infraestructura básica. Descripción paso a paso del proceso de desarrollo de las características básicas del producto, distinguiendo entre las características del coche, de la aplicación y la conexión entre ambos.

Apartado 5. Sistema de construcción guiada de circuitos. Exposición del sistema de construcción implementado en el videojuego, detallando la forma en que se crean las pistas de carreras y cómo se presentan en pantalla al jugador.

Apartado 6. Contadores de vueltas. Implementación del sistema que permite al juego contabilizar y cronometrar las vueltas del jugador por los circuitos haciendo uso de la tecnología RFID.

Apartado 7. Sistema de juego y desafíos. Presentación del sistema de niveles del videojuego acompañada por la explicación de las decisiones tomadas para implementar el sistema en la aplicación.

Apartado 8. Contenidos de la aplicación y prototipo final. Exposición de cuáles son los contenidos necesarios para que el videojuego no esté vacío y cómo se han creado. Así como breve descripción del proceso de construcción del prototipo del vehículo.

Apartado 9. Conclusiones y líneas futuras. Valoración sobre el grado de cumplimiento de los objetivos del proyecto junto con la enumeración de posibles trabajos futuros derivados de él.

Bibliografía. Enumeración de las fuentes consultadas durante toda la vida del proyecto, tanto para la redacción de este documento como para el desarrollo del sistema.

Anexos. Conjunto de diagramas y modelos de distintas características del sistema como la circuitería del coche radiocontrol o diagrama de clases del sistema de construcción guiada de circuitos. Documentos usados en las pruebas de usabilidad y otros.

2. Estudio del arte

En este apartado, se analizan y comparan productos del mercado que presentan características similares a las del prototipo. Además, se describirán las tecnologías y componentes electrónicos que participan en el proyecto.

2.1 Mercado actual

En el mercado actual, existen productos que ofrecen experiencias similares a la que se trata de conseguir con el resultado del proyecto. A continuación, se enumeran los que se consideran más interesantes y/o que se asemejan más, así como los que han servido de inspiración.

2.1.1 Juguetes de construcción de circuitos para niños

Cuando se dice juguetes de construcción de circuitos para niños, se refiere a esos pequeños kits formados por piezas de madera o raíles fáciles de encajar que están dirigidos a niños menores de siete años. Dentro de esta "categoría", es difícil escoger a un solo producto representante, pues todos tienen pequeños matices. Por ejemplo, algunos están preparados para que una vez el circuito esté montado, el coche circule solo por los raíles, en otros es necesario que el niño/a lo empuje.

Ilustración 1. Juguete de Magic Tracks
Fuente: https://www.toysrus.es/Magic-Tracks-Rescue/p/K129717

Aunque no se duda que sean divertidos o favorables para la creatividad de los niños, están bastante limitados, la mayoría de ellos no recurren a la tecnología y además están enfocados a un público de un rango de edades muy estrecho.

2.1.2 Scalextric

Aunque coloquialmente se conoce por el nombre de *Scalextric*, su nombre real es *slot*. *Scalextric* es simplemente la marca más famosa de *slot*, aunque existen otras como *Carrera*, de hecho esta empresa lanzó en 2017 un producto muy novedoso en el que interviene una aplicación que propone desafíos a los jugadores. El *slot* está muy extendido por el mundo, existiendo incluso competiciones. Además, en cuanto a las posibilidades de construcción de circuitos, es el rey del mercado. Pues oferta piezas de todas las formas y colores.

El *slot* hace uso de la corriente eléctrica para impulsar los motores de corriente continua que tienen los coches. Las carreteras tienen unos raíles conductores que son alimentados por un transformador enchufado a la corriente (los hay que usan baterías). Los jugadores tienen un mando con el que controlan cuanta corriente fluye por los raíles, y el coche tiene unas escobillas de cobre con las que transmite corriente al motor. De esta forma, el jugador puede controlar la velocidad a la que gira el motor, logrando una jugabilidad muy divertida. [3]

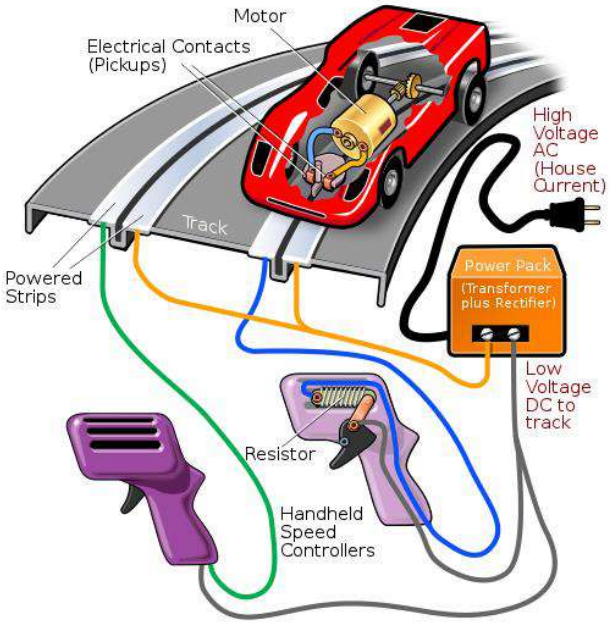


Ilustración 2. Funcionamiento de un coche slot
Fuente: https://www.wikiwand.com/en/Slot_car

La cantidad de vehículos de *slot*, así como las posibilidades de expansión de los circuitos, son casi infinitas. Aunque a diferencia de lo que busca este proyecto, todo lo relacionado con el mundo del *slot* es caro, lo que hace que no esté al alcance de todo el público.

Ilustración 3. Coches de slot sobre una pista
Fuente: https://www.wikiwand.com/en/Slot_car

2.1.3 Apps para controlar un RC Arduino

El mundo Arduino y DIY (Do It Yourself) está muy extendido. Existen proyectos muy creativos hechos por personas inquietas, como un sistema de goteo automático, manos robóticas o cajas expendedoras. El mundo del radiocontrol no iba a ser menos, de hecho los proyectos de vehículos radiocontrol son muy frecuentes. Es por esta razón por la que existen numerosas aplicaciones que permiten manejar un vehículo radiocontrol hecho con Arduino.

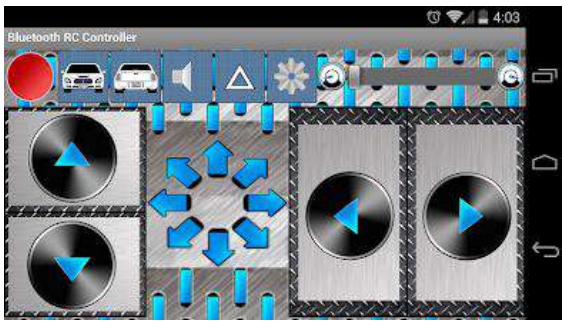


Ilustración 4. Interfaz de Bluetooth RC Controller
Fuente: https://play.google.com/store/apps/details?id=braulio.calle.bluetoothRCcontroller&hl=es&gl=US

La principal diferencia entre estas aplicaciones y Race your Track, es que Race your Track funciona como videojuego y no solo como un mando a distancia.

2.1.4 Drift Hybrid Gaming

El Drift Hybrid Gaming (también conocido como DR!FT) es un producto relativamente joven, pues solo lleva en el mercado dos años y aún está en alza. Propone encadenar derrapes manejando un coche de escala 1:43 desde una aplicación móvil.

El poder hacer derrapar el vehículo radiocontrol fácilmente hace gala de lo divertido y preciso que es el manejo, así como de la novedosa tecnología que incorpora. Los coches tienen seis ruedas, dos de las cuales se encuentran en el centro del chasis y en lugar de en un eje, están en una base giratoria que es la que permite al coche derrapar y deslizarse horizontalmente.

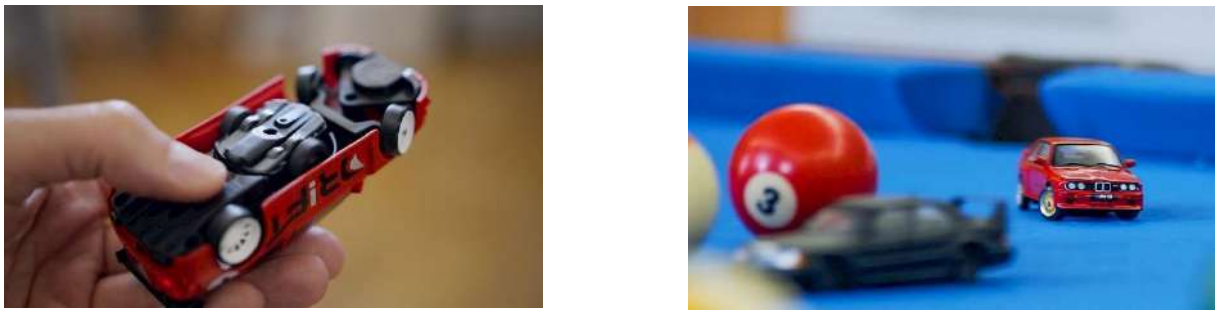


Ilustración 5. Coches de Drift Hybrid Gaming
Fuente: https://www.sturmkind.com/es/

La aplicación móvil ofrece un interfaz de control que transmite sensaciones, la cual ha inspirado a Race your Track. Además, la aplicación no solo sirve de mando, sino que también propone desafíos y muestra puntuaciones.

Ilustración 6. Interfaz de control de Drift Hybrid Gaming
Fuente: https://www.sturmkind.com/es/

El Drift Hybrid Gaming ofrece una conducción muy realista e incitan a sus jugadores a crear pistas sobre las que correr, aunque no de forma guiada por estar dirigido a un público más adulto. Toda una fuente de inspiración para Race your Track. [4]

2.1.5 Hot Wheels A.I. Intelligent Race System

Hot Wheels es una marca referente en el mundo de coches a escala, famosa por una amplia variedad de modelos y espectaculares pistas de carreras. El Hot Wheels A.I. Intelligent Race System es sin duda alguna el que menos difiere con Race your Track. Pues ofrece una experiencia de construcción de circuitos sobre la que luego correr con un coche radiocontrol, el cual mediante I.A., es capaz de autoconducirse por el circuito mediante sensores. De esta forma, el jugador podrá competir con otros amigos o contra la I.A., otorgándole una valiosa versatilidad al producto. [5]

Ilustración 7. Hot Wheels A.I. Intelligent Race System
Fuente: https://play.hotwheels.com/es-es/ai.html

Aunque no es un producto caro para todo lo que ofrece, e implementa características que este proyecto no aborda (resistencia de los vehículos, uso de I.A…), sí que Race your Track al usar una aplicación, abre un sinfín de puertas a oportunidades de mejora e inclusión de nuevas ideas. Haciéndolo así un producto más escalable.

2.2 Tecnologías y componentes electrónicos

Al tratarse de un proyecto en el que se comunican un sistema móvil y un coche radiocontrol, se usan numerosas tecnologías y componentes electrónicos. Se describirán todos ellos a continuación.

2.2.1 Arduino

El proyecto Arduino surgió en Italia en el año 2003 de la mano de un grupo de estudiantes con la intención de facilitar el acceso a la electrónica y la programación a los estudiantes que no podían permitirse otras alternativas.

Arduino es una plataforma de creación de electrónica de hardware y código abierto. Esto permite a los fabricantes diseñar sus propias placas tomando como base el hardware abierto. Arduino es una plataforma de prototipado, por tanto no está pensado para productos comerciales, sino para prototipos y maquetado, debido a que sus especificaciones no se ajustan a las necesarias en un producto final. *[6]*

Las placas Arduino están basadas en el microcontrolador ATMEL y dependiendo de la placa dispondrá de más o menos memoria, distintos accesorios como botones, diodos LED, altavoces… En cualquier caso, las placas tienen un conjunto de pines mediante los cuales se pueden conectar distintos periféricos como motores, módulos de conexión, sensores… Sobre los periféricos se hablará un poco más adelante, pero respecto a los pines que ofrece, aunque depende de la placa en concreto, los más habituales son:

- Pines digitales: Son pines de entrada/salida usados para comunicarse con sensores y actuadores. Dentro de los pines digitales se encuentran los pines de comunicación TX y RX usados para las comunicaciones. En algunas placas, se encuentran pines compatibles con las señales PWM.
- Pines analógicos: Son pines exclusivamente de entrada usados con sensores que proporcionan mediciones con relativa precisión y sensibilidad.
- Pines de energía: Son los pines usados para alimentar el Arduino o los distintos periféricos que se encuentren conectados a la placa. Se ofrecen dos salidas, una de 5V y otra de 3’3V (los voltajes de trabajo más frecuentes entre los componentes compatibles) así como un pin de tierra para cerrar los circuitos.

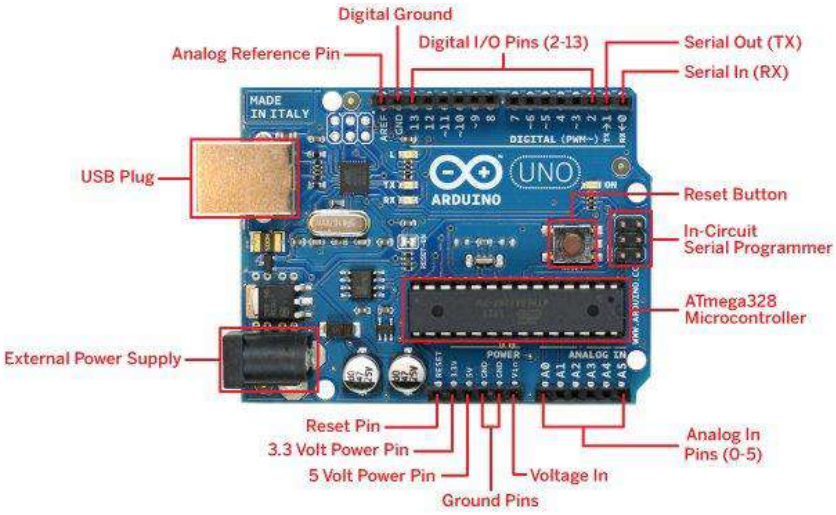


Ilustración 8. Mapa de pines de la placa Arduino UNO
Fuente: https://aprendiendoarduino.wordpress.com/2016/06/27/arduino-uno-a-fondo-mapa-de-pines-2/

Aunque es variable según el tipo de placa, la memoria suele estar muy limitada. Por lo que como en cualquier sistema empotrado es vital prestar atención al consumo de memoria de los programas.

Arduino, proporciona su propio IDE con el que es posible compilar y subir a la placa los programas Arduino. Arduino tiene su propio lenguaje de programación, aunque es muy similar a C. Todo programa Arduino, llamado *sketch*, está formado por dos funciones:

- setup(): función en la que se inicializa la ejecución del programa. Se ejecuta una sola vez y se hace nada más encender la placa o tras un reset.
- loop(): función que se ejecuta indefinidamente justo después del setup.

Gracias a que Arduino es de software libre, existe una amplia comunidad que ofrece numerosas librerías de código abierto que permiten interactuar con todo tipo de periféricos. Así como una gran cantidad de información en la red en forma de videotutoriales, ejemplos de código y proyectos completos (comunidad DIY), entradas en foros…

2.2.2 Sensores y actuadores

La lista de componentes electrónicos capaces de interactuar con una placa Arduino es prácticamente interminable. Existen sensores y actuadores de todas las clases y tamaños, permitiendo a la comunidad Arduino dar rienda suelta a su imaginación y desarrollar proyectos muy creativos como máquinas expendedoras, sistemas de goteo, manos robóticas, radares, domótica, etc.

Los sensores son capaces de medir características del entorno y transformar el valor medido en una señal digital que puede ser enviada a un Arduino. Una vez recibido, ese valor puede ser traducido e interpretado mediante un software desarrollado según las especificaciones que establece la ficha técnica del componente. Existen sensores de movimiento, temperatura, presión, humedad, pulsadores…

Los actuadores permiten a la placa Arduino comunicarse con el exterior. Esto puede ir desde el accionamiento de algún mecanismo a la emisión de un sonido o la impresión de información. Algunos ejemplos de actuadores son motores, pantallas LCD, altavoces…

Además, existen muchos módulos (sensores/actuadores más sofisticados) y controladores idóneos para tareas más específicas como los protocolos de comunicación y trabajos más complejos como llevar el control de varios motores CC o imprimir sobre una matriz de LEDs. [8]

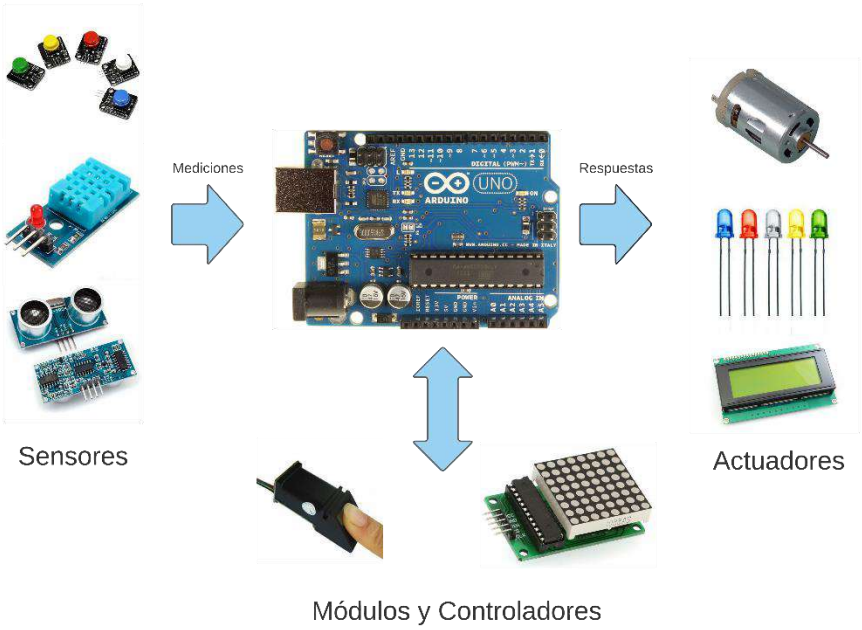


Ilustración 9. Algunos sensores y actuadores compatibles con un controlador Arduino

Por los requisitos del prototipo, no se incluye ningún sensor en su sistema, aunque en líneas futuras se plantea añadir un sensor de luminosidad con el que simular el sistema de encendido automático de luces de un vehículo. Sin embargo, sí que intervienen diversos actuadores, un controlador L298N y dos módulos (sobre los que se hablará un poco más adelante en este mismo apartado).

Los actuadores que participan en el proyecto son:

- Diodo LED: Un diodo es un componente electrónico con dos conexiones, ánodo y cátodo, que solo deja pasar la corriente en uno de sus sentidos (del ánodo al cátodo), en ese caso se dice que está polarizado directamente. [9]
  Un diodo LED es un diodo que cuando deja pasar la corriente emite luz.

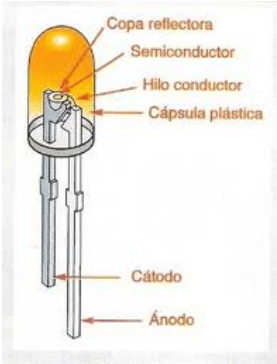


Ilustración 10. Partes de un diodo LED
Fuente: https://www.areatecnologia.com/electronica/como-es-un-led.html

  La ventaja de los LEDs frente a las bombillas convencionales es la durabilidad y el consumo.
- Motor CC: los motores de corriente continua tienen dos terminales de conexión y producen movimiento cuando circula corriente por ellos. Están formados por dos partes: estator (parte inductora donde se produce el campo magnético) y un rotor (contiene el bobinado, el eje y las escobillas).

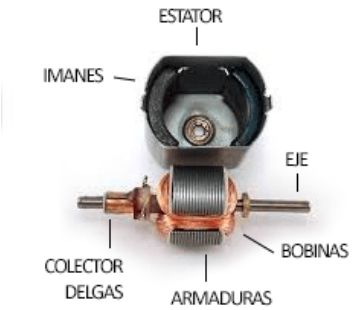


Ilustración 11. Partes de un motor CC
Fuente: https://www.luisllamas.es/tipos-motores-rotativos-proyectos-arduino/

El movimiento rotatorio del eje se genera cuando circula corriente por el bobinado del rotor, que debido al campo magnético producido por los imanes del estator se produce una fuerza de Lorentz. El eje del motor continuará girando mientras exista un paso de corriente eléctrica por su bobinado, y podrá girar en la dirección opuesta invirtiendo la polarización. [10]

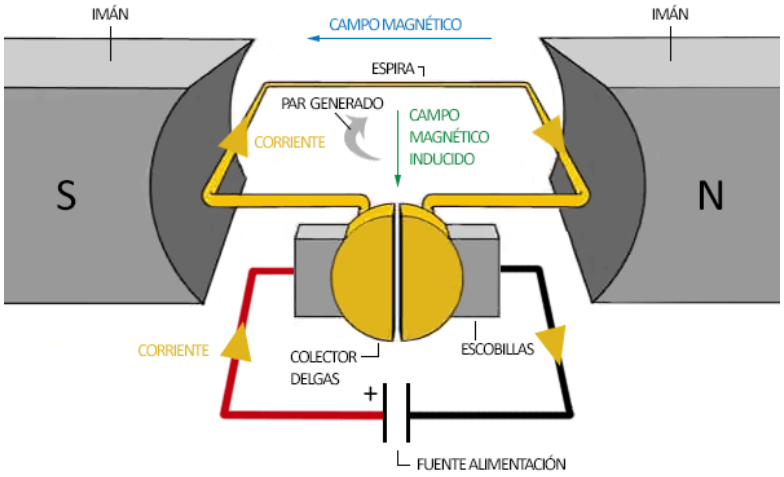


Ilustración 12. Esquema de funcionamiento de un motor CC
Fuente: https://www.luisllamas.es/tipos-motores-rotativos-proyectos-arduino/

- Servomotor: un servomotor es un motor de precisión capaz de girar un número determinado de grados y mantenerse en esa posición. Está formado por un controlador, un motor de corriente continua y un tren de engranajes.

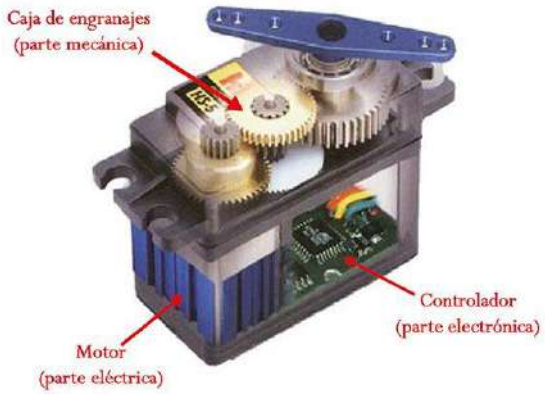


Ilustración 13. Partes de un servomotor
Fuente: http://panamahitek.com/que-es-y-como-funciona-un-servomotor/

El controlador se comunica con el Arduino y envía las señales eléctricas al motor de corriente continua enganchado al tren de engranajes. El tren de engranajes es quien permite realizar movimientos con precisión. [11]

El módulo L298N proporciona un interfaz que permite controlar dos motores de corriente continua o un motor paso a paso de forma sencilla. Contiene un controlador L298 IC con dos puentes H que es capaz de emitir señales PWM para controlar la velocidad de giro de los motores e incluso admite una alimentación externa lo que le permite trabajar con motores de más de 5V (tensión máxima de salida de un controlador Arduino). [12]

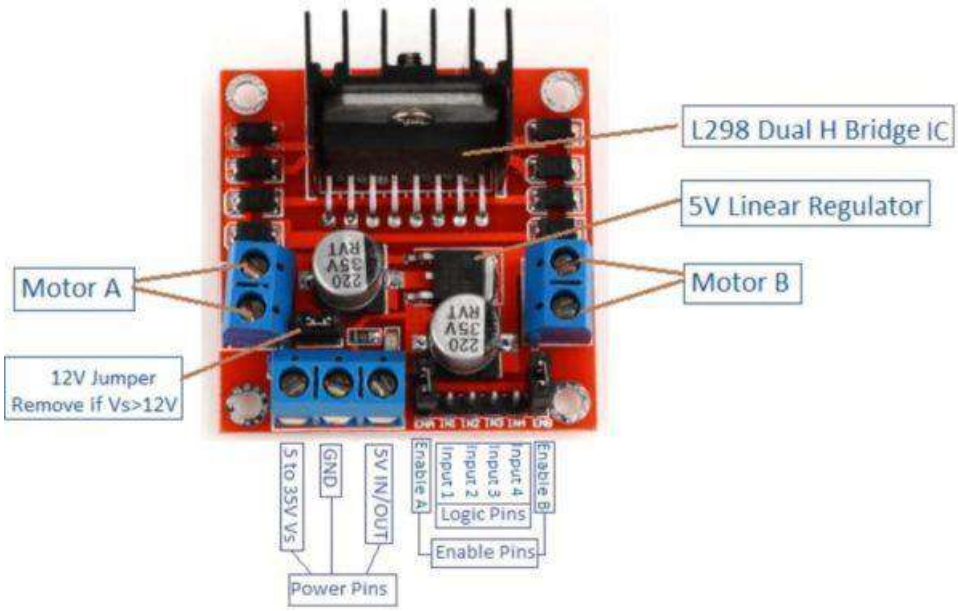


Ilustración 14. Partes del módulo L298N
Fuente: https://descubrearduino.com/modulo-l298n/

Por último, las placas Arduino pueden ser alimentadas de diversas formas. Por ejemplo, durante el desarrollo se suele alimentar mediante un cable USB, otra forma es suministrar 5V estables y controlados al pin 5V de la placa. En este proyecto, como consecuencia de las restricciones inalámbricas, se ha alimentado al controlador por su clavija Jack usando una batería de 9V incrustada en un portapilas con interruptor. [13]

Ilustración 15. Portapilas con switch y conector Jack para baterías de 9V
Fuente: https://suconel.com/product/caja-de-portapila-para-bateria-9v-con-suiche-y-plug-pp9vsp/

2.2.3 Android

Android es un sistema operativo pensado para dispositivos móviles, una colección de aplicaciones preinstaladas y un *framework* para aplicaciones. A día de hoy se encuentra en la mayoría de los dispositivos del mercado, contando con el 86'1% de los dispositivos móviles distribuidos en el año 2019 (perteneciendo el resto a iOS) [14]. Obviamente sigue en mantenimiento y constante evolución con revisiones y actualizaciones cada 6 meses.

El sistema operativo Android está basado en Linux, y cuenta con ciertas características especiales derivadas de los requisitos específicos de los dispositivos móviles. Algunas de estas características son realizar una mejora en la eficiencia del uso de la batería, restringir los permisos de los procesos para aumentar la seguridad, llevar una gestión de memoria *Low Memory Killer*… El *Low Memory Killer* es un mecanismo que permite a una app obtener memoria antes de que se quede sin ella eliminando de la RAM procesos según su nivel de prioridad.

La plataforma Android proporciona un sinfín de artículos y documentación oficial sobre su API, además de un *framework* de aplicaciones con el que los desarrolladores pueden crear apps e instalarlas. El *framework*, aparte de definir la forma en la que se programa una aplicación Android, ofrece numerosos componentes visuales y utilidades que agilizan el trabajo de los desarrolladores.

Android tiene su propio IDE (Android Studio), cargado de herramientas y facilidades para los desarrolladores. Tales como, facilidad para añadir múltiples idiomas a la aplicación, opciones de accesibilidad y configuraciones para distintos tamaños de pantalla. Android se puede programar en dos lenguajes: Java y Kotlin. Kotlin es el más joven y no busca reemplazar a Java, sino complementarlo, prueba de ello es que corre también sobre una JVM.

Ilustración 16. Logos de Android, el IDE Android Studio y el lenguaje de programación Kotlin

Abordar todo lo que esconde la programación Android escapa a los propósitos de este proyecto, sin embargo conviene familiarizarse con alguna terminología fundamental concreta de la programación Android. Los proyectos Android están formados esencialmente por tres elementos:

- Manifiesto: fichero XML que incluye la información esencial de la aplicación tales como permisos, dependencias…
- Actividades y Fragmentos: Las actividades son los componentes más importantes de la programación Android. Los usuarios interactúan con las actividades para completar tareas y navegan por ellas para desplazarse por la aplicación. Todas las actividades deben de estar declaradas en el manifiesto y siguen el mismo ciclo de vida, el cual viene marcado por la interacción del usuario con la aplicación.

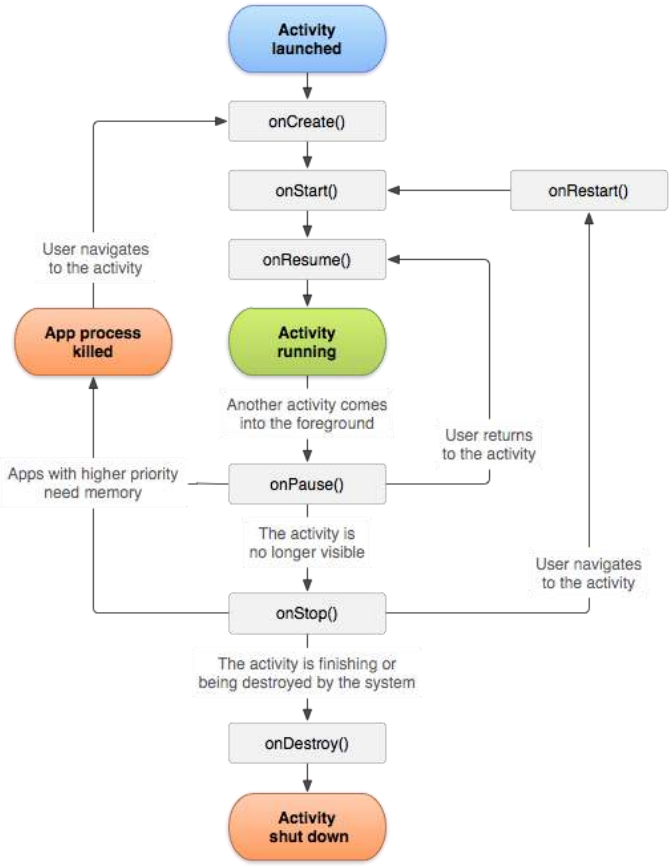


Ilustración 17. Ciclo de vida simplificado de una actividad Android
Fuente: https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es

En la versión 3.0 de Android se incluyeron los Fragmentos. Estos se deben de incluir dentro de las actividades y se usan como elemento de navegación

dentro de una misma actividad o elemento reutilizable entre distintas actividades.

Para tareas con alta carga computacional que no necesitan de la interacción con el usuario, no se deben usar las actividades. En su lugar se usan los Servicios, cuyo ciclo de vida es muy similar al de las Actividades solo que se ejecutan en segundo plano.

- Recursos: directorio en el que se localizan todos los recursos que se necesitan en la aplicación. Como recursos podemos encontrar estilos, listas de constantes, ficheros multimedia… Dentro de los recursos, destacar la definición de los *layouts*, los cuales especifican la disposición que los componentes visuales deben adoptar en pantalla para cada vista de la aplicación.

2.2.4 Comunicación Bluetooth

El estándar IEEE 802.15.1, más conocido como Bluetooth, es una tecnología LAN inalámbrica de área personal de corto alcance que permite realizar conexiones espontáneas con un rango aproximado de hasta 10m (según la clase). A pesar de parecer que vive a la sombra del protocolo Wifi, hoy en día está presente en todo tipo de dispositivos como teléfonos, impresoras, auriculares, sensores… Bluetooth lo constituye una arquitectura en capas, distinta a la de internet, formada por distintos protocolos para cada nivel. Además en los niveles superiores se ofrecen distintos perfiles según el propósito del servicio: transferencia de archivos, de audio, control de llamadas…

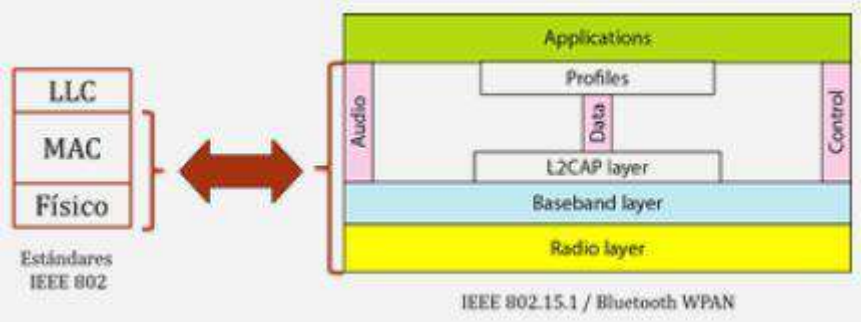


Ilustración 18. Arquitectura de capas IEEE 802.15.1
Fuente: Universidad de Málaga, Redes y Sistemas Distribuidos, Gabriel Luque

Explicar los protocolos que se implementan en todos y cada uno de los niveles de la arquitectura, no entra dentro de los propósitos del proyecto. Sin embargo, sí que interesa conocer que Bluetooth implementa de forma inalámbrica el modelo de comunicación maestro-esclavo. En él, existe un maestro encargado de iniciar y administrar la comunicación con uno o más esclavos que se encontrarán a la espera de que el maestro establezca la conexión. En el proceso de comunicación, será el maestro quien envíe peticiones a los esclavos, y estos se limitarán a responderlas. Es un protocolo muy usado, por ejemplo en SGBD (Sistemas Gestores de Base de Datos) o en comunicaciones cliente-servidor (maestro y esclavo respectivamente). *[22]*

Ilustración 19. Comunicación maestro-esclavo
Fuente: https://www.uaeh.edu.mx/scige/boletin/tizayuca/n5/p7.html

Como adicional, aunque irrelevante para este proyecto, un protocolo de comunicación alámbrico que aparece frecuentemente en proyectos Arduino es el bus I2C. Lo hace porque permite entablar comunicaciones entre un maestro y varios esclavos usando solo tres conexiones. Es un protocolo de comunicación muy usado para comunicar distintas partes de un mismo circuito electrónico. *[23]*

La placa Arduino, a diferencia de los dispositivos móviles, no incorpora el protocolo de comunicación Bluetooth. Por eso, es necesario conectarlo a un módulo externo que sí lo tenga. En el mercado hay numerosas opciones, aunque las más populares son los módulos HC-05 y HC-06. Ambos son muy similares en cuanto al esquema de conexión, precio y rendimiento. La principal diferencia es que el módulo HC-05 puede ser configurado como maestro o esclavo, mientras que el módulo HC-06 solo como esclavo.

Para los requerimientos de este proyecto, con un módulo HC-06 hubiera sido suficiente, sin embargo se ha usado un módulo HC-05 por disponibilidad del material. Estos módulos son configurables mediante comandos AT, pudiendo cambiar el modo de trabajo, el nombre, el pin de conexión… En el apartado *3.3 Comunicación Bluetooth*, se presenta un ejemplo de configuración con comandos AT. Respecto al esquema de montaje, los módulos son alimentados con 5V y para la conexión con el Arduino solo se necesitan dos cables, uno que conecta la transmisión del Arduino con la recepción del módulo y otro que conecta la transmisión del módulo con la recepción del Arduino. *[24]*

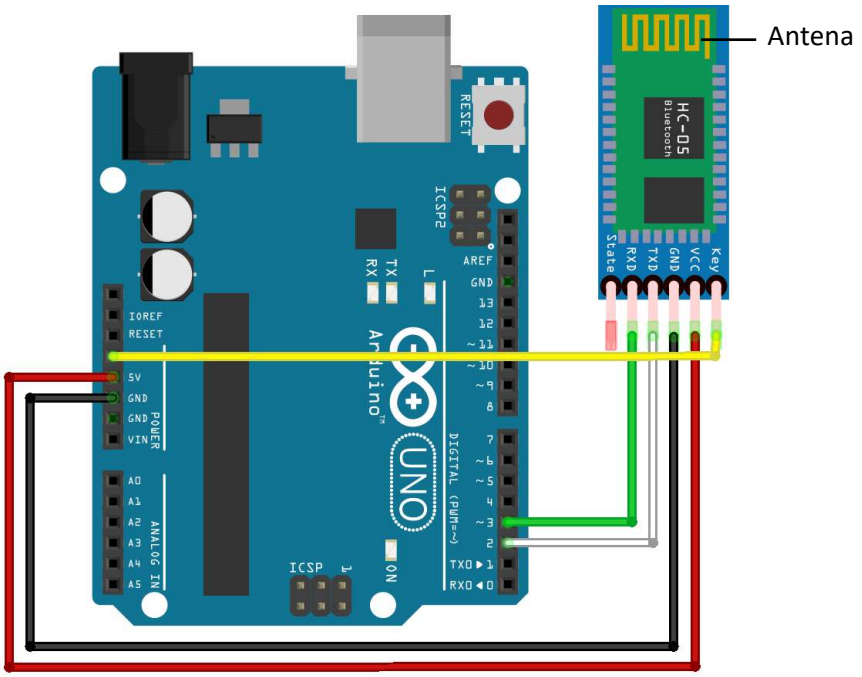


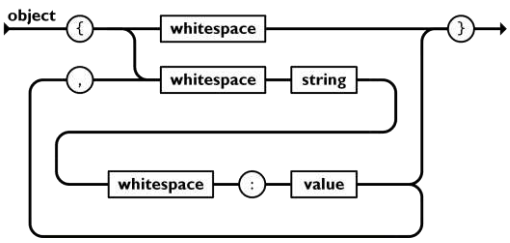
Ilustración 20. Esquema de conexión Arduino – HC-05
Fuente: https://www.aranacorp.com/es/tu-arduino-se-comunica-con-el-modulo-hc-05/

2.2.5 JSON, Notación de Objetos de JavaScript

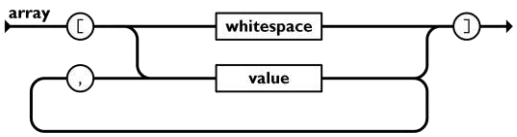
JSON es un formato de texto sencillo para el intercambio de datos que surgió como alternativa a XML (eXtensible Markup Language) porque llegaba a ser lento en las lecturas de grandes cantidades de datos. A día de hoy, JSON es una tecnología referente y es usada por grandes empresas como Google, Yahoo! y Mozilla gracias a que, al estar basado en la sintaxis de JavaScript, permite a los navegadores leer/escribir muchos ficheros de datos con efectividad. Un ejemplo de uso: los datos con información meteorológica que se envían desde una API a un sitio web suelen ir en formato JSON.

A pesar de que su sintaxis es semejante a la de JavaScript, JSON es un lenguaje independiente y su estructura tiene como base tres elementos:

- Objetos: conjunto de pares nombre/valor. Los pares se separan por comas y el objeto se encierra entre llaves { }.



- Colecciones: conjunto de valores separados por comas. Las colecciones se encierran entre corchetes [ ].



- Valores: un valor puede ser un número, una cadena de caracteres, un objeto e incluso una colección.

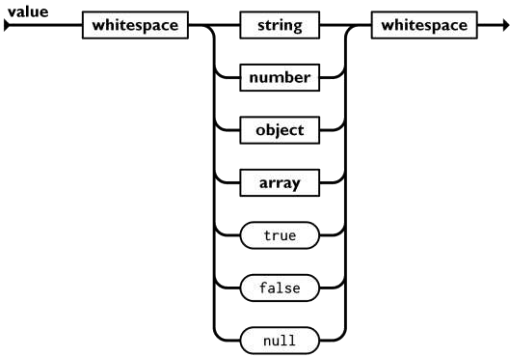


Ilustración 21. Especificaciones gráficas formales de las estructuras JSON

Fuente: https://www.json.org/json-es.html

Gracias a estas tres estructuras es posible diseñar conjuntos de datos complejos que son sencillos de leer para los humanos y rápidos de interpretar por los ordenadores. *[30]*

Existen multitud de librerías para trabajar con ficheros de texto JSON y XML para los lenguajes de programación más populares. Estas librerías permiten leer y escribir ficheros JSON, o XML respectivamente, con relativa facilidad y eficiencia.

Para este proyecto, se requiere de un lenguaje para el intercambio de datos con el fin de transformar pequeños ficheros de datos en un objeto Java (deserialización). En un principio se pensó en usar una librería Java para el tratamiento de XML, llamada JAXB. Con ella es posible transformar la información de un objeto Java en un fichero XML, y viceversa, gracias a un *Marshaller*. El problema es que la librería no está incluida en Android de forma nativa por lo que se buscó una alternativa. La elección fue GSON, una librería desarrollada por Google específica para tratar ficheros JSON que viene nativamente en Android. Con ella es posible transformar el contenido de un JSON en un objeto Java fácilmente, siendo necesario solamente realizar unas anotaciones obre la clase Java en cuestión. *[31][32]*

2.2.6 Tecnología RFID

La tecnología RFID forma parte del estándar NFC (Near Field Communication). NFC es un protocolo de conectividad inalámbrica de corto alcance que mediante la inducción magnética permite establecer redes P2P (Peer To Peer) para el intercambio de datos. Es una tecnología muy usada en controles de acceso y métodos de pago. Su presencia, al igual que la de los códigos QR, se ha visto en aumento como consecuencia de la pandemia del COVID19, ya que debido a la necesidad de evitar el contacto, han transformado tareas como pedir la carta de un restaurante o pagar la cuenta.

La comunicación RFID la componen dos elementos: un lector conectado a un computador y un tag con un circuito integrado que habitualmente se encuentra incrustado junto con una antena en una tarjeta, un llavero o un adhesivo. La comunicación la inicia el lector emitiendo una onda por radiofrecuencia de 13,56 MHz. El tag tiene una antena que aprovecha esa onda para generar pequeños impulsos eléctricos que alimentan al circuito integrado encargado de modular la onda que regresará al lector. Por último, el lector demodula la onda y obtiene el UID del tag.

Un UID es un código formado por 8 bytes que identifica inequívocamente al tag. Para satisfacer la necesidad de este proyecto, ha bastado con leer el UID del circuito integrado usando un lector RFID-RC522, pero mencionar que el circuito es capaz de almacenar y transmitir más información gracias la memoria EEPROM que contiene.

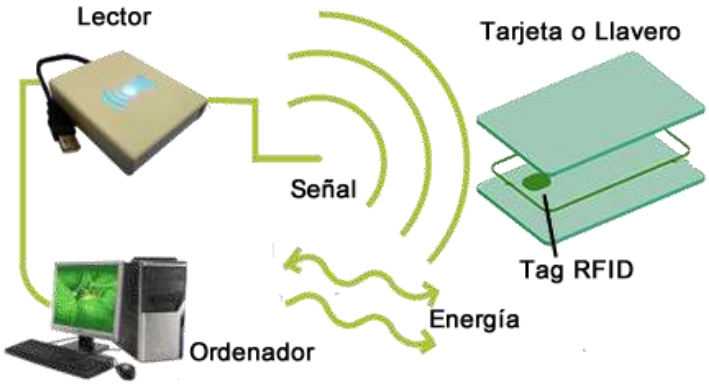


Ilustración 22. Esquema de comunicación RFID
Fuente: https://www.softwarecannabis.es/blog/tarjetas-contacto-rfid-identificacion-socio

El lector RFID-RC522 trabaja a 3,3V. En su circuito integrado contiene una antena encargada de emitir y recibir las ondas de radio, que serán moduladas y demoduladas por el microcontrolador MFRC522. Este microcontrolador se comunica con el exterior mediante el protocolo SPI, que es compatible con Arduino. *[33]*

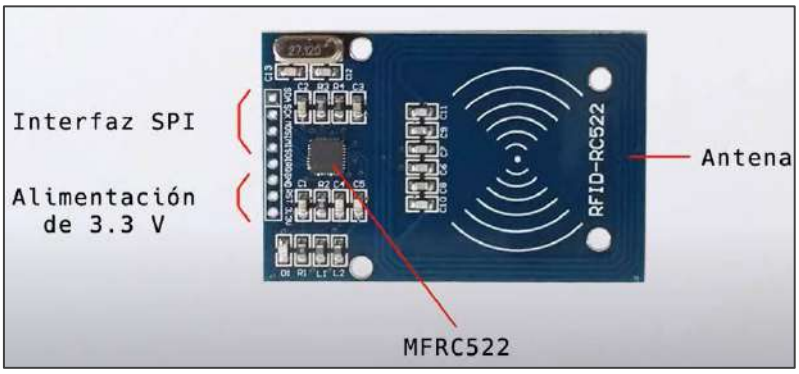


Ilustración 23. Módulo RFID-RC522
Fuente: https://www.youtube.com/watch?v=LvRfxGTUEpE

3 Metodología de trabajo

Para el desarrollo del proyecto se ha aplicado una metodología incremental descompuesta en un total de cinco incrementos. A pesar de que toda la vida del proyecto se sustenta de una planificación y un análisis inicial, en cada incremento se han realizado tareas de planificación, recogida de requisitos, modelado, diseño, desarrollo y pruebas del sistema.

Tras una recogida de los requisitos más importantes del prototipo y analizar las dependencias que presentan, se planificó la forma en que se iban a desarrollar las funcionalidades del sistema dividiendo el proceso en cinco fases y se estimó el tiempo necesario para cada una de ellas.

Fase	Estimación	Dependencias
1. Desarrollo de la infraestructura básica	95	-
1.1 Desarrollo de la infraestructura básica del vehículo	45	-
1.2 Desarrollo de la infraestructura básica de la aplicación	40	-
1.3 Comunicación inalámbrica entre los sistemas	10	1.1, 1.2
2. Implementación del sistema de construcción guiada de circuitos	54	1.2
3. Construcción de contadores de vueltas	20	1
4. Desarrollo del sistema de juego	50	2, 3
5. Creación de contenidos de la aplicación y construcción del prototipo final	17	2, 4

Ilustración 24. Tabla con la planificación de las fases del proyecto

Además, el proceso de desarrollo ha estado marcado por tres hitos establecidos por la verificación del sistema mediante pruebas de usabilidad. Las cuales generarían resultados a tener en cuenta en el siguiente incremento.

En cuanto al modelado, diseño y aplicación de patrones, en cada incremento se han generado modelos e identificado patrones de diseño que podrá encontrar explicados en el apartado correspondiente a cada fase.

Respecto a la forma de trabajar, se ha usado la aplicación web Trello para organizar las tareas a realizar en cada fase. Además, con la intención de llevar un histórico del software desarrollado, se ha usado Git como sistema de control de versiones. Estos sistemas proporcionan numerosas ventajas, que aunque son más notorias cuando se trabaja en equipo o en proyectos de grandes dimensiones, resultan muy útiles para el desarrollo de cualquier producto sin realmente importar las circunstancias. Como repositorio en lugar de una red local se ha recurrido a los servicios de GitHub. Los siguientes enlaces le dirigen a los repositorios relacionados con este proyecto:

https://github.com/sergio2297/RaceYourTrack
https://github.com/sergio2297/arduino_rc_car

4. Desarrollo de la infraestructura básica

La infraestructura básica del prototipo marcará el resto del desarrollo, por lo que ser muy cuidadoso y construir una estructura escalable y mantenible ante posibles errores, es imprescindible. Como consecuencia, esta fase del proyecto ha sido la más extensa, tal y como se planificó. Debido a su extensión, esta fase se ha dividido en tres subfases, más una de pruebas. Todas y cada una de ellas serán descritas en detalle unas líneas más abajo.

Para poner en situación, al terminar la fase, se obtendrá la circuitería y el código de un coche radiocontrol hecho con Arduino manejable desde una aplicación Android mediante Bluetooth.

4.1 Infraestructura básica del coche radiocontrol

Como ya se ha mencionado, la construcción de los cimientos del sistema marcará el resto de la vida del proyecto. Es por eso que para desarrollar la infraestructura básica del coche radiocontrol se realizó un análisis previo con un nivel de detalle notorio. Acciones como recoger los requisitos y límites del sistema, valorar qué componentes electrónicos usar en el prototipo del coche, y diseñar una estructura software robusta y abierta al cambio, son algunos ejemplos de las tareas ejecutadas en esta fase.

El vehículo está compuesto por tres sistemas: iluminación, dirección y transmisión. Estos tres sistemas funcionan independientemente entre sí, y es por esto que se ha seguido un proceso iterativo, donde en cada iteración se ha desarrollado un sistema. Entendiéndose por sistema, los componentes hardware y el software que lo forman.

Antes de hablar de los detalles y peculiaridades de cada sistema, conviene explicar a grandes rasgos cómo funciona el coche. El funcionamiento del sketch Arduino es simple. Tras inicializar los distintos sistemas, se ejecutará un bucle indefinidamente. Donde cada iteración consiste en:

1. Leer, si lo hay, un comando desde el serial.
2. Descomponer ese comando en acciones.
3. Añadir cada acción a la cola de su sistema correspondiente.
4. Ejecutar la rutina de cada sistema.

De este modo, se consigue encapsular el comportamiento de los sistemas y desacoplarlos de la implementación concreta del sketch, lo cual incita a la reutilización de código y favorece un desarrollo iterativo.

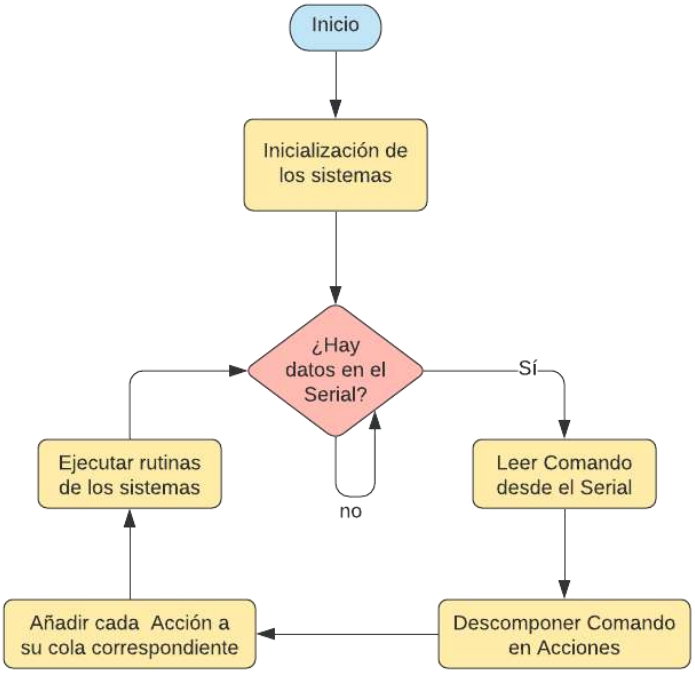


Ilustración 25. Diagrama de flujo del Sketch Arduino

Para entender mejor la rutina que realiza el vehículo, así como las explicaciones venideras, conviene familiarizarse con los siguientes términos:

- Acción: cadena de caracteres formada por una letra que identifica al sistema al que la acción va dirigida, seguida de un guion bajo más otro carácter que indica la acción a realizar.
- Comando: cadena de caracteres constituida por una o más acciones separadas por punto y coma, donde ninguna de las acciones va dirigida al mismo sistema.

(Todas las acciones se han definido como constantes del tipo carácter (1 byte). Se valoró definirlas como cadenas de caracteres (tipo String), pero por el hecho de trabajar en un sistema empotrado con memoria muy limitada se optó por usar variables del tipo char, aunque esto supusiera un esfuerzo inicial mayor)

A pesar de que cada sistema es independiente del resto, todos presentan un comportamiento prácticamente idéntico. Cada sistema referencia a una cola, la cual almacena las acciones que recibe desde el sketch principal, y ejecuta una secuencia de acciones periódicamente en función de su frecuencia particular, determinada por su prioridad (por ejemplo, el sistema de transmisión es más crítico que el de iluminación, por tanto su periodo es menor). La secuencia de acciones ejecutada es la siguiente:

1. Extraer una acción de su cola.
2. Interpretar la acción.
3. Actuar según el significado de la acción. Ej.: encender un LED.

Para la implementación de los sistemas, lo idóneo hubiera sido lanzar cada sistema en su propio hilo. Pero debido a que el entorno Arduino es monohilo, se ha recurrido al uso de máquinas de estado y ciclos de trabajo. En este caso, cada sistema ejecutará una vez por periodo su tarea rutinaria. Adicionalmente, se añadió a la funcionalidad de cada sistema la posibilidad de ejecutar un pequeño método fuera de su ciclo de trabajo, con la restricción de que el método no suponga un coste computacional alto, para evitar bloquear el procesador.

Respecto a la parte hardware del prototipo radiocontrol, recordar que tanto la placa como los componentes que forman cada uno de los sistemas, ya han sido descritos en el apartado *2. Estudio del arte*. En este punto se hablará sobre cómo se han usado esos componentes y en qué subsistema se han incluido.

Tras esta introducción al funcionamiento básico del sistema, se hablará sobre los procesos de desarrollo particulares de cada subsistema. El orden de exposición se corresponde con el orden de desarrollo, el cual viene establecido según la prioridad de cada sistema.

4.1.1 Sistema de transmisión

El sistema de transmisión es el encargado administrar las acciones relacionadas con hacer girar el motor de corriente continua. Dicho así parece algo simple pero realmente controla más detalles, como la caja de cambios. Buscando hacer más divertida la conducción del coche, se añadió la posibilidad de que el coche trabajara con distintas marchas, donde, a diferencia de en el mundo real, la marcha simplemente define la velocidad con la que gira el motor, siendo posible salir con facilidad de parado en quinta marcha.

Para poder indicar al motor que gire a una velocidad u otra se hace uso del envío de señales PWM. Las señales PWM (Pulse Width Modulation) modularizan el ancho del pulso, es decir, permite indicar qué porcentaje del periodo, la señal se encontrará a nivel alto (5V) y cuánto tiempo a nivel bajo (0V), es lo que se conoce como ciclo de trabajo. Las señales PWM son muy usadas en electrónica, por ejemplo para variar la potencia con la que calienta una estufa, a mayor tiempo del periodo a nivel alto, más corriente fluirá por las resistencias y más calor disiparán. *[19]*

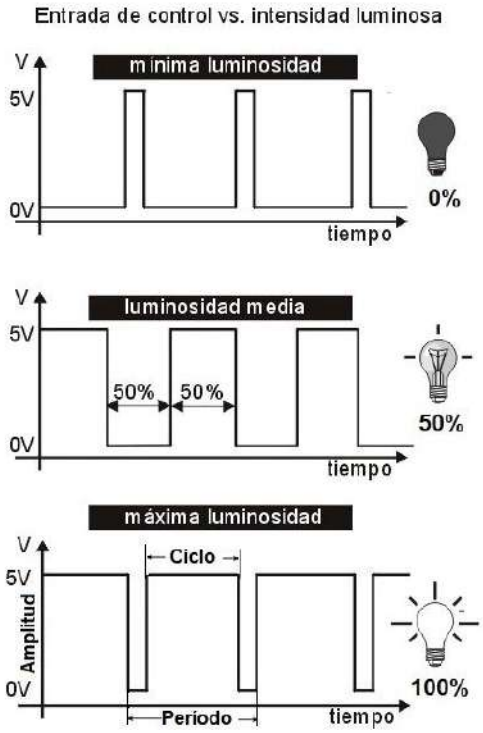


Ilustración 26. Cómo afecta una señal PWM a la luminosidad de una bombilla
Fuente: https://www.shoptronica.com/curiosidades-tutoriales-y-gadgets/4517-que-es-pwm-y-como-funciona-0689593953254.html

Inicialmente se pretendía construir el vehículo con un motor de 5V alimentado desde el propio Arduino, pero tras hacer un montaje de pruebas en una maqueta se observó que el motor no tenía el torque suficiente como para mover las ruedas del coche en marchas bajas. Fue en ese momento, en el que se decidió acudir a un motor más grande, de 12V. La inclusión de este motor desencadenó tener que usar el controlador L298N junto con una alimentación propia de 14,4V para el motor (12 pilas AAA recargables de 1,2V cada una). Aunque el controlador consume 3V, por lo que al motor realmente le llegan 11,4V.

Por último, el sistema de transmisión tiene un método adicional que se ejecuta fuera de su periodo. En él, si se detecta que se ha cambiado de marcha, se recalcula la señal PWM que se le debe enviar al controlador.

4.1.2 Sistema de dirección

El sistema de dirección controla un servo motor, y las acciones que puede realizar son girar a la izquierda, a la derecha y volver a la posición central. Es habitual en proyectos de esta índole, implementar la dirección haciendo uso de cuatro motores de corriente continua asignados a cada rueda del vehículo y que al girar en sentidos opuestos consiguen hacer que el coche gire. Desde un principio se huyó de esta idea por tratar de darle un toque de realismo a la conducción del radio control.

Para entender el cómo y el por qué se ha implementado la dirección de la forma en que se ha hecho, se debe saber que el servo motor necesita de un mínimo *delay* en su controlador para alcanzar su posición objetivo. Esto es una limitación bastante grande, puesto que nuestro sistema sería incapaz de recibir o ejecutar más acciones mientras se esté girando el servo, ya que se debería dormir al procesador. Para solventar esto, se ha recurrido al método que se ejecuta fuera de periodo.

El sistema de dirección almacena el número de grados en el que se debe encontrar, de este modo las acciones nunca giran el servo, sino que alteran ese valor. Será el método fuera de periodo el que gire el motor un grado por iteración, hasta alcanzar la posición objetivo. Esta solución presenta dos ventajas:

- Posibilidad de cambiar la dirección de giro sin que se haya terminado de alcanzar el antiguo número de grados objetivo.
- No bloquear el procesador.

4.1.3 Sistema de iluminación

El sistema de iluminación es el de menor prioridad y además el más simple, pues solo se encarga de encender y apagar luces LED. Entre las posibles acciones, se encuentra encender las luces cortas, largas, de freno, marcha atrás y ambos intermitentes.

Para las luces largas y de freno, se ha vuelto a usar señales PWM, consiguiendo así con un mismo LED mostrar distintas intensidades de luminosidad.

Para terminar, destacar que no sería hasta la fase *7. Contenidos de la aplicación y prototipo final* cuando se construiría el prototipo del vehículo. Hasta entonces, se trabajó siempre sobre la placa de pruebas y maquetas, ya que sobre ellas es más sencillo realizar modificaciones y validarlas.

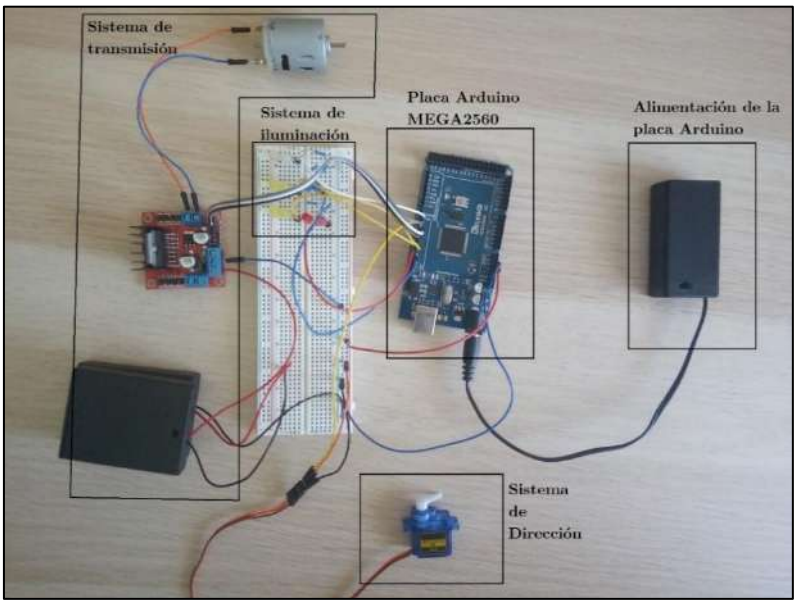


Ilustración 27. Circuitería de los sistemas del vehículo

En el *Anexo I*, puede encontrar el diagrama de clases de los sistemas del vehículo.
En el *Anexo II*, puede encontrar el modelo completo de la circuitería del vehículo.

4.2 Infraestructura básica de la aplicación móvil

Tal y como ocurrió con la infraestructura básica del coche Arduino, para la aplicación móvil se ha buscado construir un sistema mantenible y escalable haciendo uso de buenas prácticas de desarrollo software. De hecho, el trabajo de esta fase ha ido dirigido casi en su totalidad a desarrollar una aplicación capaz de interactuar con el usuario y el coche Arduino a la vez. Por este motivo se ha podido extrapolar la estructura de subsistemas desarrollados para el coche a la aplicación móvil, siendo necesario solo adaptarse a los requerimientos específicos de Android. Aunque antes de entrar en detalle sobre el trabajo realizado en esta fase, se va a describir de forma superficial qué ofrece la aplicación móvil al jugador y cómo se juega con el juguete.

Para empezar, es importante tener presente que la aplicación móvil actúa como mando a distancia y videojuego a la vez. Siendo estas dos características, perfectamente separables, las que se complementan para constituir al producto.

- Como mando a distancia, se debe conseguir que la aplicación sea capaz de generar instrucciones interpretables por el Arduino cuando el jugador presione o deslice los actuadores. Además, el control debería ser configurable y adaptable para varios perfiles de usuario.
- Como videojuego, se busca desarrollar un sistema de juego que desafíe al jugador con distintos niveles de construcción de circuitos sobre los que correr con el coche.

¿Cómo se pretende que el juguete sea usado? Tal y como ocurre con otros juguetes de construcción del mercado, no existen más reglas que las que el jugador se quiera imponer. Sin embargo, habitualmente estos tipos de juego ofrecen contenido por sí solos (en el caso de otros juegos de construcción, el contenido es algunos ejemplos sobre qué construir y cómo hacerlo).

Ilustración 28. Ejemplo de un juguete de construcción para niños
Fuente: https://juguetodo.com/Meccano-839550

Race your Track ofrece al jugador una serie de desafíos de construcción de circuitos, los cuales deberán ser superados por el jugador para poder avanzar a los siguientes. En el *Apartado 6. Sistema de juego* se hablará más en detalle sobre los desafíos y cómo se han creado.

Una sesión de juego habitual la forman los siguientes pasos:

1. El jugador abre la aplicación y selecciona un desafío.
2. Todos los desafíos tienen asignada una pista de carreras que el jugador debe construir, donde será la aplicación quién guíe al jugador con la secuencia de instrucciones que debe seguir para construir la pista de carreras.
3. Una vez el jugador ha terminado de construir el circuito, llega la hora de correr sobre la pista de carreras. Para ello el jugador primero coloca el coche en la parrilla de salida y de forma automática se establece la conexión entre el coche y la aplicación para así poder conducirlo.
4. Tras una cuenta atrás el tiempo empieza a correr y el jugador puede conducir su coche por el circuito. Los tiempos y el número de vueltas del jugador son cronometrados con unos contadores de vueltas implementados en este mismo proyecto.
5. Una vez se ha terminado el desafío (se ha completado el número de vueltas que establece el desafío), se muestra al jugador los resultados. En el caso de haber establecido una marca de tiempo menor que la impuesta por el desafío, este se dará por superado y se desbloquearán nuevos desafíos.

En los siguientes puntos del documento se entrará más en detalle en el funcionamiento del sistema de construcción, los contadores de vueltas y el sistema de desafíos; tener presente cómo se juega desde un principio puede ayudar a entender mejor el resto de los puntos explicados.

En esta primera fase, todo el trabajo está dirigido al control del vehículo (sistema capaz de interactuar con el usuario y el coche). Aunque también se ha preparado el terreno a las características que están por venir y se ha diseñado e implementado el menú principal de la aplicación de cara a lograr un mayor realismo en las pruebas de usabilidad de esta fase.

Destacar que se está desarrollando un videojuego dirigido a los más pequeños, es por eso que se busca ofrecer una experiencia divertida a la par que accesible para jugadores de todos los niveles. Por esta razón se ofrecen varias formas de control, totalmente configurables, diferenciadas en la forma de manejar la dirección, la caja de cambios y el acelerador del vehículo.

Ilustración 29. Capturas de la pantalla de inicio y configuración

Como ya se ha mencionado, se ha extrapolado la estructura desarrollada en el código Arduino a la aplicación Android. Por tanto, en la aplicación también existen distintos sistemas totalmente independientes encargados de controlar y construir su propia vista según el modo de conducción que haya elegido el jugador. Además, todos los sistemas son controlados a su vez por una entidad superior (patrón de

diseño Mediador) que administra las acciones que recibe de cada uno de ellos. No se va a describir en detalle ni la estructura implementada, porque sería repetir la lógica en la que se basa el código Arduino, ni el tratamiento de los mensajes, ya que se detallará en el apartado siguiente donde se habla sobre la comunicación entre la aplicación y el vehículo.

Sin embargo, si se considera interesante hacer un paseo por las distintas formas de control, su construcción y el funcionamiento de los controles más interesantes. La vista de control se construye dinámicamente, a partir de una plantilla, según la configuración elegida por el usuario. La plantilla se diseñó contemplando detalles como la inmersión y la velocidad de aprendizaje del jugador. Por ejemplo, se obliga al jugador a conducir con la pantalla en apaisado obteniendo un manejo más inmersivo, o para resultar más intuitivo, los controles de un mismo sistema siempre se localizan en la misma región de la pantalla independientemente del modo de conducción.

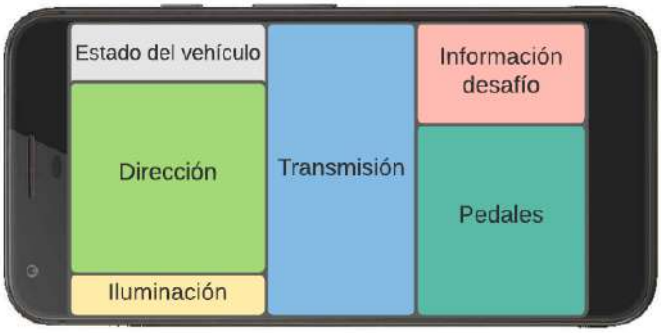


Ilustración 30. Plantilla vista control. Versión final

Ilustración 31. Captura controles de las distintas formas de conducción: principiante, realista y piloto

Todos los controles tienen el mismo funcionamiento base: generar acciones que serán tratadas por un controlador que construirá comandos con ellas. Las acciones se crearán cuando los controles sean presionados o deslizados y lo harán atendiendo a ciertas restricciones. Algunos como los interruptores (botones del sistema de iluminación) generan una acción u otra según su propio estado (patrón

de diseño Estado), pero otros como el volante o el acelerador presentan cierta complejidad. Ambos están implementados con una barra deslizante (componente *SeekBar*) enmascarada bajo un diseño y ciertas peculiaridades, como una posición origen a la que retornan suavemente cuando el usuario deja de sostener la barra. Respecto a la generación de acciones, lo natural hubiera sido que se crearan cada vez que se desplace la barra, pero si el jugador la desliza rápidamente se generarían numerosas acciones que terminan por colapsar el sistema, para corregirlo se ha añadido un tiempo de espera entre acciones.

Nota: No todas las capturas mostradas en este apartado se corresponden con las de la versión final de la aplicación. Algunas son capturas tomadas durante la fase actual de desarrollo que sufrieron cambios en fases posteriores.

4.3 Comunicación Bluetooth

Para comunicar la aplicación Android con el coche Arduino se ha elegido el protocolo de comunicación Bluetooth. Se ha escogido por las siguientes razones:

- A día de hoy, prácticamente todos los dispositivos móviles lo incorporan.
- Sencillo de configurar, tanto en Android como Arduino.
- Al necesitar conectar solo dos dispositivos entre sí, es la forma de comunicación inalámbrica más directa.

Se estuvo evaluando usar Wifi como protocolo de comunicación debido a que Bluetooth solo permite conexiones uno a uno (salvo para dispositivos de audio), y el sistema incluye un contador de vueltas que debe ser capaz de comunicarse con el videojuego. Pero el necesitar una red Wifi restringiría el producto y además complicaría innecesariamente las comunicaciones del sistema. La solución aplicada al contador de vueltas se explicará en el apartado *5. Contadores de vueltas*.

La aplicación móvil actúa de maestro, es decir, es quien inicia la comunicación con el esclavo, el coche radiocontrol. Es por esto que la aplicación controla detalles como que:

- El dispositivo sea compatible con la comunicación Bluetooth.
- El dispositivo tenga disponible Bluetooth, y si no es así permitir al usuario activarlo desde la propia aplicación.
- Buscar el vehículo entre los dispositivos enlazados, y en el caso de no encontrarse vinculado, buscarlo. Un dispositivo está enlazado cuando ya se ha establecido una conexión con él en algún tiempo pasado.

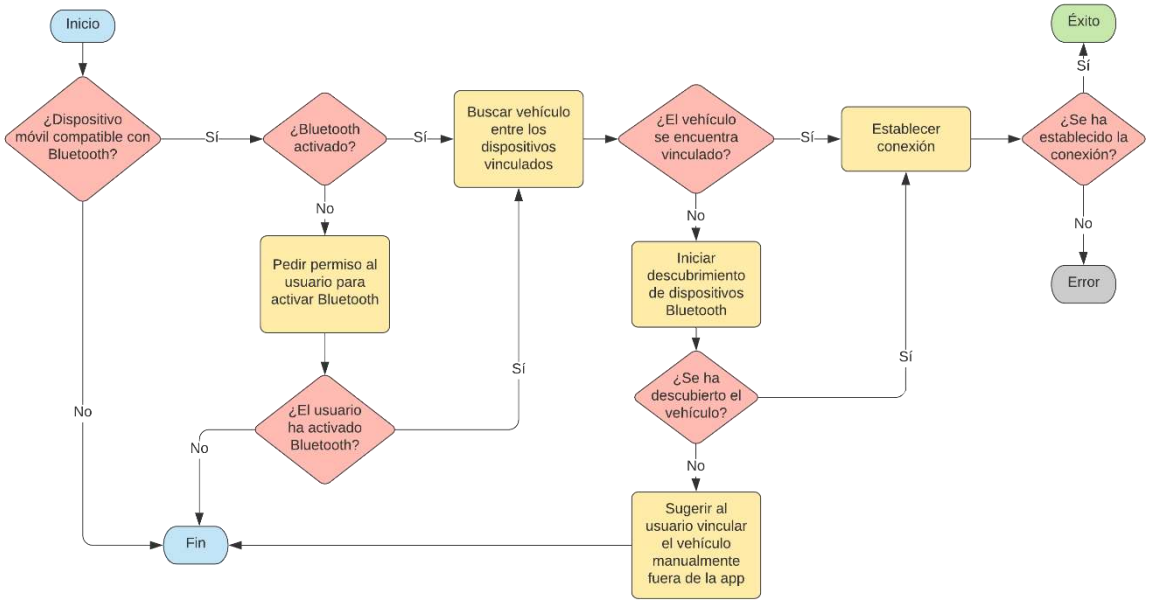


Ilustración 32. Proceso de establecimiento de la conexión Bluetooth

Más a nivel de programación, destacar la importancia de ejecutar estos procesos en *threads* distintos del principal ya que tienen una alta carga computacional que pueden bloquearlo. El *main thread* en Android es el encargado de administrar la vista (de hecho, eso debería ser lo único a lo que se dedicara) por lo que bloquearlo le haría pensar al usuario que la aplicación se ha colgado, cuando en realidad no es así. Se trata de un problema de usabilidad del que hay que huir, para ello se recomienda lanzar en hilos aparte procesos como llamadas a la BD, operaciones de red o servicios web.

De cara a la usabilidad, desde la aplicación se han incluido detalles como un cuadro de diálogo que muestra al usuario el estado del proceso de conexión en todo momento y la emisión del sonido de un motor al arrancar cuando la conexión se ha establecido.

Respecto a la parte Arduino de la conexión, se ha usado un módulo HC-05, descrito en el apartado *2. Estudio del arte*. Para el cableado, puntualizar el detalle de que los pines RX y TX (recepción y transmisión) van cruzados entre el módulo HC-05 y la placa Arduino, de forma que el pin de RX del módulo HC-05 está conectado con el TX del procesador y viceversa. Una vez terminado el montaje, se configuró el módulo mediante comandos AT, con los cuales es posible indicarle al módulo parámetros como el rol, el nombre, el pin de conexión…

```
>> AT                                (Comprobar si el módulo responde)
OK

>> AT+NAME=RaceYourTrack_Car         (Configurar nombre del dispositivo)
OK

>> AT+PSWD=”7806”                    (Configurar pin de vinculación)
OK

>> AT+UART=38400, 0, 0               (Configurar velocidad de comunicación)
OK

>> AT+ROLE=0                         (Indicar rol como esclavo)
OK

>> AT+CMODE=1                        (Permitir conexión con cualquier dispositivo disponible)
OK

>> AT+RESET                          (Salir del modo de configuración)
```

Ilustración 33. Secuencia de comandos AT para configurar módulo HC-05

Aunque el propio módulo HC-05 incluye un pequeño LED que indica el estado de la conexión según la frecuencia de parpadeo, éste no es visible desde el exterior al estar oculto bajo la carrocería. Es por eso que hubiera estado bien incluir otro de mayor tamaño que sí fuera visible, para así indicar tanto desde el videojuego como desde el propio coche el estado de la conexión. *[25][26][27]*

Por último, en cuanto al intercambio de mensajes, aunque ya se ha comentado, recordar que en este punto de la implementación el coche radiocontrol actúa solo como receptor. Es la aplicación quién, mediante los actos del usuario, crea las acciones y construye un comando que envía mediante Bluetooth al coche, quien descompone el comando y almacena las acciones en las colas de los sistemas.

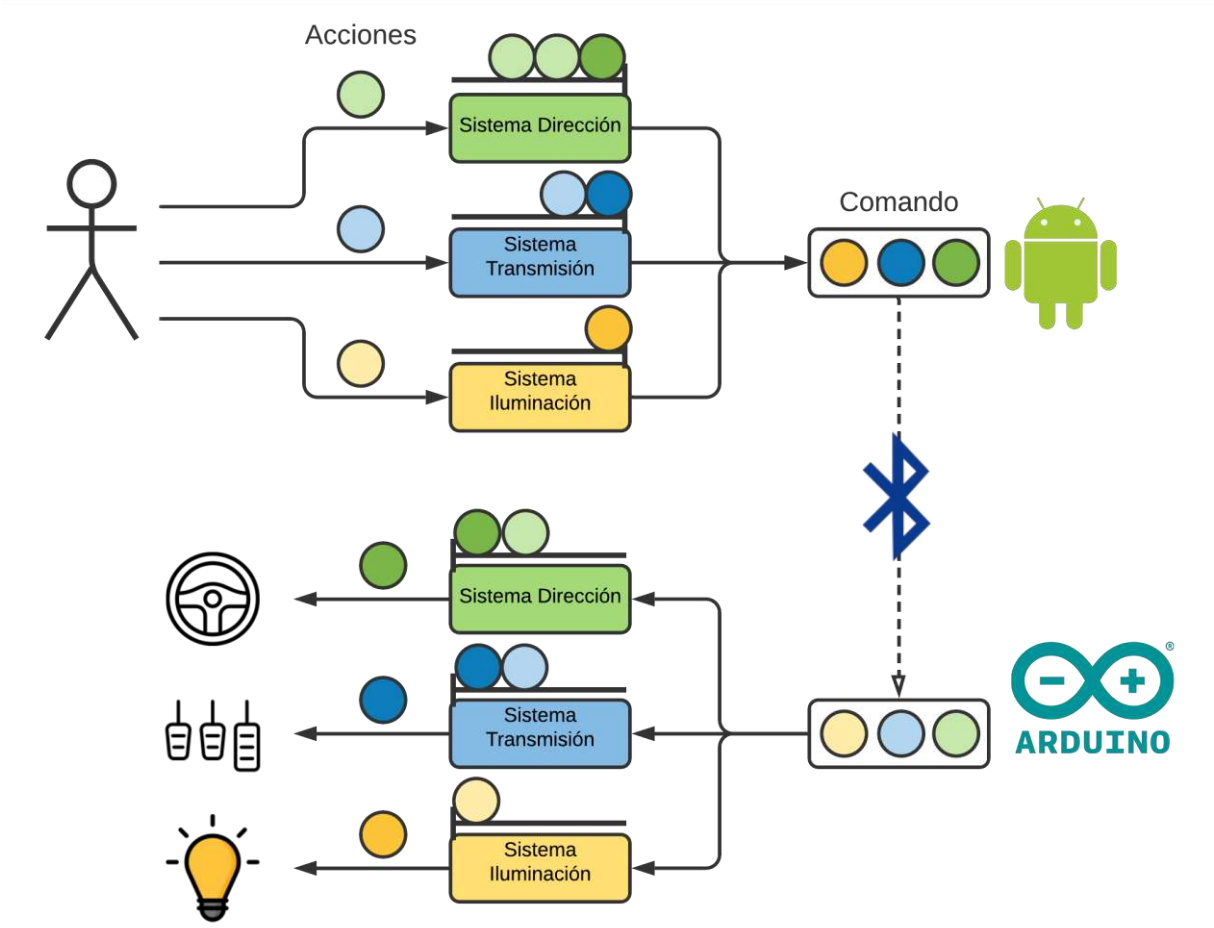


Ilustración 34. Ciclo de vida de las acciones a lo largo de los sistemas

Es necesario almacenar las acciones a enviar/ejecutar en colas, por el hecho de que el envío y la ejecución de las acciones son tareas con periodo. Si no se hubiera adoptado esta solución solo se procesarían las acciones creadas justo en el mismo instante en el que se ejecuta el periodo del proceso, algo poco probable al trabajar con milisegundos.

4.4 Primera prueba de usabilidad

Terminada la infraestructura básica del producto, llegaría el turno de probar la usabilidad del sistema. Durante el proyecto se realizarán tres pruebas de usabilidad, pero al ser esta la primera, se va a presentar de manera superficial en qué consiste exactamente una prueba de este tipo:

Toda prueba de software trata de encontrar errores y validar el sistema que prueba. Las pruebas de usabilidad no son menos, solo que en lugar de ejecutar un banco de pruebas se recurre a un usuario que es quién prueba el sistema en tiempo real delante de un entrevistador. En una frase, una prueba de usabilidad se resume en: "Un proceso donde un entrevistador observa cómo un usuario utiliza un sistema informático, o bocetos de este, con el objetivo de identificar problemas de usabilidad". En la prueba intervienen tres partes:

- Sistema a probar: aplicación o sistema con el que interactuar. Puede ir desde un conjunto de bocetos o wireframes de la aplicación a una parte totalmente funcional del sistema.
- Entrevistado: persona que prueba el sistema ejecutando las tareas que el entrevistador a preparado previamente y ayuda a extraer conclusiones rellenando un cuestionario.
- Entrevistador: persona encargada de planificar las pruebas, guiar al entrevistado durante las mismas y extraer conclusiones una vez acabadas. El entrevistador siempre debe tentar al entrevistado a pensar en voz alta y recordarle que no es él a quién se está evaluando, sino que se evalúa al sistema con su ayuda y que si se siente perdido es culpa de la usabilidad del sistema, no suya.

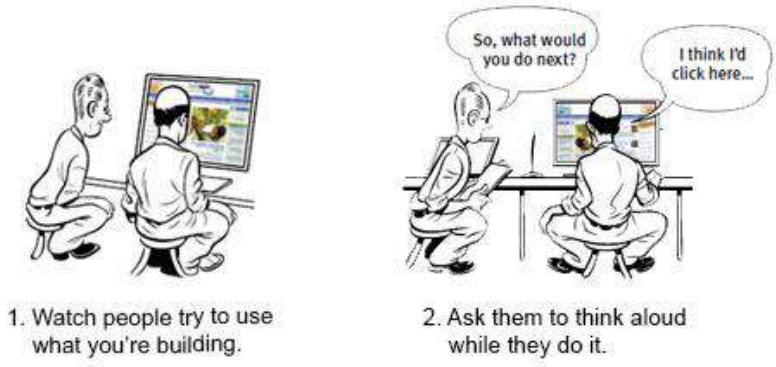


Ilustración 35. Dibujo de un entrevistador y entrevistado durante una prueba de usabilidad
Fuente: https://sensible.com/rocket-surgery-made-easy/

Aunque el proceso siempre es el mismo, existen numerosas plantillas de documentos que pueden ser útiles en las pruebas de usabilidad. Para este proyecto, se ha tomado como referencia las que proporciona Steve Krug, aunque con algunas modificaciones como omitir el cuestionario de identificación del usuario, y modificar el post-test, ya que el que ofrece Steve Krug está muy enfocado a un sitio web y no sería de ayuda para un videojuego. Por eso, se ha creado un post test a medida para la aplicación RaceYourTrack y orientado a esta primera fase de trabajo. [28]

Para esta prueba de usabilidad, se le ha pedido al entrevistado realizar cuatro tareas tratando de cubrir todos los aspectos desarrollados en esta primera fase de trabajo. Solicitando al usuario que pruebe a conducir una maqueta del vehículo usando todos y cada uno de los modos de conducción disponibles.

Antes de las pruebas, se tenía la duda de si "esconder" los modos de conducción en el menú *Garaje* era una buena idea de cara a la usabilidad. Sin embargo, el usuario no tuvo problema alguno en encontrarlos pues se dirigió por intuición directamente hacia el lugar correcto. De las pruebas se extrayeron los siguientes puntos de mejora:

- Dar "vida" a los botones del menú prinicipal para hacerlos más responsivos.
- Añadir iconos a las opciones de configuración de la conducción para identificar rápidamente la elección deseada. El entrevistado pasó demasiado tiempo leyendo las opciones.
- Mostrar la información referente al vehículo (nombre, marcha actual…) en la pantalla de control del vehículo.

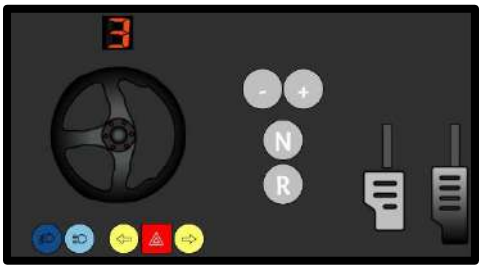


Ilustración 36. Vista de control profesional con la mejora de usabilidad

En el *Anexo III*, puede encontrar los documentos usados en la prueba de usabilidad.

5. Sistema de construcción guiada de circuitos

La función del sistema de construcción guiada de circuitos es ayudar al jugador a construir un circuito paso a paso, mostrando en cada punto qué parte del circuito debe construir y qué piezas necesita. Para lograr esto, a partir de la información almacenada por un circuito, el sistema de construcción debe ser capaz de crear la secuencia de pasos a seguir.

Antes de detallar la información que almacenan los circuitos, y para entender la idea detrás de la creación de los pasos, conviene conocer dos aspectos clave en el proceso:

- Todas las pistas de carreras se construyen usando dos piezas: rectas y curvas de 90 grados. Ambas piezas se encuentran en un archivo .png con dimensiones cuadradas, y mediante rotaciones de 90 grados se obtienen sus variantes.

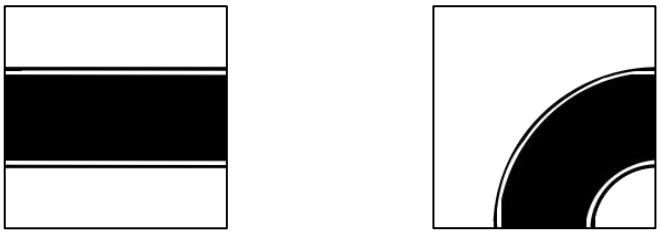


Ilustración 37. Piezas usadas en la construcción de circuitos

- Para construir la vista de los circuitos se usa un GridLayout. Un GridLayout es un esquema que permite disponer componentes visuales en forma de tabla.

Teniendo estos aspectos presentes, se entenderá mejor la información que almacenan las pistas de carreras y el por qué lo hacen. En el modelo de clases, hay que distinguir tres entidades: pista de carreras, celda y pieza.

- Pista de carreras: además de información útil para el videojuego como tipo, nombre, descripción…, define su tamaño y referencia a un conjunto de celdas. El tamaño es un atributo que indica el número de celdas que contiene la pista.

- Celdas: almacenan una coordenada (x, y) dentro de la matriz de la pista de carreras, un orden que será usado por el sistema de construcción para crear los pasos y una lista de piezas.
- Piezas: tienen una coordenada (x, y) y un tipo de pieza (recta o curva) junto con su rotación.

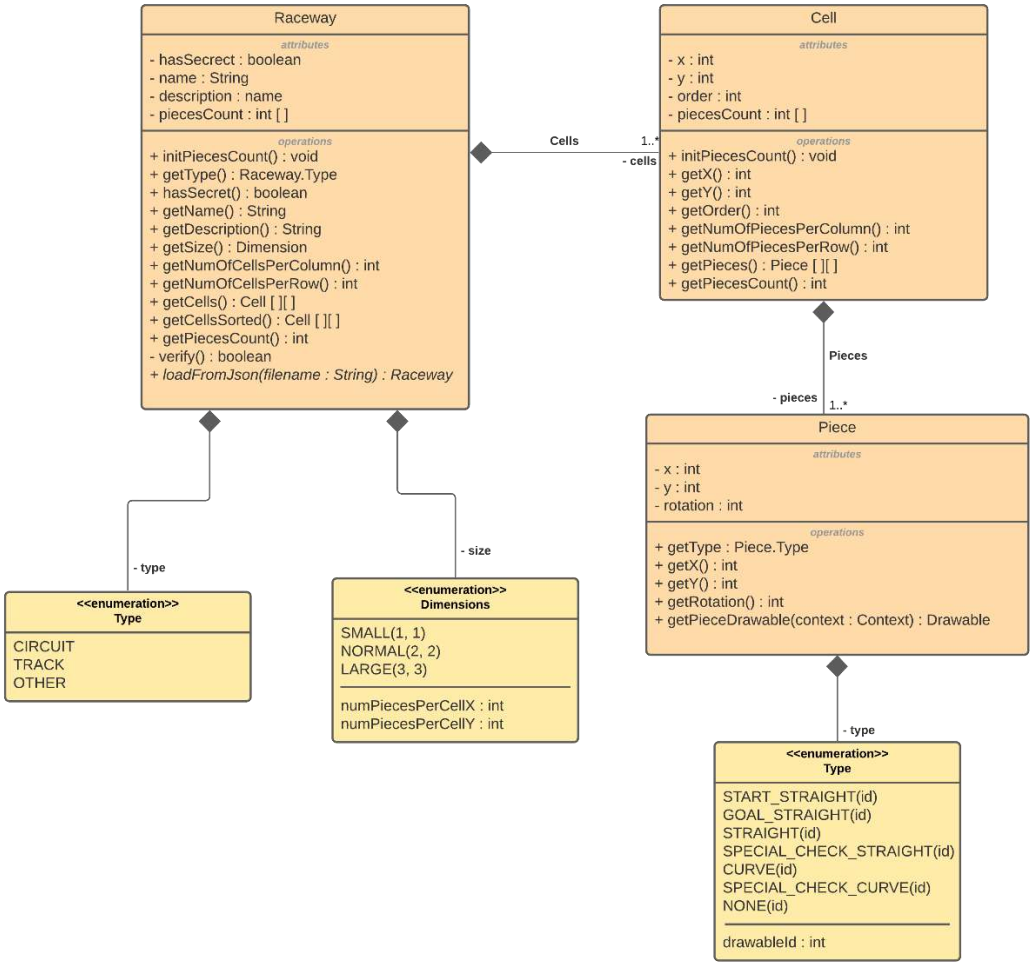


Ilustración 38. Diagrama de clases de las entidades usadas por el sistema de construcción

Con esta información el sistema de construcción es capaz de disponer visualmente la pista de carreras y crear la secuencia de pasos de construcción, donde será el jugador quien marque el ritmo, de forma que el sistema servirá como navegador entre las distintas partes del circuito.

Para construir visualmente el circuito se hace uso de los dos aspectos mencionados antes: piezas cuadradas y un GridLayout. Lo primero es calcular cual es la longitud máxima admitida en pixeles del lado de las celdas cuadradas, esto se calcula atendiendo a las dimensiones de la pantalla y el número de celdas del circuito. Una

vez hecho esto, se definen tantas casillas en el GridLayout como celdas hay en la pista, realizando una asignación uno a uno según las coordenadas que tienen definidas las celdas. A su vez, para las celdas se sigue el mismo procedimiento: se calcula el tamaño máximo de pieza, se crean tantas casillas en el GridLayout como piezas tiene la celda y se coloca cada pieza en su celda correspondiente según sus coordenadas.

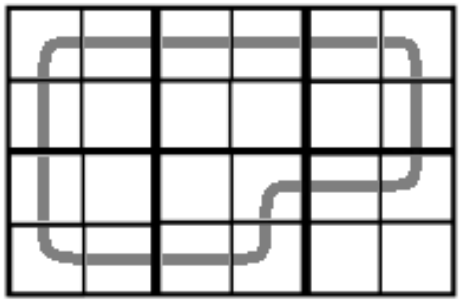


Ilustración 39. Esquema de construcción de un circuito con 2x3 celdas y 4 piezas por celda

Respecto a los pasos que crea dinámicamente el sistema de construcción, consisten en mostrar en pantalla una celda concreta siguiendo el orden definido por el circuito.

¿De dónde obtienen los circuitos toda esa información? De la lectura de ficheros JSON. La solución adoptada para evitar tener que definir en código todos los circuitos es leer su información de ficheros .json. Con esto, se obtiene una herramienta fácil de usar y eficiente para la creación de nuevos circuitos (basta con crear un nuevo fichero .json) y se evita tener en memoria RAM todos los circuitos al mismo tiempo por estar definidos en código. Además, se ha creado un validador que verifica si el circuito está bien definido en el fichero JSON, esto pensando en evitar errores en un futuro cuando se creen nuevas pistas de carreras. Para la lectura de los ficheros JSON se ha usado la librería GSON, puede encontrar más información sobre ella y sobre JSON en el *apartado 2. Estudio del arte.*

En el *Anexo IV* se muestra el contenido de un fichero .json de un circuito de ejemplo.

A pesar de que el sistema de construcción permite crear fácilmente circuitos gracias al uso de JSON, presenta limitaciones como el hecho de que solo se pueden usar rectas y curvas de 90 grados. Esto se debe a la restricción establecida de que todas las piezas deben ser cuadradas, aunque mencionar que sería posible escalar este sistema, pues los GridLayouts no restringen que todas sus casillas deban ser de las mismas dimensiones. Por tanto, se podrían crear piezas rectangulares e indicarle al layout que esa pieza ocupa dos columnas/filas.

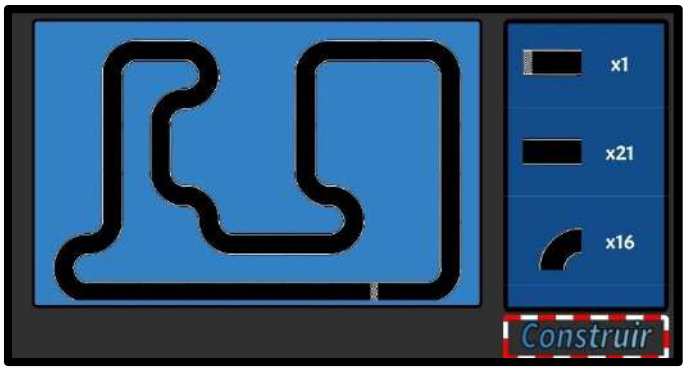


Ilustración 40. Circuito complejo que demuestra el potencial del sistema de construcción guiada de circuitos

Por último, se creó un pequeño kit de construcción de circuitos para el videojuego usando cartones. Sin embargo, hay que recordar que los circuitos se pueden construir con cualquier tipo de material u objeto casero, no existen realmente limitaciones en ese aspecto más que las que el jugador se quiera imponer.

Ilustración 41. Piezas del kit de construcción hechas de cartón junto con los check-points

5.1 Segunda prueba de usabilidad

Como tras la primera fase de desarrollo, se realizó una prueba de usabilidad para comprobar cómo de usable es el sistema de construcción, y corregir los problemas encontrados con la ayuda del usuario.

En esta prueba, se le ha pedido al entrevistado que construya dos pistas, un circuito pequeño y un tramo de longitud media. Se valoró el añadir una tercera tarea a la prueba, pero esta idea fue descartada porque supondría una gran inversión de tiempo por parte del entrevistado que tampoco serviría de gran ayuda, puesto que al repetir la misma tarea una y otra vez (aunque con distintas pistas de carreras) el propio usuario se acomodaría a los errores de usabilidad y sería capaz de usar el sistema.

En la primera tarea, el usuario al terminar la construcción del circuito pequeño no sabía que realmente había terminado la construcción. Es decir, el circuito estaba construido pero la aplicación no le había dado ningún indicador de que el proceso había terminado. Para solventar esto, se decidió incorporar la emisión de un sonido especial al terminar la construcción así como mostrar un diálogo (ventana flotante) que dejara claro al usuario el final del proceso.

Los resultados extraídos de la segunda tarea fueron aún más interesantes, puesto que al tratarse de una pista de carreras más grande (más piezas en una misma celda) se pudieron identificar dos problemas de usabilidad bastante serios:

- El entrevistado por momentos no sabía realmente qué parte del tramo estaba construyendo, es decir no sabía que celda se le estaba mostrando en pantalla.
- El entrevistado mostró confusión al elegir la cantidad de piezas a usar en un paso concreto, a pesar de que el número de piezas necesarias para el paso está siempre visible en pantalla.

Finalmente el usuario completó la tarea gracias a su propia intuición pero encontró algunas dificultades que pueden ser evitadas. Las correcciones adoptadas fueron:

- Añadir la posibilidad de que el usuario mostrara al completo el circuito para identificar que parte está construyendo. Esta medida más que una solución es un parche, pues se cree que hay mejores. Por ejemplo, tener el circuito en

miniatura siempre visible y resaltar la celda en construcción, o mostrar la celda del paso específico rodeada por las adyacentes. Pero son medidas que requieren cambiar casi por completo la interfaz, lo que supondría un costo de tiempo muy grande que desajustaría la planificación inicial. Además, no se trata de un error que entorpeciera en exceso o impidiera al usuario completar la tarea, por lo que no tiene prioridad máxima.

- En cada paso, mostrar las piezas de una misma celda separadas entre sí, de forma que al usuario le resulte más sencillo identificar donde colocarlas.

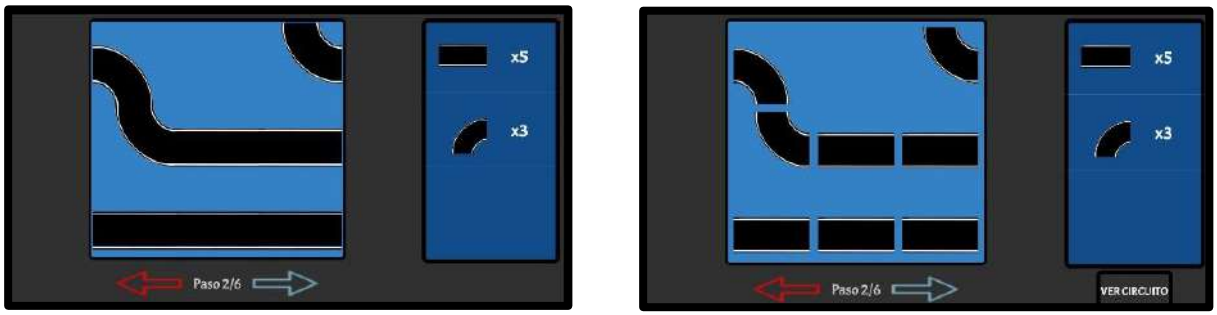


Ilustración 42. Paso de construcción guiada antes y después de la prueba de usabilidad

Como conclusión, se ha desarrollado un sistema que aunque no es perfecto y presenta algunas limitaciones, guía al jugador a construir pistas de carreras y permite al desarrollador diseñar nuevos circuitos con rapidez. Además, abre las puertas a nuevas características, como la inclusión de piezas no cuadradas, y gracias a la prueba de usabilidad ha podido ser mejorado.

6. Contadores de vueltas

Con la intención de potenciar la jugabilidad de Race your Track, se decidió añadir alguna forma de poder contabilizar y/o cronometrar las vueltas del jugador a las pistas de carrera que construya.

La idea inicial era construir un contador de vueltas controlado por un Arduino Nano, que mediante un sensor ultrasónico HC-SR04, notificara a la aplicación móvil de que "algo" había pasado frente al contador. Aunque esta idea fue rechazada por dos motivos:

- Primero, ese "algo" no tiene por qué ser el coche radiocontrol. Sería sencillo para el jugador acercar cualquier objeto al sensor y engañar así al sistema.
- Segundo, y el motivo principal, realizar una implementación así causaría un exceso de complejidad en las comunicaciones del sistema. Pues sería necesario usar el protocolo Wifi, ya que Bluetooth solo admite comunicación entre pares de dispositivos (salvo para dispositivos de audio, donde se permiten varios esclavos).

Por esto, se dio una vuelta de tuerca a la idea, y se decidió recurrir a la tecnología RFID (Identificación por Radiofrecuencia). Incorporando al coche el lector RFID-RC522 y sustituyendo los contadores por simples tags, compatibles con la lectura por RFID, que actuarían como *check-points*. Con esta implementación, se solucionan los dos problemas mencionados antes, pues el coche está obligado a pasar por el *check-point* (evitando las trampas del jugador) y será él mismo quién notifique a la aplicación mediante la comunicación Bluetooth ya implementada.

En cuanto al proceso llevado a cabo, lo primero fue incorporar el lector RFID-RC522 a la circuitería del vehículo.*[34]* Tras esto, con el propósito de mantener la escalabilidad conseguida en la primera fase del proyecto, se añadió al sketch Arduino un nuevo subsistema independiente, cuya función es leer tarjetas por RFID y enviar su código a la aplicación móvil mediante comunicación Bluetooth. El código, solo será enviado si se corresponde con alguno de los registrados, que para este prototipo son los UID de una tarjeta y un llavero. La rutina simplificada del sistema es la siguiente:

1. Comprueba si hay una tarjeta que leer.
2. Si la hay, lee su UID.
3. Comprueba si el UID leído es igual a alguno de los que tiene registrado.
4. Si está registrado, envía por Bluetooth el código correspondiente.

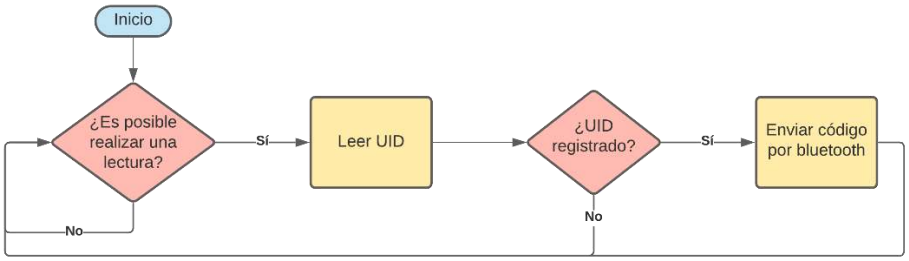


Ilustración 43. Diagrama flujo rutina sistema lector RFID

Sobre la lectura RFID desde el Arduino, destacar que se ha recurrido a una librería *open source* que ya implementa los métodos necesarios para realizar lecturas.

Hasta ahora, se ha hablado sobre cuál es y cómo se ha ideado el indicador de paso por meta. Pero no se debe olvidar que se ha hecho con el propósito de poder contabilizar y cronometrar las vueltas del jugador por la pista. Para ello, en la parte Android, aprovechando que ya se tiene implementada la comunicación Bluetooth, se ha añadido el tratamiento de los códigos de los *check-points* recibidos desde el vehículo. Este tratamiento implica identificar qué tipo de *check-point* se ha recibido (tarjeta o llavero) y en función de cuál sea realizar una acción u otra.

Con la intención de encapsular funcionalidad, se ha incorporado un nuevo componente al código Android llamado *LapCounter*. Una instancia de esta clase abstrae todo el comportamiento necesario para contabilizar y cronometrar las vueltas del vehículo por un circuito. De esta forma, el sistema de desafíos del juego será más sencillo de desarrollar y se podrán controlar más fácilmente los tiempos y el número de vueltas desde un único punto especializado en ello. Respecto al funcionamiento, tras una configuración (número de vueltas, número de *check-points*…) el componente se limita a calcular la diferencia de tiempo en milisegundos entre la recepción de dos mensajes de paso por *check-point*, luego esa diferencia se registra formateada a un tiempo de vuelta con minutos, segundos y milisegundos.

Pensando en la usabilidad del prototipo, se han añadido efectos de sonido para cada *check-point*. Con esto se consigue que el usuario identifique mientras juega si el vehículo ha registrado su paso por el *check-point* (sobre este inconveniente se habla a continuación).

Esta solución no es perfecta, pues presenta algunas limitaciones:

- Por restricciones de proximidad de la lectura RFID, se exige al vehículo pasar cerca de la tarjeta, si no la vuelta no se contabilizará. Como parche a este problema, en la construcción guiada de circuitos se puede indicar que la zona donde se encuentran los tags RFID sea estrecha para obligar al jugador a pasar cerca de ella.
- Es propensa al error de sucesivas lecturas consecutivas en poco tiempo, lo que causaría comportamientos anómalos en el juego. Para solventarlo se han establecido restricciones de tiempo mínimo entre una lectura y otra. Aunque la solución más apropiada hubiera sido añadir un nuevo *check-point*, de forma que siempre se debe de leer uno para tener en cuenta la lectura del siguiente.

Como resultado de esta fase, se ha implementado un sistema de contadores de vueltas, que con algunas limitaciones, es altamente escalable a nivel de *check-points*, pues sería sencillo incorporar al prototipo más tarjetas compatibles con la tecnología RFID y registrarlas en la aplicación para así poder dividir los circuitos y tramos en sectores (como en la realidad), ofreciendo una experiencia más divertida y realista.

7. Sistema de juego y desafíos

Fundamentalmente, el sistema de juego se sostiene de una mecánica: la progresión del jugador mediante la superación y el desbloqueo de desafíos cada vez más difíciles. Los desafíos son niveles de construcción de pistas de carreras sobre los que una vez terminados, el jugador conducirá el coche sobre ellas tratando de marcar un tiempo que le permita obtener una medalla y así desbloquear más niveles.

A nivel técnico, los desafíos son objetos que almacenan la siguiente información:

- La pista de carreras (circuito o tramo) que se debe construir y sobre la que se debe correr.
- Detalles cómo la dificultad del circuito, si está desbloqueado o si tiene un secreto entre sus piezas. Esta última es una propiedad derivada del circuito.
- Las marcas necesarias para cada una de las medallas así como, en el caso de existir, el tiempo que ha marcado el jugador en ese nivel.

Recalcar que, existe información que se debe almacenar, y aunque no se ha mencionado antes, para la persistencia de la aplicación se está usando una base de datos SQLite. En fases anteriores ya se almacenaba información, un *flag* sobre si el juego se había inicializado o la configuración de control que el jugador está usando. La base de datos SQLite está bastante limitada, por ejemplo la variedad de tipos de datos escasea, no obstante se le ha dado un uso simple, por lo que resulta más que suficiente. En este caso, para cada desafío existe un registro en base de datos, para el cual la única información que se almacena es si ha sido desbloqueado y el tiempo que el jugador marcó en él.

Tal y como se pensó para las pistas de carreras, en este caso tampoco se han definido los desafíos de forma estática en la memoria de la aplicación. Sino que existe un fichero JSON en el que se encuentran las definiciones de todos los desafíos de la aplicación, y de ahí se van cargando. Nótese que la carga de un desafío la forman dos pasos: primero se carga desde el fichero JSON y luego se usa su identificador para obtener de la base de datos el resto de información.

```
{
    "id": 3, "racewayFile": "large.json", "laps": 3,
    "scores": [
      {"requestTime": {"hours": 0, "mins": 2, "secs": 30, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 2, "secs": 0, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 1, "secs": 30, "millis": 0}}
    ], "challengesRequired": [1, 2]
 }
```

Ilustración 44. Estructura de la definición de un desafío en JSON

En el *Anexo VI*, puede encontrar un ejemplo del contenido del fichero donde se definen los desafíos.

En cuanto a la dificultad, cada desafío tiene la suya propia. La cual se calcula automáticamente según el número de curvas del circuito y su tamaño. Se valoró la posibilidad de cronometrar el tiempo de construcción e incluso forzar al jugador a conducir con una configuración u otra en ciertos niveles, todo con la intención de potenciar la dificultad. Sin embargo fueron rechazadas, puesto que se pretende que los circuitos se construyan bien y no rápido necesariamente, además los modos de conducción se diseñaron con el propósito de que el jugador experimentara y escogiera el modo de control que le resulte más cómodo y divertido y no cómo una característica que dificulte al jugador.

Ilustración 45. Lista de dasfíos de tramos de la aplicación, donde aparecen algunos desbloqueados y otros incompletos

Aprovechando la sencillez de uso de los *check-points*, para hacer el juego más divertido se ha usado el llavero como un *check-point* especial que se debe recoger al menos una vez por carrera. Su potencial es alto, y los límites los pone el diseñador de niveles, aunque normalmente se le ha exigido al jugador elegir una ruta alternativa como ocurre con la *Joker Lap* en el Rallycross (en las carreras de rallycross, todos los pilotos deben dar una vuelta pasando por la zona comodín, que dependiendo del circuito puede ser un tramo más complejo o un atajo).

Realmente implementar el sistema de desafíos ha sido relativamente sencillo, pues ha bastado con integrar y comunicar entre sí los tres subsistemas ya desarrollados:

- Sistema de construcción guiada de pistas de carreras: para indicar al jugador cómo construir la pista concreta del desafío.
- Contador de vueltas: inicializado con el número de vueltas que defina el desafío y usado para cronometrar los tiempos del jugador.
- Mando de control: sistema que el jugador usará para manejar el coche radiocontrol.

El mando de control ha sido necesario crear uno nuevo que hereda del anterior. En él, se ha incluido la información del contador de vueltas (vuelta actual y si se ha encontrado el *check-point* especial), así como una cuenta atrás al inicio de la carrera.

Por último, tras terminar el desafío se muestra una pantalla en la que aparecen los resultados obtenidos.

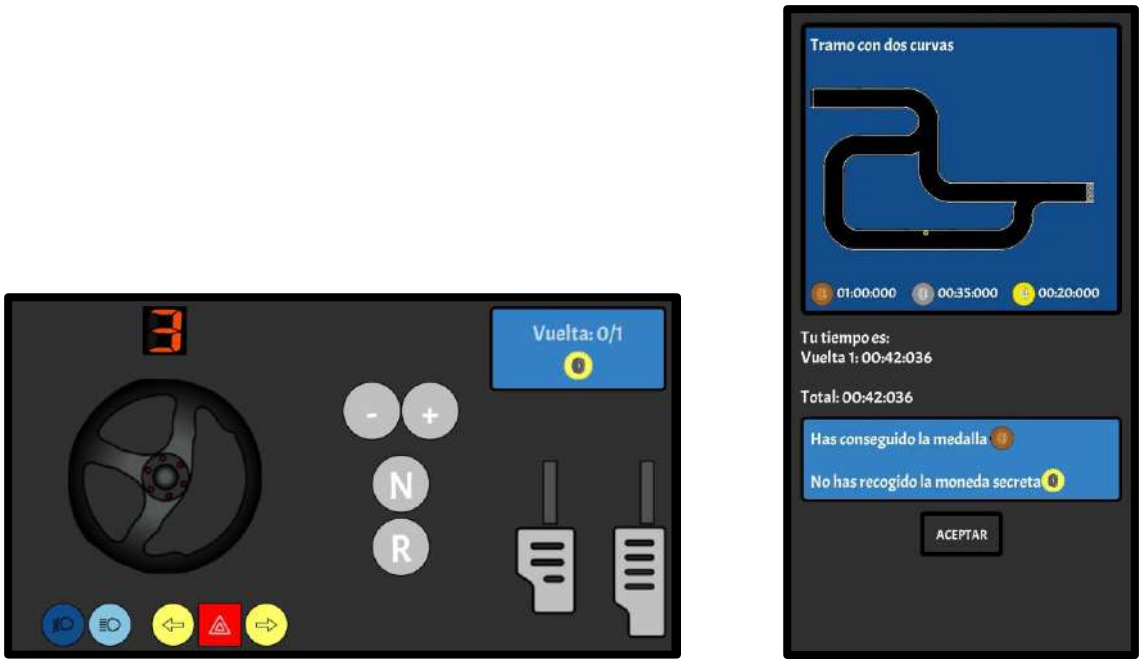


Ilustración 46. Capturas de pantalla de la aplicación durante y tras un desafío

7.1 Tercera prueba de usabilidad

Aunque la usabilidad no solo tiene sentido cuando se habla de aplicaciones informáticas (pues se aplica a cualquier interfaz de usuario), para el estado actual del proyecto las características nuevas a probar son prácticamente mínimas pues al integrar distintos sistemas ya probados pocas funcionalidades nuevas existen. Sin embargo, nuevamente la prueba de usabilidad a ayudado a detectar dos problemas de usabilidad, uno de ellos grave tratándose de un videojuego.

Mientras el usuario ejecutaba la única tarea de esta prueba de usabilidad, se identificó un problema de navegación cuando el usuario se dirigía a la selección de tramos. Pues el entrevistado trató de cambiar entre las pestañas deslizando la pantalla a la izquierda, cuando la única forma implementada de hacerlo es haciendo click en los botones superiores.

Aunque el problema serio vino al terminar el proceso, en donde al pedirle al jugador que interpretara en voz alta la información que mostraba la vista de resultados del desafío, resaltó la necesidad de un botón “reiniciar nivel” para el caso de que un jugador no alcance la máxima calificación. Especialmente, en un juego como este donde la construcción de circuitos puede llevar tiempo, incluir ese botón puede ser muy útil y reducir tiempos al jugador.

Como conclusión de esta fase. Se ha implementado un sistema de juego escueto en el que se involucran todos los sistemas desarrollados a lo largo del proyecto. Un sistema de juego que ha podido ser mejorado gracias a la prueba de usabilidad y en el que la diversión reside en la propia jugabilidad y no en las recompensas que se ofrecen, sin embargo en las líneas futuras se enumeran algunas mejoras respecto a este apartado.

8. Contenidos de la aplicación y prototipo final

Llegados a este punto, la lógica del prototipo estaría completa, restaría añadir contenidos al videojuego para que no esté vacío y montar el prototipo del coche radiocontrol.

8.1 Contenidos de la aplicación

Por contenidos se refiere a los circuitos y niveles del videojuego. A nivel informático, por la forma en que se ha implementado el sistema, añadir contenidos al juego es relativamente sencillo, pues basta con definir un circuito en un nuevo fichero .json (puede encontrar un ejemplo en el *Anexo* IV) y luego crear un nivel que lo referencie.

Durante la creación de los niveles se ha prestado especial atención a aspectos como la curva de dificultad, crear una progresión con la que el jugador aprenda todas las características del videojuego y aprovechar la reutilización de circuitos ya construidos.

Además, se ha tratado exprimir al máximo la característica del *check* especial, pues se considera que presenta un atractivo particular para el jugador, pues lo obliga a tener que escoger entre dos caminos, uno normalmente más complejo que el otro pero que ofrece una recompensa. Para poder añadir bifurcaciones en el sistema de construcción, ha sido necesario crear dos nuevas piezas que mediante sus rotaciones permitan abrir dos caminos en los circuitos.

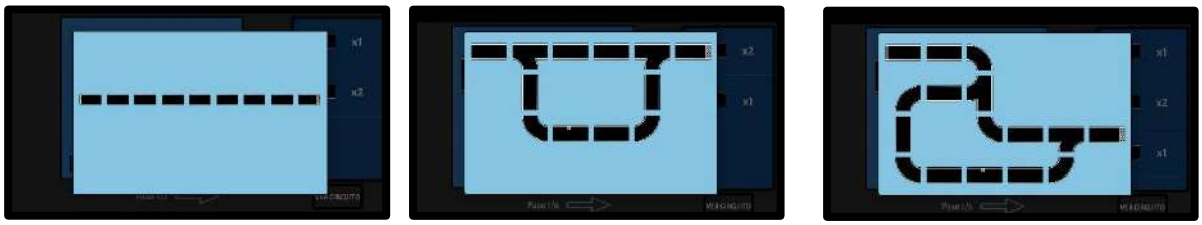


Ilustración 47. Tramos correspondientes a los tres primeros niveles

8.2 Construcción del prototipo final

Tal y como se mencionó, no entra dentro de los objetivos del proyecto construir un coche radiocontrol que cumpla con los requisitos de un juguete de este calibre, como lo son la fiabilidad, seguridad, resistencia o estética. El propósito es construir un prototipo que permita comprobar que tanto el código y la circuitería del coche, como la comunicación entre la aplicación y el vehículo funcionan correctamente. Por esta razón, no se ha hecho un diseño del producto desde cero, sino que se ha tomado como base un coche radiocontrol al que se le han aplicado las modificaciones necesarias para incrustar toda la circuitería. A continuación, se expone brevemente la secuencia de pasos del proceso de montaje del prototipo.

Lo primero fue desmontar el coche y ver que traía en su interior para eliminar las piezas de serie que no fueran útiles y dejar aquellas que se pudieran aprovechar. Por la forma en la que viene está diseñado el eje delantero resultaba imposible instalar la dirección como se tenía pensado en un inicio, por lo que fue necesario acoplar un pequeño módulo debajo del vehículo.

Ilustración 48. Carrocería e interior del prototipo del coche radiocontrol (47 x 20 cm)

El proceso seguido en el montaje es igual al del desarrollo de los subsistemas de la circuitería: transmisión, dirección e iluminación. Para empezar se instaló un motor CC en el eje trasero para poder impulsar al vehículo. Tras comprobar que funcionaba correctamente, se construyó el módulo de la dirección, el cual fue inspirado por los productos de Drift Hybrid Gaming, y se instaló en los bajos del vehículo.

Ilustración 49. Transmisión y dirección del prototipo del vehículo

El siguiente paso fue instalar el sistema de iluminación, aunque previamente se incorporó la placa y las baterías de alimentación del vehículo. Para instalar el sistema de iluminación fue necesario hacer algunos orificios en la carrocería del vehículo, así como construir y soldar cables a medida.

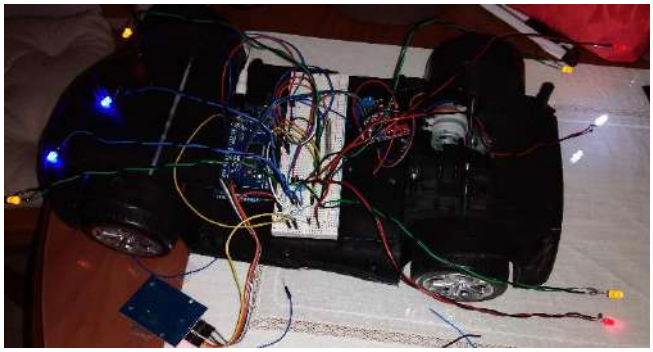


Ilustración 50. Sistema de iluminación instalado en el prototipo radiocontrol

Por último, se añadió el módulo lector de tarjetas RFID al lateral de la carrocería. Como el ángulo que forman la tarjeta y el lector durante las comunicaciones es un detalle a tener en cuenta, se le añadió al lateral del coche un pequeño soporte para hacer que el lector se sitúe perpendicularmente al suelo.

Ilustración 51. Lector RFID instalado en el prototipo del vehículo

Para dar por terminado el proceso de construcción se realizó una prueba completa del prototipo en funcionamiento y los resultados fueron algo decepcionantes. Pues, a pesar de la placa ser capaz de accionar todos los mecanismos cuando el vehículo está en alza, el prototipo no puede desplazar su propio peso, siendo solo capaz de hacerlo en la marcha más alta y con dificultades. Además, la fiabilidad de las lecturas RFID es baja, pues requiere que el lector y la tarjeta se encuentren muy próximos. Por otro lado, aunque si que es cierto que este no es el resultado esperado, tanto el software como el esquema de montaje usados para el vehículo son funcionales al 100%, por lo que podrían perfectamente ser usados en otro prototipo o producto.

El resultado de esta fase es el propio resultado del proyecto, pues ya se han implementado todas las funcionalidades y se han creado todos los contenidos de la primera versión de la aplicación y se ha construido el prototipo del coche radiocontrol, las conclusiones sobre el mismo se exponen en el siguiente punto.

9. Conclusiones y líneas futuras

A continuación, se valorará el grado de cumplimiento de los objetivos del proyecto y se enumerarán posibles líneas de trabajo derivadas del mismo.

9.1 Conclusiones

El objetivo principal del proyecto era desarrollar el prototipo de un juguete compuesto por un coche radiocontrol y un videojuego para móviles que proponga al jugador la construcción de circuitos con materiales caseros para luego conducir el coche por ellos. Todo esto, alcanzando un grado de usabilidad aceptable respaldado por unas pruebas de usabilidad. Tras el desarrollo del proyecto, extraemos las siguientes conclusiones:

- Se ha construido un prototipo de coche radiocontrol usando Arduino que puede ser controlado desde la aplicación móvil mediante comunicación Bluetooth.
- Se ha implementado un sistema de construcción guiada de circuitos versátil para el que se puede generar contenidos fácilmente.
- Se ha diseñado un sistema de desafíos que propone la construcción de pistas de carreras y que cronometra los tiempos que el jugador marca en ellas.
- Se ha implementado un mando de control para el vehículo que admite distintas configuraciones adaptables a las necesidades del jugador.
- Se han realizado pruebas de usabilidad que han ayudado a identificar problemas y corregirlos.

A pesar de haber logrado los objetivos en mayor o menor medida, existen una serie de mejoras y limitaciones que se deben mencionar:

- El prototipo de coche radiocontrol construido no es todo lo fiel al producto final que se querría, pues presenta problemas de fiabilidad y resistencia, así como unas dimensiones mayores de las deseadas. Además, el hecho de que su desplazamiento esté limitado a la máxima potencia incumple las expectativas depositadas al inicio del proyecto. Sin embargo, todos sus sistemas funcionan correctamente cuando el vehículo está en alza, y aunque su control y comportamiento no es todo lo suave y responsivo que se desea, ha resultado de utilidad para verificar el sistema desarrollado.

- El sistema de construcción guiada es mejorable, pues aunque no limita al jugador a usar unas piezas concretas en sus circuitos, tampoco sugiere objetos ni materiales caseros a usar como rectas o curvas ni propone crear un kit de construcción propio.
- La tecnología RFID usada para los contadores de vueltas puede presentar problemas de fiabilidad y arruinar la experiencia de juego si el coche no pasa lo suficientemente cerca de los contadores de vueltas.
- Las recompensas del juego están limitadas a desbloquear nuevos niveles. Se podría incorporar un sistema de monedas mediante el que desbloquear cosméticos o niveles especiales.
- No se incluyen características necesarias en un videojuego de este tipo tales como tutoriales, mensajes de seguridad como “Busca un lugar espacioso donde jugar”, etc.

De la metodología aplicada durante el proceso de desarrollo del proyecto se pueden extraer los siguientes resultados:

- La aplicación de buenas prácticas software, así como el empleo de modelos, esquemas y bocetos son fundamentales para el desarrollo de buen software.
- Ceñirse a una planificación en proyectos software de cierta envergadura que involucran tecnologías con las que no se está familiarizado es difícil. Aunque se han cumplido los objetivos del proyecto, no ha sido posible implementar todas las características que se tenían en mente al inicio por falta de tiempo a causa de los imprevistos surgidos por el desconocimiento de algunas tecnologías. Realizar un análisis de riesgos habría ayudado al proceso de desarrollo del proyecto.
- Las pruebas de usabilidad pueden llegar a ser muy útiles y eficaces si se siguen dos “máximas” que sugiere Steve Krug: “Una mañana al mes, esto es todo lo que pedimos” y “Céntrese implacablemente en los problemas más serios”. Con esto Steve Krug lo que quiere decir es que realizar pruebas de usabilidad no requiere gran cantidad de recursos (en este caso tiempo) y que corregir todos y cada uno de los problemas de usabilidad de un sistema es prácticamente imposible, por lo que conviene centrarse en los más graves. *[28]*

Como conclusión, se considera que el resultado del proyecto ha sido satisfactorio, pues se ha logrado construir el prototipo de un juguete que, aunque lejos de ser un producto final, es totalmente funcional y cumple con los requisitos tanto técnicos como de usabilidad del proyecto. Además, la metodología iterativa seguida (recogida de requisitos, creación de modelos, desarrollo y pruebas de usabilidad) junto con el empleo de tecnologías desconocidas para el estudiante, han contribuido a su formación.

9.2 Líneas futuras

El prototipo de juguete generado presenta potencial y es altamente escalable. Estas son algunas de las posibles tareas para mejorar al prototipo en un futuro:

- Aprovechar el acelerómetro y el giroscopio que traen instalados los dispositivos móviles para añadir una nueva forma de controlar el vehículo.
- Mejorar el vehículo radiocontrol para que se acerque a un producto final. Reducción de tamaño, aumento en fiabilidad y resistencia o sustitución de la placa Arduino por un PCB personalizado son algunas de las posibles mejoras.
- Añadir al sistema de recompensas monedas y al juego una tienda, donde el jugador pueda adquirir coches virtuales y cosméticos. Estos objetos aportarían a la experiencia de juego modificando el número de marchas del vehículo, la potencia máxima, el tipo de cambio manual, el color de las luces delanteras (sería necesario cambiar los diodos LED actuales por unos RGB), el volante mostrado en pantalla entre otras.
- Mejorar el sistema de construcción. Actualmente, está muy limitado por el tipo de piezas que admite (rectas y curvas de 90º), pero gracias a que se construyen usando un grid, sería posible incluir nuevas piezas más complejas que ocupen más de una celda del grid.
- Incluir la validación del circuito construido dentro del propio sistema de juego. Para ello sería necesario que tras terminar la construcción, el jugador fotografiara el resultado y la aplicación mediante visión por computador verificara cual es el grado de parecido con el circuito objetivo.
- Buscar una alternativa a las lecturas por RFID. A pesar de ser una solución aceptable, presenta algunas limitaciones que conviene evitar. Además sería interesante añadir nuevos contadores de vueltas que permitan obtener marcas de tiempo más precisas, pudiendo así dividir las pistas de carreras en sectores e incluso crear niveles más creativos como desafíos de aparcamiento.

Bibliografía

[1] Yiminshum. Situación global móvil 2021. Febrero 19, 2021. Recuperado el 3 de mayo de 2021 de https://yiminshum.com/mobile-movil-mundo-2021/

[2] Carrera Go Plus. Mayo 4, 2017. Recuperado el 8 de mayo de 2021 de https://slot4ever.com/blog/2017/circuitos/como-funciona-carrera-go-plus/

[3] ¿Cómo funciona el slot? (s.f.). Recuperado el 8 de mayo de 2021 de https://www.wikiwand.com/en/Slot_car

[4] Drift Hybrid Gaming (s.f.). Recuperado el 9 de mayo de 2021 de https://www.sturmkind.com/es/

[5] Hot Wheels A.I. Intelligent Race System (s.f.). Recuperado el 9 de mayo de 2021 de https://play.hotwheels.com/es-es/ai.html

[6] Qué es Arduino y cómo funciona. Agosto 3, 2020. Recuperado el 10 de mayo de 2021 de https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

[7] Entradas analógicas en Arduino. Septiembre 23, 2014. Recuperado el 11 de mayo de 2021 de https://www.luisllamas.es/entradas-analogicas-en-arduino/

[8] Actuadores y Sensores. Diciembre 11, 2018. Recuperado el 12 de mayo de 2021 de https://arduinofacil.com/actuadores-y-sensores/

[9] Diodo LED. (s.f.). Recuperado el 12 de mayo de 2021 de https://www.areatecnologia.com/electronica/como-es-un-led.html

[10] Motor CC. (s.f.). Recuperado el 12 de mayo de 2021 de https://automatismoindustrial.com/curso-carnet-instalador-baja-tension/motores/1-3-5-motores-de-corriente-continua/1-3-5-2-principios-de-funcionamiento/

[11] ¿Qué es y cómo funciona un servomotor? Diciembre 2, 2016. Recuperado el 12 de mayo de 2021 de http://panamahitek.com/que-es-y-como-funciona-un-servomotor/

[12] El módulo L298N. Julio 3, 2021. Recuperado el 12 de mayo de 2021 de https://descubrearduino.com/modulo-l298n/

[13] Alimentación Arduino. (s.f.). Recuperado el 12 de mayo de 2021 de https://aprendiendoarduino.wordpress.com/2016/11/09/alimentacion-arduino/

[14] Android e iOS dominan el mercado smartphone. Julio 30, 2021. Recuperado el 13 de mayo de 2021 de https://es.statista.com/grafico/18920/cuota-de-mercado-mundial-de-smartphones-por-sistema-operativo/

[15] Arduino Reference. (s.f.). Último acceso el 20 de mayo de 2021 de https://www.arduino.cc/

[16] Queue handling library (designed on Arduino). (s.f.). Recuperado el 9 de mayo de 2021 de https://github.com/SMFSW/Queue

[17] StackExchange. (s.f.). Último acceso el 22 de mayo de 2021. https://stackexchange.com/

[18] Eleego Starter kit tutorial (s.f.). Recuperado el 10 de mayo de 2021 de https://www.elegoo.com/blogs/arduino-projects/elegoo-uno-r3-project-the-most-complete-starter-kit-tutorial

[19] Qué es PWM y para qué sirve. Diciembre 19, 2017. Recuperado el 15 de mayo de https://www.rinconingenieril.es/que-es-pwm-y-para-que-sirve/

[20] Android Documentation. (s.f.). Último acceso el 24 de agosto de 2021 de https://developer.android.com/

[21] StackOverflow. (s.f.). Último acceso el 2 de septiembre de 2021 de https://stackoverflow.com/

[22] Modelo de comunicación maestro-esclavo. (Diciembre, 2015). Recuperado el 5 de Junio de 2021. https://riuma.uma.es/xmlui/bitstream/handle/10630/11444/A.J.%20Garc%C3%ADa%20Pineda_TFG.pdf

[23] Protocolo de comunicación I2C. (s.f.). Recuperado el 5 de Junio de 2021. https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/

[24] Conectar Arduino por Bluetooth. (Noviembre 27, 2015). Recuperado el 5 de Junio de 2021. https://www.luisllamas.es/conectar-arduino-por-bluetooth-con-los-modulos-hc-05-o-hc-06/

[25] Configuración módulo HC-05 mediante comando AT. (s.f.). Último acceso el 7 de Junio de 2021. https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usando-comandos-at.html

[26] Introducción general a Bluetooth. (Diciembre 27, 2019). Último acceso el 9 de junio de 2021. https://developer.android.com/guide/topics/connectivity/bluetooth?hl=es-419

[27] Android to control Arduino. (Abril 4, 2021). Último acceso el 9 de junio de 2021. https://www.electroniclinic.com/android-app-development-to-control-arduino-over-bluetooth-using-android-studio/

[28] Krug, S. (2010). *Haz fácil lo imposible.* Madrid, España: Anaya.

[29] Krug, S. (2019). *No me hagas pensar. Actualización.* Madrid, España: Anaya.

[30] Introducción a JSON. (s.f.). Recuperado el 12 de junio de 2021 de https://www.json.org/json-es.html

[31] Introducción a JAXB y el manejo de XML. Enero 23, 2017. Recuperado el 12 de junio de 2021 de https://www.arquitecturajava.com/introduccion-java-jaxb/

[32] Read JSON file with GSON. Enero 8, 2020. Último acceso el 16 de junio de 2021. https://www.bezkoder.com/java-android-read-json-file-assets-gson/

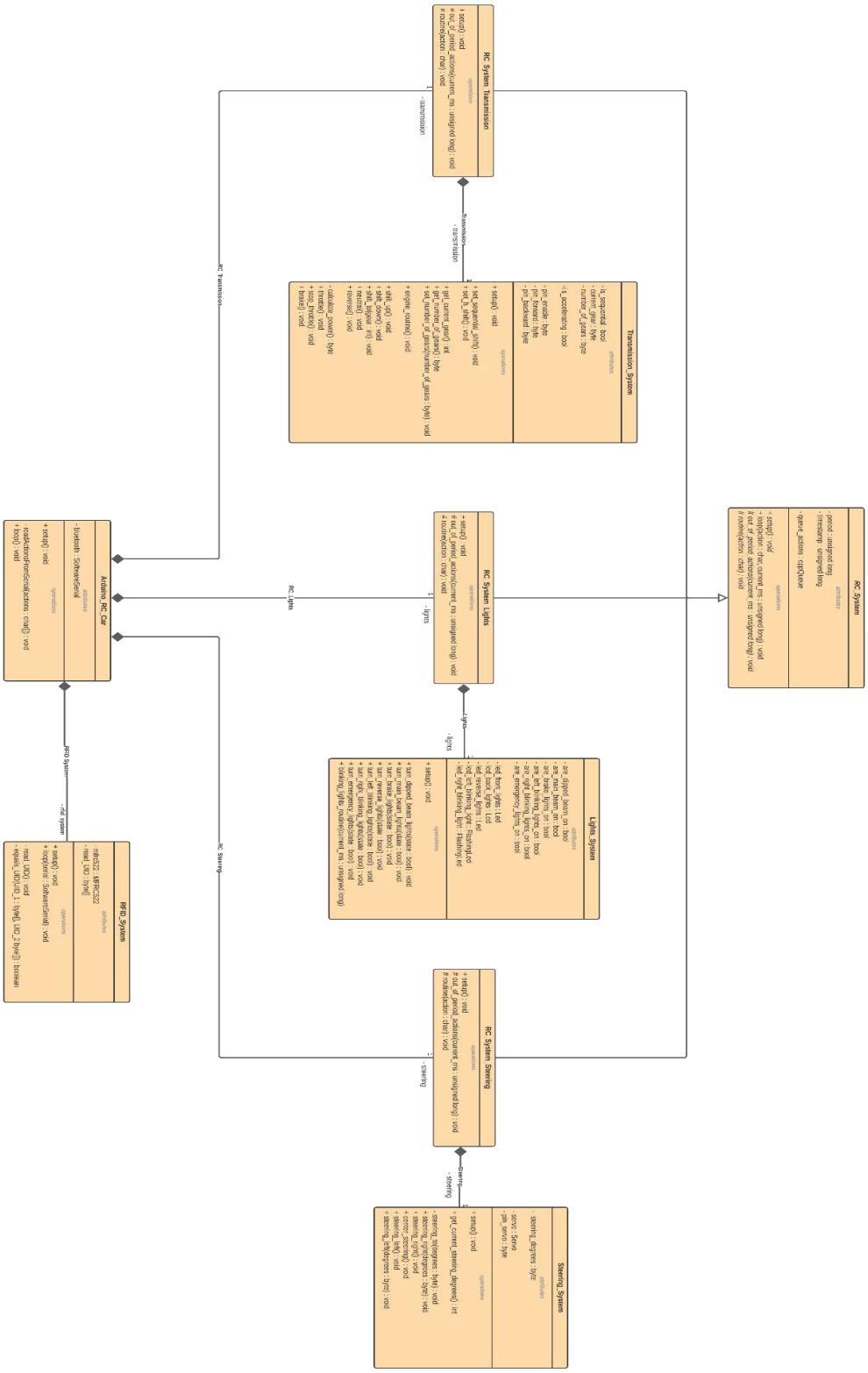
[33] Lectura de tarjetas RFID con Arduino. (Octubre 13, 2016). Último acceso el 13 de julio de 2021. https://www.luisllamas.es/arduino-rfid-mifare-rc522/

[34] Tutorial configuración RFID. (s.f.). Recuperado el 14 de julio de 2021. https://www.youtube.com/watch?v=LvRfxGTUEpE

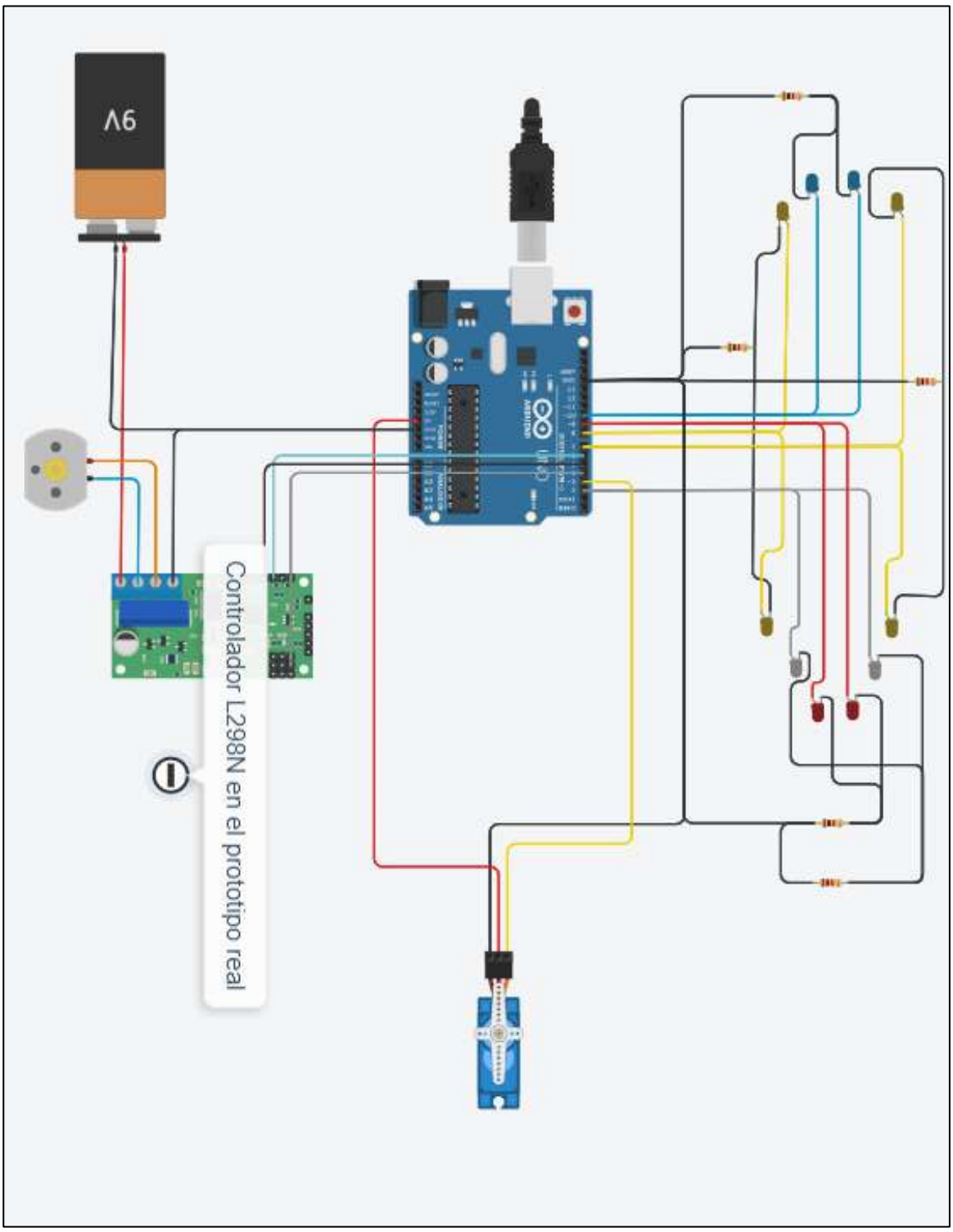
[35] Refactoring Guru. (s.f.) Último acceso el 20 de agosto de 2021. https://refactoring.guru/es/design-patterns

Anexos

I. Diagrama de clases de los sistemas del vehículo



II. Esquemático del coche radio control



III. Documentos usados en la primera prueba de usabilidad

Recuerde que NO SE LE ESTÁ EVALUANDO A USTED, sino a la usabilidad de las interfaces de nuestro sistema. Si se siente perdido o comete algún error, el fallo reside en el diseño del interfaz y usted nos ayudará a corregirlo.

Durante las tareas a realizar se le plantea una situación ficticia para situarle en un contexto "real" donde puede tener sentido el uso de nuestro sistema. Lea detenidamente por favor.

"Acaba de comprar nuestro producto y está ansioso por ver de qué es capaz el coche radiocontrol, por lo que quiere comprobar cómo se comporta y se maneja."

Tarea 1

Tarea: Entre en la aplicación y sin más preámbulos, adquiera el control de su coche y manéjelo libremente dando un par de círculos.

Cuestionario post-tarea:

1. ¿Ha sido fácil completar la tarea?
2. Comentarios:

Muy fácil	Fácil	Normal	Difícil	Muy difícil

Tarea 2

Ya ha probado los controles básicos del coche, pero ha visto que el producto se anunciaba como que propone varios tipos de manejo. Por esta razón, quiere investigar sobre qué se trata exactamente.

Tarea: Actualmente el modo de control se encuentra en *básico*, cambie el modo de control y pruébelo poniendo el coche en marcha.

Cuestionario post-tarea:

1. ¿Ha sido fácil completar la tarea?
2. Comentarios:

Muy fácil	Fácil	Normal	Difícil	Muy difícil

Tarea 3

Tarea: Pruebe el tipo de control que aún no ha probado.

Cuestionario post-tarea:

1. ¿Ha sido fácil completar la tarea?
2. Comentarios:

Muy fácil	Fácil	Normal	Difícil	Muy difícil

Tarea 4

Por último, una vez ha visto de todo lo que es capaz la aplicación, cree su propia configuración personalizada con los controles que más le gusten.

Tarea: Cree su propia configuración con los controles que más le hayan gustado.

Cuestionario post-tarea:

1. ¿Ha sido fácil completar la tarea?
2. Comentarios:

Muy fácil	Fácil	Normal	Difícil	Muy difícil

Post-Test

1. Creo que me gustaría usar con frecuencia esta aplicación móvil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

2. He encontrado la aplicación móvil innecesariamente compleja

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

3. Pienso que ha sido fácil utilizar la aplicación móvil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

4. He encontrado las diversas posibilidades de la aplicación móvil bien integradas

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

5. Pienso que había demasiada inconsistencia en la aplicación móvil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

6. Me sentí muy confiado en el manejo de la aplicación

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

7. He encontrado la forma de control básica cómoda de usar

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

8. He encontrado la forma de control profesional cómoda de usar

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

9. He encontrado la forma de control realista cómoda de usar

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

10. Me ha resultado sencillo crear mi propia configuración de controles

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

IV. JSON de un circuito ejemplo


```
{
  "type": "CIRCUIT",
  "hasSecret": false,
  "name": "Round Raceway",
  "description": "It's a simple racetrack. Only you have to do is turn
right",
  "size": "SMALL",
  "cells": [
    [
      {
        "x": 0, "y": 0, "order": 6,
        "pieces": [[
          {"x": 0, "y": 0, "piece": "CURVE", "rotation": 0}
        ]]
      },
      {
        "x": 1, "y": 0, "order": 1,
        "pieces": [[
          {"x": 0, "y": 0, "piece": "GOAL_STRAIGHT", "rotation": 0}
        ]]
      },
      {
        "x": 2, "y": 0, "order": 2,
        "pieces": [[
          {"x": 0, "y": 0, "piece": "CURVE", "rotation": 1}
        ]]
      }
    ],
    [
      {
        "x": 0, "y": 1, "order": 5,
        "pieces": [[
          {"x": 0, "y": 0, "piece": "CURVE", "rotation": -1}
        ]]
      },
      {
        "x": 1, "y": 1, "order": 4,
        "pieces": [[
          {"x": 0, "y": 0, "piece": "STRAIGHT", "rotation": 0}
        ]]
      },
      {
        "x": 2, "y": 1, "order": 3,
        "pieces": [[
          {"x": 0, "y": 0, "piece": "CURVE", "rotation": 2}
        ]]
      }
    ]
  ]
}
```

V. Documentos usados en la segunda prueba de usabilidad

Tareas

Recuerde que NO SE LE ESTÁ EVALUANDO A USTED, sino a la usabilidad de las interfaces de nuestro sistema. Si se siente perdido o comete algún error, el fallo reside en el diseño del interfaz y usted nos ayudará a corregirlo.

Durante las tareas a realizar se le plantea una situación ficticia para situarle en un contexto "real" donde puede tener sentido el uso de nuestro sistema. Lea detenidamente por favor.

"Hace poco que adquirió nuestro producto, ya probó cómo se manejaba el coche. Ahora quiere jugar a algún nivel de construcción."

Tarea 1

Tarea: Entre en la aplicación, y juegue a un desafío de construcción. Para empezar, el circuito 1 parece un sencillo.

Cuestionario post-tarea:

1. ¿Ha sido fácil completar la tarea?
2. Comentarios:

Muy fácil	Fácil	Normal	Difícil	Muy difícil

Tarea 2

Ya ha comprobado cómo funciona la aplicación le ayuda a construir un circuito. Pruebe a construir algo más complicado.

Tarea: Entre de nuevo en la aplicación, y juegue al desafío 2 de construcción.

Cuestionario post-tarea:

1. ¿Ha sido fácil completar la tarea?
2. Comentarios:

Muy fácil	Fácil	Normal	Difícil	Muy difícil

Post-Test

1. Creo que me gustaría usar con frecuencia esta aplicación móvil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

2. He encontrado la aplicación móvil innecesariamente compleja

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

3. Pienso que ha sido fácil utilizar la aplicación móvil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

4. He encontrado las diversas posibilidades de la aplicación móvil bien integradas

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

5. Pienso que había demasiada inconsistencia en la aplicación móvil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

6. Me sentí muy confiado en el manejo de la aplicación

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

7. He encontrado la forma de guiarme durante la construcción útil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

8. Me he sentido perdido durante algún momento de la prueba

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

9. He encontrado la información mostrada en cada paso de construcción útil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

10. Me ha resultado difícil entender qué se me pedía construir en cada paso

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

VI. Contenido del fichero con los desafíos de la aplicación


```
[
  {
    "id": 1, "racewayFile": "circle.json", "laps": 1,
    "scores": [
      {"requestTime": {"hours": 0, "mins": 1, "secs": 50, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 1, "secs": 30, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 1, "secs": 10, "millis": 0}}
    ], "challengesRequired": []
  },
  {
    "id": 2, "racewayFile": "normal.json", "laps": 2,
    "scores": [
      {"requestTime": {"hours": 0, "mins": 2, "secs": 30, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 2, "secs": 0, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 1, "secs": 30, "millis": 0}}
    ], "challengesRequired": [1, 2]
  },
  {
    "id": 4, "racewayFile": "drag_track.json", "laps": 1,
    "scores": [
      {"requestTime": {"hours": 0, "mins": 0, "secs": 50, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 0, "secs": 30, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 0, "secs": 15, "millis": 0}}
    ], "challengesRequired": []
  },
  {
    "id": 5, "racewayFile": "make_a_decision_track.json", "laps": 1,
    "scores": [
      {"requestTime": {"hours": 0, "mins": 1, "secs": 0, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 0, "secs": 40, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 0, "secs": 30, "millis": 0}}
    ], "challengesRequired": [4]
  },
  {
    "id": 6, "racewayFile": "two_turns_track.json", "laps": 1,
    "scores": [
      {"requestTime": {"hours": 0, "mins": 1, "secs": 0, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 0, "secs": 35, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 0, "secs": 20, "millis": 0}}
    ], "challengesRequired": [5]
  },
  {
    "id": 7, "racewayFile": "ultimate_track.json", "laps": 1,
    "scores": [
      {"requestTime": {"hours": 0, "mins": 1, "secs": 10, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 0, "secs": 50, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 0, "secs": 35, "millis": 0}}
    ], "challengesRequired": [6]
  },
  {
    "id": 8, "racewayFile": "ultimate_track_with_special_check.json",
"laps": 1,
    "scores": [
      {"requestTime": {"hours": 0, "mins": 1, "secs": 30, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 1, "secs": 0, "millis": 0}},
      {"requestTime": {"hours": 0, "mins": 0, "secs": 40, "millis": 0}}
    ], "challengesRequired": [6]
  }
]
```

VII. Documentos usados en la tercera prueba de usabilidad

Tarea

Recuerde que NO SE LE ESTÁ EVALUANDO A USTED, sino a la usabilidad de las interfaces de nuestro sistema. Si se siente perdido o comete algún error, el fallo reside en el diseño del interfaz y usted nos ayudará a corregirlo.

Durante la tarea a realizar se le plantea una situación ficticia para situarle en un contexto "real" donde puede tener sentido el uso de nuestro sistema. Lea detenidamente por favor.

"Hace unos días un amigo suyo le desafió a superar un nivel concreto de RaceYourTrack usando la configuración de control profesional y pasando por el check-point especial. Un desafío que ya había superado anteriormente pero usando el modo de conducción principiante, esta vez toca ver si es capaz de igualar la marca de su amigo."

Tarea 1

Tarea: Entre en la aplicación, y tras seleccionar la configuración de control que su amigo le ha pedido usar, dentro de los desafíos seleccione el tramo "Tramo con dos curvas" y juegue el desafío.

Cuestionario post-tarea:

1. ¿Ha sido fácil completar la tarea?
2. Comentarios:

Muy fácil	Fácil	Normal	Difícil	Muy difícil

Post-Test

1. Creo que me gustaría usar con frecuencia esta aplicación móvil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

2. He encontrado la aplicación móvil innecesariamente compleja

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

3. Pienso que ha sido fácil utilizar la aplicación móvil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

4. He encontrado las diversas posibilidades de la aplicación móvil bien integradas

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

5. Pienso que había demasiada inconsistencia en la aplicación móvil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

6. Me sentí muy confiado en el manejo de la aplicación

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

7. Me he sentido perdido durante algún momento de la prueba

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

8. Navegar por la aplicación me ha sido sencillo

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

9. Entender el estado actual de los desafíos me ha resultado complicado

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

10. Interpretar la información mostrada en los resultados del desafío ha sido fácil

En completo desacuerdo — Completamente de acuerdo

1	2	3	4	5

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA