UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Gandia

DISEÑO, MONTAJE Y PROGRAMACIÓN DE UN SISTEMA MÓVIL PARA REALIZAR CAPTURAS 360 Y GRABACIONES DE VÍDEO OMNIDIRECCIONAL, CONTROLADO POR UNA APLICACIÓN MÓVIL

Trabajo Fin de Grado

Grado en Tecnologías Interactivas

AUTOR/A: Peñaranda Jara, Jose Julio

Tutor/a: Boronat Segui, Fernando

CURSO ACADÉMICO: 2022/2023

PALABRAS CLAVE: ANDROID; ARDUINO; OMNIDIRECTIONAL VIDEO; VEHÍCULO MOTORIZADO; VIDEO 360

KEYWORDS: ANDROID; ARDUINO; OMNIDIRECTIONAL VIDEO; MOTORIZED VEHICLE; 360 VIDEO

Resumen:

En este proyecto se va a realizar el diseño, montaje y programación de un sistema móvil para realizar capturas y grabaciones de vídeo omnidireccionales (imágenes y vídeos 360), controlado a través de una aplicación móvil. El proyecto consta de 2 partes bien diferenciadas:

a. La primera parte consiste en el montaje de todos los componentes de un sistema móvil disponible en el grupo de I+D en el que se encuadra el proyecto (cuyo responsable es el tutor del proyecto). Dicho vehículo transportará un mástil en el que se colocará, a cierta altura, una cámara de video 360. Se ha diseñado una PCB (Printed Circuit Board o Placa de Circuito Impreso) que incluye un microcontrolador ESP32 en el que se ha programado el control de los 4 motores de las 4 ruedas del vehículo para poder manejar los movimientos del mismo. Además, se ha estabilizado el mástil al máximo para evitar los movimientos y vibraciones de la cámara.
b. La segunda parte consiste en el desarrollo de una aplicación para móvil con SO Android, que permite controlar, de forma táctil, los movimientos del vehículo a distancia, así como visualizar la captura de la cámara (tanto en un visor plano como en un visor estereoscópico), y tomar fotos y grabar vídeo 360. Además, se ha integrado en la aplicación el uso de un mando de control con botones conectado vía bluetooth con el móvil.

Summary:

This project aims to design, assemble, and program a mobile system for capturing and recording omnidirectional video (360 images and videos). The system will be controlled through a mobile application. The project consists of two distinct parts:

a. The first part involves assembling all the components of a mobile system available in the R&D group to which the project belongs (led by the project supervisor). This vehicle will carry a mast on which a 360-video camera will be placed at a certain height. A Printed Circuit Board (PCB) has been designed, which includes an ESP32 microcontroller programmed to control the 4 motors of the vehicle's 4 wheels to handle its movements. Additionally, the mast has been maximally stabilized to prevent camera movements and vibrations.
b. The second part involves developing an Android mobile application that allows remote control of the vehicle's movements through touch, as well as viewing the camera's capture (both in a flat viewer and a stereoscopic viewer) and taking photos and recording 360 videos. Furthermore, the application has been integrated with the use of a button-equipped control joystick connected to the mobile device via Bluetooth.

Índice

Índice de Figuras

Capítulo 1. Introducción

1.1 Presentación y objetivos

En este Trabajo Final de Grado (TFG, en adelante) se ha realizado el diseño, montaje y programación de un sistema móvil para realizar capturas y grabaciones de vídeo omnidireccionales (imágenes y vídeos 360), controlado a través de una aplicación móvil, mediante un móvil o tablet con SO Android, y también con un mando de control remoto conectado a dichos dispositivos vía Bluetooth. El TFG consta de 2 partes con objetivos principales bien diferenciados:

c. El objetivo principal de la primera parte consiste en el montaje de todos los componentes de un sistema móvil con 4 motores, disponible en el grupo de I+D en el que se encuadra el proyecto, Immersive and Interactive Media R&D group (IIM), cuyo responsable es el tutor del proyecto. Dicho vehículo transporta un mástil en el que se coloca, a cierta altura, una cámara de video 360. Se deberá diseñar una placa electrónica (PCB, Printed Circuit Board o Placa de Circuito Impreso) que incluirá un microcontrolador ESP32 en el que se programará el control de los 4 motores del vehículo para poder manejar sus movimientos. Además, se deberá estabilizar el conjunto al máximo para evitar los movimientos y vibraciones de la cámara, que pueden repercutir en la calidad del contenido capturado.
d. El objetivo principal de la segunda parte consiste en el desarrollo de una aplicación para móvil o tablet, con SO Android, que permitirá controlar, de forma táctil, los movimientos del vehículo a distancia, así como visualizar la captura de la cámara (tanto en un visor plano como en un visor estereoscópico), y capturar fotos 360 y grabar vídeo 360. Además, se ha integrado en la aplicación el uso de un mando de control remoto con botones, conectado vía bluetooth con el móvil o tablet.

Estos objetivos se dividen en varios subobjetivos:

Subobjetivos de la parte 1, relacionados con el diseño del sistema móvil:

- Debe poder adaptar su velocidad y dirección, de manera progresiva y controlada
- Debe detenerse si pierde la conexión Bluetooth.
- Debe poderse controlar mediante la aplicación móvil y también con algún tipo de mando a distancia.
- Debe moverse de forma que la cámara esté lo más estabilizada posible. Se deben reducir los temblores que lleguen a la parte superior del mástil donde se colocará la cámara
- Debe seguir la dirección especificada por el usuario en línea recta, sin desviarse o, al menos, permitir el ajuste de la dirección de forma fina mediante los controles de la aplicación o del mando.

Subobjetivos de la parte 2, relacionados con el diseño de la aplicación móvil:

- Debe permitir controlar los movimientos, la velocidad y la dirección del sistema móvil, así como detenerlo cuando se desee.
- Debe permitir apagar y encender los motores del sistema móvil.
- Debe permitir conectarse y desconectarse del sistema móvil vía Bluetooth.
- Debe permitir controlar la velocidad y aceleración del sistema móvil, así como corregir los desvíos de la dirección de movimiento.
- Debe permitir visualizar el streaming de video 360 de la cámara utilizada en el TFG (Insta 360).

- El contenido del vídeo debe poderse ver adecuadamente en un reproductor estereoscópico y en unas gafas de RV o HMD.

1.2 Estructura de la memoria

Esta memoria está organizada de la siguiente forma:

- En el capítulo 2, *Estado del arte*, se presenta una investigación exhaustiva de trabajos, proyectos y productos relacionados con el realizado en este TFG, comparándolos, con el fin de identificar las novedades y características únicas del TFG.
- En el capítulo 3, *Herramientas Hardware y Software*, se explican todas las herramientas y dispositivos, tanto a nivel de aplicación (software) como de montaje (hardware) necesarias para entender el trabajo realizado en el TFG, descrito en los capítulos siguientes.
- En el capítulo 4, *Desarrollo del proyecto*, se describe en detalle el funcionamiento, las características y la programación realizada del TFG, recogiendo todas las características implementadas en el mismo y sus funcionalidades. Además, también se analiza la estructura de trabajo seguida, la estructura de carpetas y cómo se han abordado las diferentes funcionalidades del código en los diferentes lenguajes. Este capítulo se divide en 2 partes:
    - 4.1) Montaje y programación en Arduino, en la que se describe el montaje del kit del sistema móvil utilizado, además de la programación del microcontrolador ESP32 que controla el sistema móvil.
    - 4.2) Aplicación móvil y uso de la cámara, en la que se describe la programación de la aplicación móvil en Android Studio.
- En el capítulo 5, *Conclusiones*, se presentan las conclusiones y posibles ampliaciones, así como otras mejoras que se pueden implementar (trabajo futuro), tanto en Arduino como en la aplicación móvil, para ampliar el control del sistema móvil.
- La memoria finaliza con la sección de *Bibliografía*, que incluye las referencias bibliográficas utilizadas durante la realización del TFG.
- Tras todo esto se encuentran los Anexos, el Anexo 1, *Manuales,* comprende diversos manuales importantes para la utilización del sistema móvil, tanto una guía de uso, una guía para grabar el vídeo 360, como un manual para instalar la aplicación y usar el sistema móvil
- En el Anexo 2, *Problemas de implementación resueltos durante el TFG,* se recogen diferentes problemas que han significado un desafío de programación o montaje de este TFG, además de contar cómo se han resuelto
- En el Anexo 3, *Listado de materiales para el montaje del proyecto,* se recogen los diferentes materiales que son necesarios para montar este TFG, además del presupuesto que supone.
- En el Anexo 4, *Tablas de variables,* se recogen todas las variables utilizadas tanto en el código del IDE de Arduino como en el código de Android Studio, junto a una pequeña descripción de cada variable.

Capítulo 2. Estado del arte

En este capítulo se describen diversos temas relacionados con el TFG. Estos temas son, en primer lugar, el concepto de Vídeo 360, tanto la manera de grabarse, como de reproducirse en diversos dispositivos. En este apartado se indagará también en los Head Mounted Display (HMD) y diversos reproductores de video 360.

En segundo lugar, se presenta una explicación de la teoría de control necesaria para entender el funcionamiento de los motores del sistema móvil, los cuales hacen uso de controladores, que serán explicados en este apartado

Además, se muestra un trabajo de investigación que compara este TFG con diversos proyectos relacionados, resaltando los puntos fuertes de cada uno y las diferencias entre ellos.

2.1 Vídeo 360

Grabación y visualización de Vídeo 360

Los videos 360, también conocidos como videos de realidad virtual o VR (Virtual Reality), son una forma de contenido audiovisual que permite a los espectadores explorar una escena en todas las direcciones utilizando tecnología de visualización panorámica. A diferencia de los videos tradicionales con una perspectiva fija, los videos 360 brindan una experiencia inmersiva en la que los espectadores pueden girar su vista en todas las direcciones, como si estuvieran presentes en el lugar de la grabación.

Estos videos se graban con cámaras 360, que son cámaras especiales equipadas con múltiples lentes, capturando una imagen en 360 grados simultáneamente. Luego, a través del software de edición, todas las imágenes se fusionan en una sola, creando un video panorámico completo. Los espectadores pueden reproducir estos videos en dispositivos compatibles, como smartphones, tabletas, computadoras o dispositivos de realidad virtual, y usar el movimiento o el toque para explorar la escena en cualquier dirección. A continuación, se muestra la imagen que capta una cámara 360 con 2 lentes:

Figura 2.1.1: Imágenes de una cámara 360 [Ref 10]

La cámara expuesta en la imagen solamente tiene 2 lentes, pero en la actualidad existen cámaras 360 con mayor número de lentes y que otorgan una mejor calidad de la imagen, como por ejemplo la Insta360 Pro, utilizada en este TFG. El desarrollo de cámaras 360 más compactas y

asequibles ha facilitado a los creadores de contenido producir videos 360 de alta calidad sin la necesidad de equipos costosos y complicados.

Después de capturar la imagen utilizando las dos lentes de la cámara 360, obtendremos un archivo de imagen con dos fotos redondas: una frontal y otra trasera.

En este punto, se realiza un proceso de fusión de la foto frontal y la foto trasera para crear una única imagen. Este proceso se conoce como Stitching (unión o ensamblaje), que consiste en combinar ambas imágenes de manera fluida y coherente, creando una sola imagen panorámica. El resultado se muestra a continuación:

Figura 2.1.2: Imagen resultado de una cámara 360 [Ref 10]

Naturalmente, para poder visualizar en 360 grados y de forma inmersiva la imagen extraída de la cámara, se utilizan visores o reproductores de video 360. Un reproductor de video 360 es una aplicación, software o plataforma que permite reproducir y visualizar contenido en formato de video 360 grados. Este tipo de reproductor está diseñado específicamente para proporcionar una experiencia inmersiva a los espectadores, permitiéndoles explorar la escena en todas las direcciones.

Cuando se reproduce un video 360, el reproductor utiliza algoritmos y técnicas de renderizado especiales para mostrar la imagen esférica o cilíndrica en una vista plana. La aplicación se encarga de desplegar la imagen en un entorno interactivo, donde los usuarios pueden mover su punto de vista utilizando el ratón, la pantalla táctil o los controles de un casco de realidad virtual. Esto permite explorar el video 360 en todas las direcciones, como si el espectador estuviera físicamente presente en el lugar de la grabación.

Además de esto, muchos videos 360 han sido grabados con tecnología de sonido espacial, lo que significa que el reproductor debe ser capaz de reproducir correctamente los efectos de sonido en función de la dirección de la mirada o punto de vista del espectador. Esto contribuye aún más a la inmersión y realismo de la experiencia.

Dispositivos de reproducción de video 360

Los dispositivos HMD (Head-Mounted Display), también conocidos como cascos de realidad virtual, son dispositivos diseñados para sumergir al usuario en una experiencia de realidad virtual, incluida la reproducción de videos en 360 grados. Estos dispositivos se colocan en la cabeza del usuario y cubren los ojos para proporcionar una visualización envolvente.

Los HMD están compuestos por 2 pantallas de alta resolución que ofrecen imágenes nítidas (una para cada ojo), lentes ópticas que enfocan y adaptan la imagen a la visión del usuario, sensores de seguimiento que capturan los movimientos de la cabeza y los ojos para hacer un seguimiento

del usuario y así actualizar la imagen en tiempo real, y auriculares integrados o externos que proporcionan una experiencia de audio envolvente para una inmersión completa en el video en 360 grados.

Hoy en día los HMD que podemos encontrar en el mercado se adaptan con facilidad a la cabeza del usuario para tener una experiencia cómoda e inmersiva. Además de proporcionar una imagen nítida y un seguimiento de los movimientos del usuario en tiempo real muy pulido. Este es el caso de las Meta Quest 2, que además incluyen mandos para interactuar con las diferentes aplicaciones y juegos que ofrecen estas gafas.

Figura 2.1.5: Meta Quest 2 [Ref 24]

Además de los HMD, también se pueden visualizar vídeos en 360 grados en pantallas planas, de PCs, tablets o smartphones. Son muchas las aplicaciones que ya hacen uso de vídeos 360 como YouTube, Vimeo o Facebook. Debido a que los dispositivos móviles actuales cuentan con sensores como acelerómetros o giroscopios, éstos pueden usarse de manera eficiente para hacer un seguimiento de los movimientos del usuario y brindar una experiencia inmersiva a la hora de reproducir vídeos en realidad virtual.

De hecho, existen numerosos casos de dispositivos HMD que sirven para colocar el dispositivo móvil en ellos y poder visualizar los vídeos en 360 como si el usuario estuviera usando unas gafas de realidad virtual. Un ejemplo de esto son las gafas VRBOX o las Google Cardboard.

Para poder visualizar el contenido en 360 mediante el uso de un dispositivo móvil, es imprescindible que la aplicación en cuestión cuente con un reproductor estereoscópico, es decir, que divida la pantalla del dispositivo en dos partes, una para cada ojo. A continuación, se muestra un ejemplo de un visor estereoscópico desarrollado por Google.

Figura 2.1.6: Visor de Google Cardboard [Ref 26]

Proyectos y artículos relacionados con video 360

Para finalizar este subapartado y resaltar la importancia del video 360 en el panorama tecnológico y de desarrollo actual, se van a describir una serie de proyectos y artículos relacionados con el uso de Video 360.

Proyecto 360ViSi de Quasar Dynamics [Ref 19]

El proyecto europeo 360ViSi, iniciado en enero de 2019, reúne a universidades de diversos países junto a la empresa desarrolladora de video 360 Quasar Dynamics, para explorar la implementación de innovación disruptiva en la educación. Entre los socios se encuentran la Universidad de Stavanger en Noruega, la Turku AMK de Finlandia y la Universidad Católica de Valencia de España. El enfoque principal del proyecto es el desarrollo de vídeo 360 interactivo para crear prácticas pedagógicas inmersivas de alto valor. El objetivo final es proporcionar una plataforma educativa que permita a los usuarios aprender en primera persona, sin importar su ubicación, utilizando solo un dispositivo móvil junto a un dispositivo HMD.

El objetivo de este proyecto es crear prácticas educativas reales y de alto valor pedagógico utilizando la tecnología de Realidad Virtual. El vídeo inmersivo 360 proporciona una experiencia inmersiva y envolvente que permite a los estudiantes explorar entornos virtuales y participar en situaciones educativas de manera más realista e interactiva.

SPHERISION [Ref 20]

Este proyecto de investigación audiovisual de la Universitat Politècnica de Mataró fue creado con el objetivo de simular una experiencia visual de 360 grados en varios lugares emblemáticos de Barcelona. Mediante la utilización de seis cámaras de bajo costo colocadas en una montura omnidireccional, se capturaron todos los ángulos de visión de los espacios grabados. Esto permitió crear un vídeo promocional de la ciudad en el cual el espectador puede explorar el entorno y realizar un recorrido virtual por el espacio filmado. Esta investigación contribuye a contextualizar y comprender las características y potencialidades del vídeo 360º como una forma innovadora de visualización inmersiva.

Reproductor web para vídeo omnidireccional con HbbTV [Ref 21]

Este proyecto de la Universitat Politècnica de València consiste en desarrollar una aplicación web compatible con el estándar Hybrid Broadcast Broadband TV, que permite la conexión y sincronización de dispositivos secundarios con el televisor, incluyendo un escenario inmersivo y un reproductor de vídeos 360°. El objetivo principal es proporcionar una experiencia de consumo más personalizada e inmersiva al transmitir contenido relacionado, incluyendo vídeos omnidireccionales, a través de streaming IP para ser reproducidos en dispositivos externos de forma sincronizada con el televisor.

El proyecto también se ha enfocado en el descubrimiento e intercambio de mensajes con el televisor híbrido para sincronizar la reproducción del contenido. Se ha desarrollado un reproductor de contenido de video 360° que permite la visualización en modo monoscópico en pantallas convencionales, así como en modo estereoscópico para una experiencia más inmersiva utilizando visores o gafas de realidad virtual. Además, se han seleccionado y preparado contenidos apropiados para la emisión tanto por transmisión broadcast como por transmisión a través de Internet.

2.2 Teoría de control

El control es un proceso mediante el cual se regula y dirige el comportamiento de un sistema, proceso o variable para alcanzar un estado deseado o mantenerlo dentro de límites establecidos. Es una disciplina fundamental en la ingeniería y otras áreas relacionadas, ya que permite mantener y optimizar el desempeño de sistemas en una amplia gama de aplicaciones, como, por ejemplo, automatización industrial, robótica, control de procesos o control de vehículos.

El control es un proceso que implica tres elementos fundamentales de manera continua y en tiempo real. Primero, se lleva a cabo la medición precisa del estado actual del sistema. A continuación, se compara este estado medido con un estado de referencia o deseado, previamente establecido, al cual se pretende llegar. Por último, se generan y aplican acciones de control adecuadas para corregir las desviaciones existentes y regular el sistema hacia el estado deseado. Este ciclo se repite de forma constante y en tiempo real para lograr un control efectivo y dinámico que se ajuste a las necesidades y requisitos específicos del sistema.

En el contexto del control, la entrada de un sistema se conoce como referencia. La referencia representa el valor deseado que se desea lograr en una o más variables de salida del sistema a lo largo del tiempo. El objetivo del control es asegurarse de que las variables de salida sigan de cerca esta referencia.

Para lograr esto, se utiliza un controlador, que es un dispositivo o algoritmo diseñado para manipular la entrada del sistema de manera adecuada. El controlador toma en cuenta la diferencia entre la referencia y la salida real del sistema, que se conoce como error. Con base en este error, el controlador genera señales o acciones de control que se aplican al sistema para influir en su comportamiento y lograr el efecto deseado en las variables de salida.

La realimentación juega un papel crucial en el control. La realimentación es el proceso mediante el cual se utiliza información de la salida del sistema para ajustar la entrada o las acciones de control.

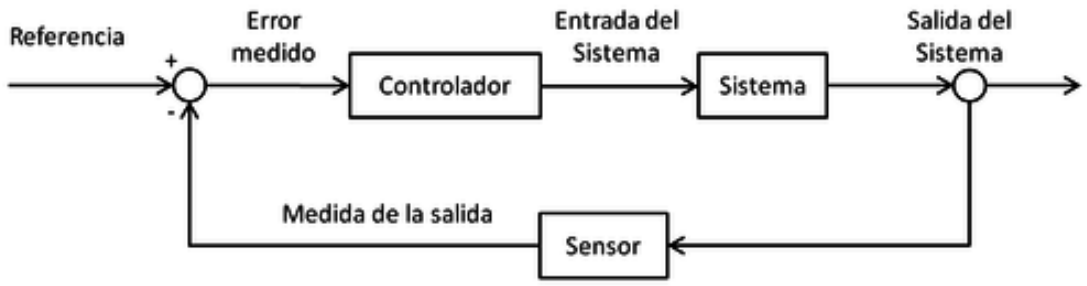


Figura 2.2.1: Lazo de control con realimentación

En el campo del control, existen diferentes tipos de controladores utilizados para lograr implementar estrategias de control. Entre los controladores más comunes se encuentran los controladores Proporcional (P), Integral (I) y Derivativo (D), que a menudo se combinan para formar controladores PI, PD o controladores PID completos.

El controlador proporcional (P) es un componente fundamental en los sistemas de control y es utilizado para corregir el error entre la referencia y el valor medido del sistema. Su principal objetivo es ajustar la acción de control de manera proporcional al error actual. Cuando se aplica un controlador proporcional, el error es calculado como la diferencia entre la referencia y el valor medido del sistema. A continuación, este error es multiplicado por una constante

proporcional (Kp), que determina la sensibilidad o la respuesta del controlador a las desviaciones.

El controlador proporcional es eficaz para reducir el error inmediato y minimizar las desviaciones entre la referencia y el valor medido. Sin embargo, puede presentar algunas limitaciones, como, por ejemplo, no ser capaz de eliminar completamente el error en estado estacionario, es decir, las desviaciones persistentes en la salida del sistema. Por lo que la señal emitida puede ¿no? llegar nunca a alcanzar el valor deseado.

Un controlador integral (I) es otro tipo de controlador utilizado en sistemas de control. A diferencia del controlador proporcional, el controlador integral toma en cuenta la acumulación de errores pasados y ajusta la acción de control en función de esta acumulación.

El controlador integral se utiliza para eliminar los errores de estado estacionario que no pueden ser corregidos por un controlador proporcional. El controlador integral tiene la capacidad de eliminar este error en estado estacionario ya que la acumulación de errores a lo largo del tiempo lleva a un aumento en la acción de control. Esto implica que cuanto mayor sea el error acumulado, mayor será la corrección aplicada al sistema.

Sin embargo, es importante tener en cuenta que introducir un controlador integral a un sistema pude conllevar una serie de complicaciones. Si el valor de Ki es demasiado alto, el controlador integral puede causar una respuesta excesiva, lo que puede resultar en oscilaciones o incluso inestabilidad del sistema.

En este TFG se ha utilizado un controlador PI, que combina las características de los 2 controladores anteriormente descritos:

Un controlador proporcional-integral (PI) es un tipo de controlador que combina las características del controlador proporcional y del controlador integral. El controlador PI se utiliza para corregir tanto el error inmediato como los errores en estado estacionario.

La acción de control de un controlador PI se calcula sumando la acción de control proporcional y la acción de control integral. La ecuación básica para un controlador PI es:

$$\text{Acción de control} = Kp * \text{error} + Ki * \int(\text{error})\, dt$$

Donde Kp es la constante proporcional y Ki es la constante integral. El primer sumando de la ecuación representa la acción de control proporcional, que ajusta la salida en función del error actual, mientras que el segundo es la acción de control integral, que tiene en cuenta la acumulación de errores pasados.

La notación ∫(error) dt indica que estamos integrando la función error con respecto al tiempo, lo que implica calcular el área bajo la curva del error en el tiempo. Para calcular la integral del error, en programación, se multiplica en cada instante el error en el instante actual por el periodo de muestreo y al producto se le suma la salida en el instante anterior, de la siguiente forma:

$$\int(\text{error})\, dt = T * \text{error} + \text{salida\_anterior};$$

El controlador proporcional responde rápidamente a las desviaciones actuales, mientras que el controlador integral corrige los errores en estado estacionario acumulando los errores pasados. Al combinar estos dos componentes, el controlador PI logra una respuesta rápida y también elimina los errores en estado estacionario.

Como se puede apreciar en la siguiente gráfica, un controlador proporcional-integral corrige el error en estado estacionario, a diferencia del controlador proporcional puro (El controlador proporcional-integral se muestra en rojo y el proporcional se muestra en amarillo):

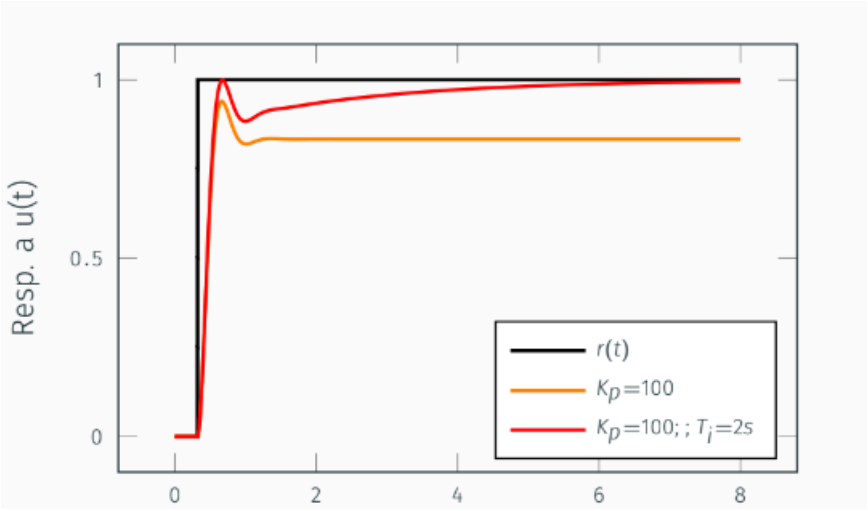


Figura 2.2.2: Gráfica de un controlador PI [Ref 16]

Al emplear un controlador PI en lugar de un PID completo (El cual añade la constante derivativa, difícil de gestionar sobre todo en sistemas con varios motores), se reduce la carga computacional requerida, lo que permite un mejor aprovechamiento de los recursos limitados. Esto resulta especialmente relevante en aplicaciones en tiempo real donde se requiere una respuesta rápida y eficiente del controlador.

2.3 Proyectos relacionados con el desarrollo de sistemas móviles y captura de datos

A continuación, se van a describir una serie de proyectos relacionados con este TFG, y que presentan algunas similitudes con el mismo en el ámbito de desarrollar un sistema móvil o robot controlado a distancia, el cual captura y muestra al usuario una serie de datos, como imágenes o vídeo.

2.3.1. Proyecto 1: Programación de robot móvil para vigilancia con visión artificial [Ref 1]

En [Ref 1] se presenta un proyecto que consiste en la programación y control de un robot móvil para la vigilancia de un espacio determinado, mediante el uso de visión artificial y un sistema de control remoto. El robot está equipado con una cámara 2D que le permite obtener imágenes 2D del entorno en tiempo real, y cuenta con un sistema de procesamiento de imágenes que le permite detectar objetos y movimientos en su campo visual.

Este robot debe actuar como un portero de vigilancia, teniendo registrados en su base de datos a las personas que pueden entrar en un área restringida. En el sistema de base de datos se registra el nombre del usuario junto con fotografías tomadas desde diferentes ángulos. Cuando el robot identifica a un usuario que se encuentra en la base de datos, le concede el acceso y lo guía en su trayecto.

El robot es controlado mediante el ordenador del usuario, el cual debe estar conectado vía Wifi o Bluetooth al robot. Para el control del robot se ha desarrollado un programa en C# con diversos botones para indicar al mismo la dirección que debe tomar.

Figura 2.3.1: Aplicación de control Proyecto 1 [Ref 1]

El hardware utilizado en este proyecto incluye un robot móvil equipado con motores y ruedas para su desplazamiento, una cámara para la captura de imágenes 2D, y una placa controladora que permite el control remoto del robot y la programación de su comportamiento.

El software utilizado en el proyecto está realizado con el lenguaje de programación Python y utiliza bibliotecas de visión artificial, como OpenCV, que permiten el procesamiento de imágenes y la detección de objetos en tiempo real. La finalidad principal de este proyecto es la vigilancia y seguridad de un espacio determinado mediante la utilización de tecnologías de automatización y visión artificial, con el fin de mejorar la eficiencia y eficacia en la detección de posibles amenazas o peligros.

Figura 2.3.2: Foto del sistema móvil Proyecto 1 [Ref 1]

2.3.2. Proyecto 2: Montaje, calibración y programación del robot SR1 [Ref 2]

El robot SR1 es un sistema autónomo de exploración submarina, capaz de tomar muestras y recolectar datos del fondo marino. Este proyecto se enfoca en el diseño y construcción de un robot submarino (SR1) con capacidad de movimiento en seis grados de libertad (6DOF), un sistema de control remoto mediante mando a distancia y una gran variedad de sensores junto con el uso del programa “Dirección Norte” para que el robot cree sus propias rutas a seguir cuando se le indica que siga una dirección en concreto (Dirección Norte Geográfico) evitando obstáculos.

El robot está equipado con un sistema de propulsión y un conjunto de sensores y actuadores que permiten la exploración y toma de muestras. El robot SR1 tiene tres modos de control, se puede controlar mediante el mando a distancia “SONYIR”, haciendo uso de señales infrarrojas, puede ser controlado mediante un programa de ordenador haciendo uso del monitor serie, el cual, también mostrará los datos que capte el robot.

El hardware utilizado incluye una carcasa resistente al agua, un conjunto de motores de corriente continua, un sistema de propulsión con hélices, y un conjunto de sensores, como un sensor de profundidad y un sensor de temperatura. El software utilizado incluye programas,

desarrollados en lenguaje C, que permiten la programación del robot para operación autónoma y el control manual del mismo.

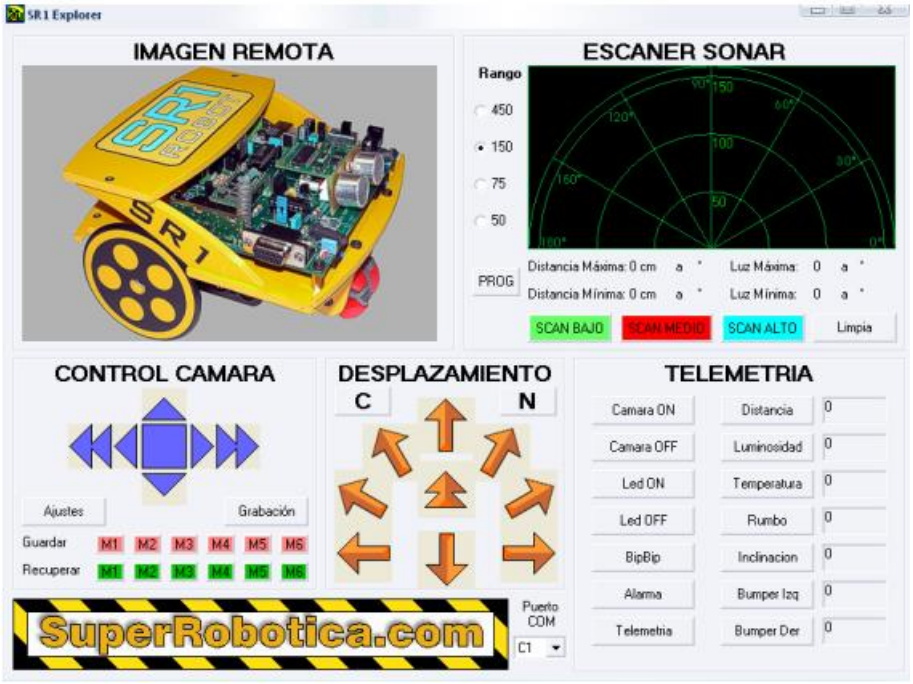


Figura 2.3.3: Aplicación de control Proyecto 2 [Ref 2]

2.3.3. Proyecto 3: Diseño de un robot móvil para estudio de la ocupación en habitaciones [Ref 3]

Este proyecto tiene como objetivo principal ampliar los campos de la robótica en la supervisión de entornos. El propósito es desarrollar un robot móvil autónomo capaz de detectar personas en diferentes habitaciones por las que se desplace. Esto permitirá estimar el número de personas detectadas en cada habitación, así como el tiempo necesario para detectar a todas las personas y el tiempo requerido para recorrer cada habitación.

Este proyecto proporcionará información relevante para la toma de decisiones. Por ejemplo, puede ayudar en la reorganización del personal en diferentes salas, planificar las tareas que el robot debe realizar según la ocupación de cada habitación en un horario determinado o simplemente ser utilizado como sistema de vigilancia en lugares donde no se tenga permitido el acceso a ciertas personas.

El robot cuenta con un microcontrolador Arduino que le permite realizar el recorrido autónomo siguiendo la pared izquierda de las habitaciones mediante sensores ultrasónicos para evitar colisiones. Es importante resaltar que el robot no cuenta con sistema de control remoto, sino que se guía por los datos recopilados por sus sensores, permitiéndole explorar su entorno de manera autónoma.

El robot detecta las personas que se encuentran en una habitación mediante el uso de una cámara conectada a la Raspberry Pi incorporada en el mismo. La cámara permite la detección de rostros para controlar quién se encuentra en cada habitación, si son demasiadas personas y si alguna habitación se encuentra vacía o con poca gente.

El hardware utilizado en el proyecto incluye un microcontrolador, un conjunto de sensores, como un acelerómetro y un giroscopio, y un conjunto de motores de corriente continua para la

propulsión del robot. El software utilizado incluye programas desarrollados en lenguaje C, que permiten el procesamiento de datos y la movilidad del robot.

Figura 2.3.4: Foto del sistema móvil Proyecto 3 [Ref 3]

2.4 Proyectos relacionados con el control de sistemas vía Android

Los siguientes proyectos se relacionan con este TFG en el ámbito del desarrollo de sistemas automatizados que utilicen Android, tanto para permitir el control a sus usuarios como para la recepción de vídeo

2.4.1. Proyecto 4: Diseño e implementación de múltiples interfaces para el control remoto de un vehículo y del sistema de visión embarcado [Ref 4]

Se trata de un TFG en el que se ha desarrollado un proyecto que consiste en un sistema automatizado de reconocimiento de matrículas de vehículos que puede ser controlado de forma remota. Este sistema tiene como objetivo principal localizar y reconocer vehículos en función de sus placas de matrícula en un amplio espacio, además de ofrecer otras funcionalidades adicionales como la vigilancia. El reconocimiento se realiza a través de una plataforma móvil que permite el control remoto, con capacidad de visión por ordenador y transmisión de video en tiempo real al puesto de control.

Se han desarrollado varias opciones de control de la plataforma que pueden adaptarse a diferentes condiciones del terreno o a las preferencias personales del operador. Estas opciones incluyen el control mediante un mando inalámbrico de Xbox 360, el control basado en movimientos y el control táctil a través de una aplicación Android en un smartphone. A continuación, se muestra una captura de pantalla de la pantalla para controlar la plataforma mediante control táctil.

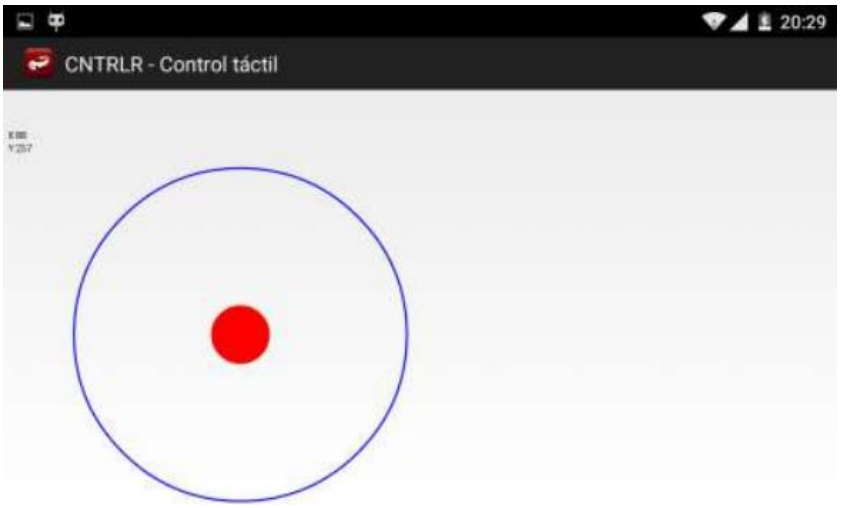


Figura 2.3.5: Screenshot de la aplicación en la pantalla de movimiento mediante uso de la pantalla táctil del Proyecto 4 [Ref 4]

Como se puede apreciar en la imagen, el usuario de la aplicación móvil cuenta con un joystick integrado en la propia pantalla de la aplicación el cual sirve para controlar el robot, orientándolo y cambiando su dirección.

Para controlar la plataforma mediante los movimientos del dispositivo móvil por medio del acelerómetro integrado, se hace uso de la pantalla de la aplicación que se muestra a continuación:

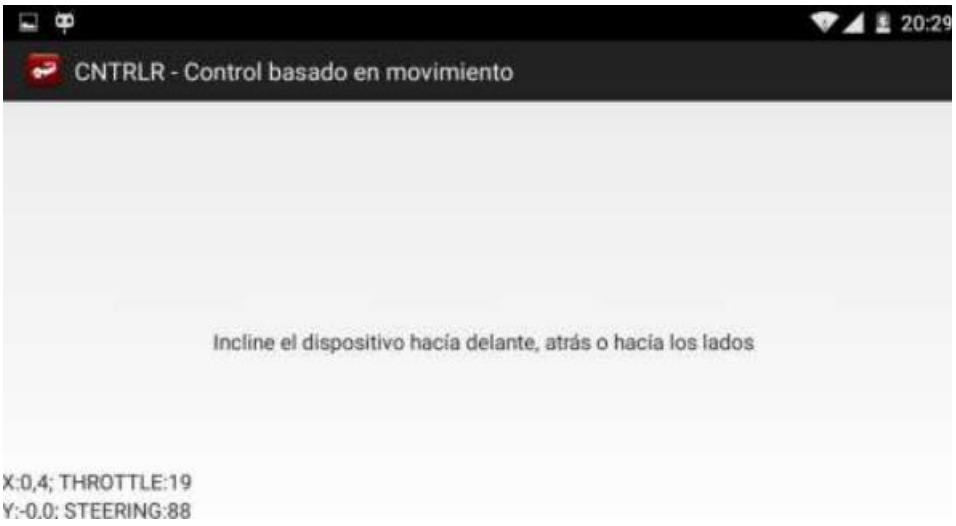


Figura 2.3.6: Screenshot de la aplicación en la pantalla de movimiento mediante uso de acelerómetro del Proyecto 4 [Ref 4]

Las señales de control son transmitidas al sistema instalado en el vehículo, el cual desempeña tres funciones principales: enviar el video capturado por la cámara del vehículo al operador en tiempo real, realizar el reconocimiento de matrículas y reenviar las señales de control recibidas al controlador integrado encargado de mover el vehículo.

En este proyecto, el software utilizado incluye un conjunto de programas desarrollados en Python, que permiten el procesamiento de datos y la comunicación entre los diferentes dispositivos como son, el mando, la aplicación móvil y el propio robot. El objetivo principal de este proyecto es ofrecer una solución integrada y eficiente que pueda llevar a cabo múltiples operaciones simultáneamente, centrándose en el reconocimiento óptico de matrículas y el control remoto del robot

Figura 2.3.7: Foto del sistema móvil Proyecto 4 [Ref 4]

2.4.2. Proyecto 5: Diseño y desarrollo de un sistema de videovigilancia basado en tecnología Android [Ref 5]

El proyecto descrito en [Ref 5] se enfoca en desarrollar una aplicación de videovigilancia para dispositivos móviles utilizando el sistema operativo Android. El objetivo principal es permitir a los vigilantes de seguridad monitorizar cámaras de seguridad de forma remota a través de sus dispositivos Android, como smartphones o tablets. La aplicación proporciona acceso, en tiempo real, a las imágenes de las cámaras, permitiendo el control y manejo de estas. Además, se explora la funcionalidad de detección de movimiento y notificación al vigilante en caso de intrusión.

El sistema de videovigilancia utiliza tecnologías de visión por computadora y aprendizaje automático para detectar y reconocer objetos en tiempo real, lo que permite la detección temprana de eventos anormales como intrusos o comportamientos extraños y la toma de decisiones en tiempo real como, por ejemplo, notificar al usuario. Las cámaras del sistema de videovigilancia pueden ser manejadas mediante el uso del joystick incorporado en la aplicación móvil.

Mediante la aplicación móvil, los empleados del recinto que utilicen este sistema podrán visualizar el estado del recinto simplemente visualizando las cámaras y controlándolas a distancia. El vídeo de las cámaras es enviado mediante RTSP a los dispositivos móviles que tengan instalados la aplicación de manera que se esté viendo el flujo de vídeo en tiempo real.

2.5 Comparativa

Este TFG implica el diseño y montaje de componentes electrónicos y mecánicos, así como la programación de un microcontrolador para controlar el movimiento del vehículo (dirección, velocidad y aceleración del vehículo) y de una aplicación para dispositivos móviles basados en SO Android para controlar de forma remota al vehículo (a través de Bluetooth) y reproducir y grabar el video360 transmitido por la cámara de vídeo 360 incluida en el sistema. Además, dicha aplicación debe permitir que el contenido del video 360 pueda ser visualizado en un visor o bien plano o estereoscópico, para poder utilizar gafas de RV o HMD.

Sin embargo, los proyectos descritos en el apartado 2.3, si bien proporcionan soluciones para sistemas móviles controlados a distancia que recolectan y envían datos como pueden ser imágenes, vídeos e incluso otros tipos de datos del fondo marino, ninguno proporciona una aplicación para dispositivos móviles en la que se pueda ver el vídeo en directo mientras se controla el sistema móvil en un dispositivo móvil.

En cuanto a los proyectos descritos en el apartado 2.4, los cuales hacen uso de Android para manejar y monitorizar sus sistemas, ninguno llega a explorar la emisión de vídeo en directo en 360 grados. El proyecto descrito en [Ref2] también utiliza un mando a distancia para el control del sistema móvil, pero en este caso, es un control de Xbox.

El sistema presentado en este TFG es más ambicioso y complejo que los sistemas desarrollados en los proyectos descritos en este capítulo, en términos de diseño, montaje y programación de componentes electrónicos y mecánicos para controlar un vehículo móvil y capturar videos omnidireccionales, además de ser el único que ofrece vídeo en directo 360.

A continuación, se muestra una tabla comparativa que relaciona el sistema desarrollado en este TFG con el resto de los proyectos relacionados descritos en este capítulo:

	Sistema desarrollado en el TFG	**Proyecto 1**	**Proyecto 2**	**Proyecto 3**	**Proyecto 4**	**Proyecto 5**
Incluye un sistema móvil a control remoto	Si	Si	Si	No cuenta con control remoto	Si	No
Aplicación para dispositivos Móviles	Sí, para dispositivos móviles con SO Android	No, uso de pantalla LCD	No, uso de mando a distancia SONY y control remoto mediante PC	No, se visualizan las imágenes mediante el PC	Sí, para dispositivos móviles con SO Android	Sí, para dispositivos móviles con SO Android
Uso de cámara de vídeo	Cámara de vídeo IP Insta 360	Cámara IP inalámbrica TV-IP672WI	Cámara IP Conectada por Wifi con	EyeToy Namtai conectada a una Raspberry Pi,	Cámara con interfaz CSI-2. Conectada por Wifi	Cámaras IP conectadas a la aplicació

		Conectada por Wifi	servomotor	conectada por Wifi al PC		n móvil vía WiFi
Streaming de vídeo (protocolo)	Sí (RTMP)	Sí HLS	Si (No se especifica protocolo)	No, solo captura imágenes y video	Sí (RTP)	Sí (RTSP)
Video 360	Sí (visor estereoscópico y reproductor ExoPlayer)	No	No	No	No	No
Control de velocidad y dirección	Sí de ambos mediante dispositivo móvil (Android)	Sólo de dirección mediante PC	De ambos, mediante sensores y aplicación para PC y mando a distancia Bluetooth	No cuenta con control remoto, se controla de manera autónoma	De ambos, de velocidad mediante sensores y dirección por app para móvil	No incluye sistema móvil
Tecnologías de Comunicación	Bluetooth para controlar el sistema y WiFi para recibir el video en directo	Serie RS-232, inalámbrica WI-FI y Bluetooth para control de dirección	Señal infrarroja para detectar obstáculos	Señal infrarroja para detección de obstáculos y WiFi para transmisión de video	WiFi para control de dirección y visualización de video	WiFi para mover las cámaras y uso de RTSP para visualizar el video en directo
Microcontrolador	ESP32 Feather	PMB5010	BasicX24	Arduino UNO	Arduino con Raspberry Pi	No utiliza al no incluir un sistema móvil al que controlar

Capítulo 3. Análisis de requerimientos Hardware y Software

En este capítulo, se realizará un análisis detallado de los requerimientos tanto hardware como software para el desarrollo de un sistema móvil destinado a la grabación de videos en formato 360. El sistema se compone de varios elementos clave, incluyendo cuatro motores para el movimiento del dispositivo, una placa ESP32 Feather como controlador principal, y una cámara Insta 360 para capturar el video panorámico. Además, se ha desarrollado una aplicación móvil en Android para el control remoto del sistema móvil y la reproducción del video en directo, por lo tanto, se incidirá también en el programa de Android Studio. Este análisis exhaustivo de los requerimientos hardware y software proporcionará una base sólida para entender la selección de componentes adecuados, la implementación del software necesario y la integración exitosa de todos los elementos involucrados.

3.1 Hardware utilizado

Componentes mecánicos del sistema móvil

En esta parte se describen los diferentes componentes que forman el sistema móvil, tanto los motores como el chasis de metal, y demás componentes utilizados para su correcto funcionamiento a la hora de desplazarse.

Chasis del sistema móvil

En el sistema móvil desarrollado en este TFG se ha utilizado el chasis de coche/tanque de seguimiento inteligente T900[1], que es un chasis resistente, hecho de una aleación de aluminio que le confiere gran durabilidad y resistencia. A pesar de ser un chasis de aluminio, resulta más liviano y fácil de manejar que si fuese de metal. Este chasis pesa alrededor de 2400g, lo que lo hace adecuado para proyectos de tamaño medio. A continuación, se muestra una foto del chasis.

Figura 3.1.1: Chasis de aluminio el sistema móvil

En el chasis original se incluía 4 motores DC de 12 Voltios, los cuales dieron problemas y fueron sustituidos por otros más nuevos.

[1] Referencia chasis: https://es.aliexpress.com/i/32869806689.html

Mástil

En el sistema móvil se ha utilizado, como mástil de sujeción de la cámara de vídeo 360, una pata de mesa de 78 cm de longitud y 2,7 cm de diámetro, adquirida en Leroy Merlín, junto con acoples adicionales de 15 cm de longitud para ganar altura. Al utilizar una pata de mesa como base del mástil, se aprovecha su resistencia y robustez, mientras que los acoples permiten adaptar la altura del mástil a las necesidades específicas del sistema. Esta combinación de elementos provee una solución económica y práctica para soportar y mantener en una determinada posición elevada a la cámara de vídeo 360.

Hache doble metálica y tensores

Al colocar la cámara 360 en una posición tan elevada, durante el desplazamiento del sistema móvil, se provocaban muchos temblores en ésta, impidiendo una visualización correcta y confortable del vídeo en directo transmitido por la cámara. Es por ello que se decidió utilizar cables metálicos o tensores que reduzcan las vibraciones o temblores sufridos por la cámara.

Por otro lado, para garantizar la estabilidad y resistencia del sistema móvil, al tensar los cables, se tuvo que colocar una estructura metálica de acero inoxidable (cortesía de la empresa Bronces Jordá, S.L., en forma de hache doble en la parte inferior del mismo. Esta elección estratégica ha sido fundamentada por su capacidad para prevenir deformaciones y soportar eficientemente el peso de los diversos componentes que conforman el sistema móvil. La estructura en forma de hache doble proporciona una base sólida y fiable, evitando el riesgo de flexión y asegurando la integridad estructural del dispositivo.

A continuación, se muestra una foto de la estructura metálica en forma de hache doble utilizada:

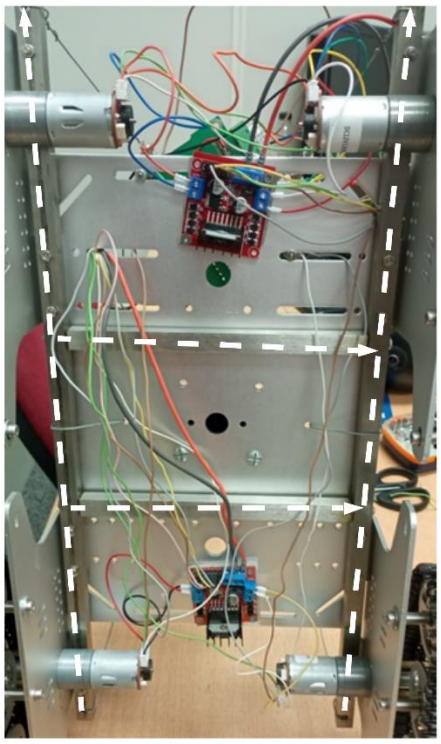


Figura 3.1.2: Hache doble metálica y conexiones

Drivers en H L298N

Los módulos de placa de controlador L298N[2] son componentes electrónicos diseñados para controlar motores de corriente continua y motores paso a paso. Se utilizan ampliamente en proyectos de robótica y automatización. El módulo L298N actúa como un puente H, que permite controlar la dirección y la velocidad del motor mediante señales de entrada.

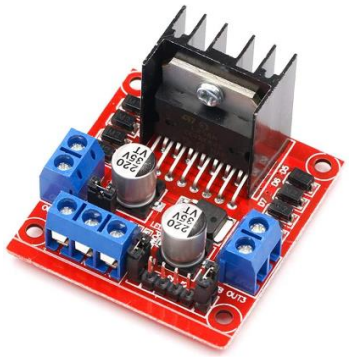


Figura 3.1.3: Driver L298N

El módulo L298N puede proporcionar la potencia y el control necesarios a los motores para mover el sistema móvil en diferentes direcciones y velocidades. Este controlador es utilizado para controlar motores de corriente continua de alta potencia. Puede manejar corrientes más altas y proporcionar una mayor potencia de salida para motores más grandes.

En este TFG se utilizan 2 de estos módulos o drivers en H, colocados en la parte inferior del chasis del sistema móvil, conectado cada uno a un par de motores (delanteros o traseros), ambos se conectan a los pines de 12V y GND de la PCB que se ha diseñado.

Los módulos L298N son utilizados en este TFG para alimentar cada par de motores (delanteros o traseros) con la misma señal de alimentación de 12V. Cada par de entradas (IN1 e IN2 o IN3 e IN4) se conectan a diversos pines de la ESP32 que emitirán la señal PWM para que cada motor tome la velocidad adecuada para que el sistema móvil siga una dirección y velocidad adecuadas. Esta señal PWM llega a los motores mediante las salidas OUT, y cada par de salidas (OUT1 y OUT2 o OUT3 y OUT4) se conectan al positivo y negativo de cada motor, respectivamente.

[2] Referencia L298N: https://components101.com/modules/l293n-motor-driver-module

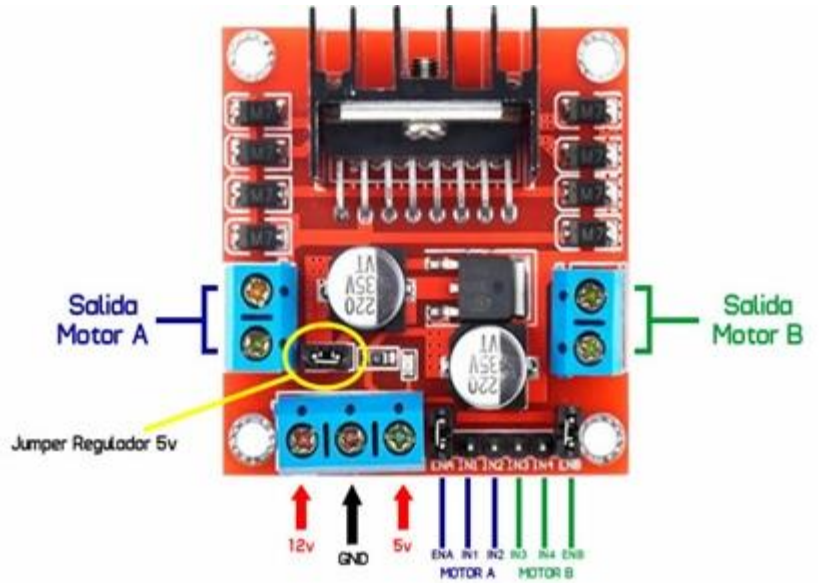


Figura 3.1.4: Entradas y salidas driver L298N

Motores DC

Los 4 motores DC utilizados en cada eje de rueda son los motores JGA-25-370[3]. Se trata de dispositivos electromecánico que convierte la energía eléctrica en energía mecánica rotativa. Tienen una velocidad nominal de 130 RPM (revoluciones por minuto) y funcionan con un voltaje nominal de 12V. Además, cuentan con un encoder de 2 canales que permite medir su velocidad con gran precisión.

Figura 3.1.5: Motor DC con encoder

Los pines del motor son los siguientes:

- Fuente de alimentación positiva del motor (12V).
- Negativo de la fuente de alimentación del encoder (GND)
- Canal A del encoder.
- Canal B del encoder.

[3] Referencia motores: https://media.digikey.com/pdf/Data%20Sheets/Seeed%20Technology/114090046_Web.pdf

- Positivo de la fuente de alimentación del encoder (5V).
- Fuente de alimentación negativa del motor (GND).

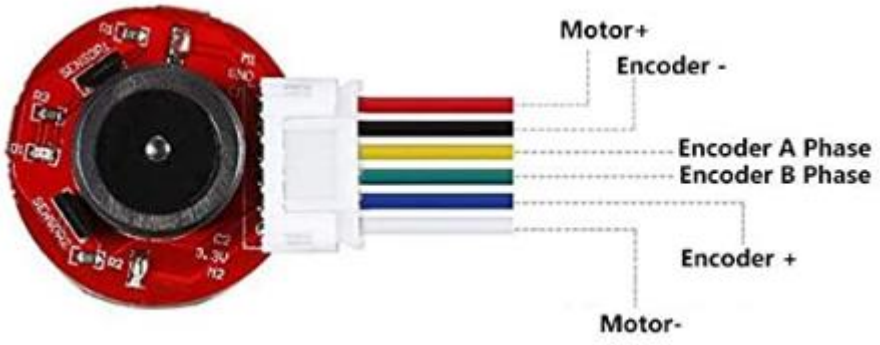


Figura 3.1.6: Conexiones Motor DC con encoder

El encoder o codificador de los motores utilizados sirve para tener un registro de la velocidad angular que lleva cada motor específico en cada momento. Este tipo de encoder consta de un disco o anillo codificado magnéticamente montado en el eje del motor y un sensor de efecto Hall que detecta los cambios en el campo magnético generado por el disco.

Cuando el motor gira, el disco codificado magnéticamente también gira junto con el eje. El disco tiene secciones magnéticas y no magnéticas dispuestas en un patrón específico. A medida que el disco gira, las secciones magnéticas y no magnéticas pasan por el sensor de efecto Hall. El sensor de efecto Hall está ubicado cerca del disco y detecta los cambios en el campo magnético a medida que las secciones del disco pasan por él. El sensor genera señales eléctricas proporcionadas a los canales de salida del encoder.

Los dos canales de salida del encoder (de 2 canales), generalmente denominados canal A y canal B, proporcionan señales “cuadrature” (Señales cuadradas que están desplazadas en fase entre sí). Cuando el disco gira en una dirección, se generan dos señales cuadrature en los canales A y B con una relación de desfase de 90 grados entre ellas.

Las señales cuadrature se utilizan para determinar la dirección de rotación y el desplazamiento angular del motor. Al analizar la secuencia de pulsos en los canales A y B, se puede determinar si el motor está girando en sentido horario o antihorario, así como la velocidad angular que lleva.

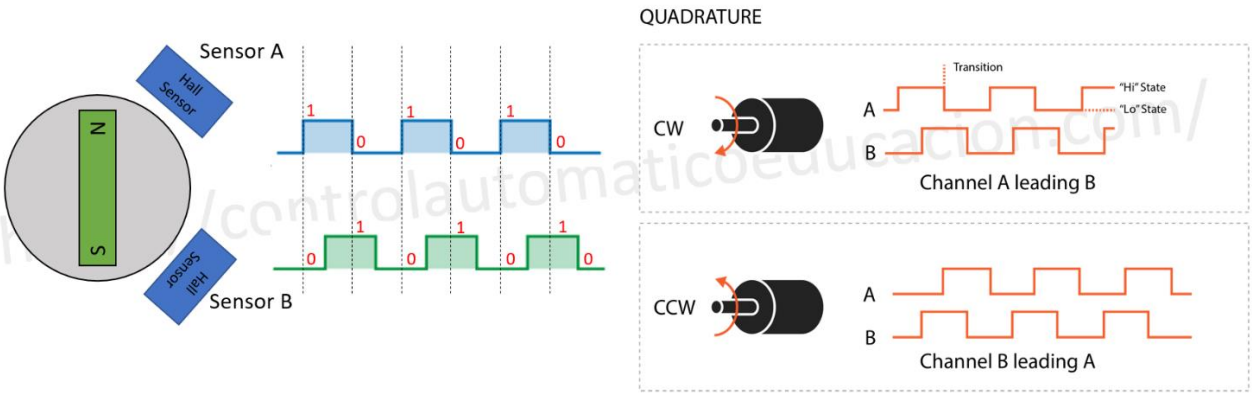


Figura 3.1.6: Esquema de funcionamiento de un encoder [Ref 11]

Cámara 360 utilizada

Cámara 360 Insta360Pro

La cámara 360 Insta360 Pro [Ref 7] es un dispositivo de alta gama diseñado para capturar video y fotos en 360 grados. Su principal característica son las seis lentes de ojo de pez de alta calidad que pueden grabar en resolución 8K, lo que permite obtener imágenes nítidas y detalladas. Además, la cámara cuenta con estabilización de imagen de 6 ejes para producir videos y fotos suaves y estables. Otra ventaja es la posibilidad de grabar en formato RAW y ajustar la imagen en postproducción para obtener una mayor flexibilidad. La cámara también incluye herramientas avanzadas de posproducción integradas en el programa para ordenador de la propia cámara para la edición de video y fotos en 360 grados, y se puede controlar mediante una aplicación para dispositivos móviles.

En este TFG el vídeo que captura la cámara es enviado vía RTMP directamente al cliente, en este caso, el dispositivo móvil del usuario, el cual se debe conectar al servidor de la cámara vía IP a través de la red WiFi local.

El Protocolo de Mensajería en Tiempo Real (RTMP, Real Time Messaging Protocol) es un protocolo de red utilizado para controlar el flujo de medios en tiempo real, como audio y video, a través de una red IP. Es un componente clave de la difusión en directo por su capacidad de mantener conexiones con baja latencia.

Figura 3.1.7: Cámara Insta360 Pro [Ref 7]

Controladores del sistema móvil

ESP32-Feather

La ESP32-Feather de Adafruit es una placa de desarrollo compacta y versátil basada en el microcontrolador ESP32. Esta placa ofrece una gran cantidad de características útiles para el desarrollo de proyectos de IoT y dispositivos electrónicos. La placa cuenta con dos núcleos de procesamiento a 240 MHz que permiten un rendimiento rápido y eficiente, así como conectividad inalámbrica WiFi y Bluetooth para facilitar la comunicación con otros dispositivos. También ofrece una amplia variedad de puertos y conectores, incluyendo USB, microSD, y pines

GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General), para permitir una mayor flexibilidad en la conexión y control de periféricos.

Además, la ESP32-Feather es compatible con el lenguaje de programación Arduino y otras herramientas de desarrollo, lo que hace que sea fácil para los desarrolladores comenzar a trabajar en proyectos rápidamente. La placa también cuenta con una batería de litio recargable y un cargador incorporado para facilitar la alimentación y la portabilidad, así como soporte para actualizaciones de firmware a través de la conexión USB.

Mando de control

Mando VRBOX

El Mando inalámbrico Bluetooth VRBOX es el controlador de las gafas de realidad Virtual VRBOX en las que se introduce un dispositivo móvil para experiencias de realidad virtual. Este mando a distancia se conecta vía Bluetooth al dispositivo móvil del usuario, quedando registrado como dispositivo de entrada.

De esta forma, al pulsar los diversos botones del mando o mover el joystick, manda una señal del tipo KeyEvent al dispositivo móvil, el cual actuará mandando la orden correspondiente al sistema móvil para moverse en una dirección u otra, o frenar.

Figura 3.1.8: Mando Bluetooth VRBOX

3.2 Software utilizado

Programas utilizados

Arduino

El software de Arduino se basa en el lenguaje de programación C y proporciona una plataforma fácil de usar para programar microcontroladores. Esto hace que sea más fácil para los programadores experimentados en C adaptarse a la plataforma. Además, la plataforma de Arduino incluye herramientas útiles para el desarrollo de proyectos, como el Serial Plotter, que permite la visualización en tiempo real de señales generadas por el microcontrolador. El Serial Plotter es una herramienta especialmente útil para el desarrollo de proyectos de robótica y electrónica, ya que permite a los usuarios analizar y depurar señales en tiempo real.

Por otro lado, la librería BluetoothSerial.h es una librería que permite la comunicación inalámbrica Bluetooth entre un dispositivo móvil y una placa Arduino. Esta librería es una

herramienta útil para aquellos que desean incorporar la conectividad Bluetooth en sus proyectos y simplifica la configuración y el control de la comunicación inalámbrica.

A continuación, se listan las librerías utilizadas en el código en Arduino:

- Arduino.h Esta librería es fundamental y se incluye de forma automática en todos los proyectos de Arduino. Proporciona las funciones y definiciones básicas necesarias para el funcionamiento del microcontrolador
- ESPmDNS.h. Esta librería permite asignar nombres de host a los dispositivos y facilita su identificación en una red local sin necesidad de conocer las direcciones IP.
- BluetoothSerial.h. Esta librería permite establecer conexiones y transferir datos entre el Arduino y otros dispositivos compatibles con Bluetooth, como dispositivos móviles u ordenadores.

Android Studio

Android Studio es un entorno integrado de desarrollo (IDE) para el desarrollo de aplicaciones móviles para el sistema operativo Android. Proporciona una plataforma fácil de usar para el desarrollo de aplicaciones móviles y ofrece herramientas útiles para la depuración y el despliegue de aplicaciones en diferentes dispositivos.

Por otro lado, las librerías SDK de Google VR proporcionan herramientas para el desarrollo de aplicaciones de realidad virtual en Android. La librería Multidex permite a las aplicaciones Android superar el límite de tamaño de archivos de aplicaciones y mejorar la compatibilidad con versiones antiguas de Android.

Exoplayer [Ref 6] es un conjunto de librerías que ofrecen un reproductor multimedia completamente personalizable para Android. Este reproductor permite visualizar tanto vídeo como streaming en diversos protocolos como HLS, HTTP, RTSP o RTMP. En este TFG se ha utilizado para poder reproducir el streaming de la cámara mediante protocolo RTMP

A continuación, se listan las dependencias utilizadas para el desarrollo del código de la app:

- El SDK de Google VR: 'com.google.vr:sdk-base:1.190.0': Librería que se utiliza para desarrollar aplicaciones de realidad virtual (VR) utilizando la plataforma de Google.
- Las librerías de Exoplayer:
    - 'com.google.android.exoplayer:exoplayer:2.17.1': Es la dependencia principal de ExoPlayer, que proporciona la funcionalidad principal del reproductor multimedia.
    - 'com.google.android.exoplayer:extension-rtmp:2.17.1': Es una extensión de ExoPlayer que agrega soporte para la transmisión de video en tiempo real a través del protocolo RTMP.
    - 'com.google.android.exoplayer:exoplayer-core:2.17.1': Es una dependencia central de ExoPlayer que contiene las clases principales y los componentes fundamentales del reproductor multimedia.
    - 'com.google.android.exoplayer:exoplayer-dash:2.17.1': Es una extensión de ExoPlayer que agrega soporte para la reproducción de contenido DASH (Dynamic Adaptive Streaming over HTTP).
    - 'com.google.android.exoplayer:exoplayer-ui:2.17.1': Es una extensión de ExoPlayer que proporciona componentes de interfaz de usuario prediseñados y personalizables para la reproducción multimedia.

- La librería Multidex: ‘androidx.multidex:multidex:2.0.1’: Esta librería permite superar el límite de métodos de la máquina virtual de Java al construir aplicaciones grandes.

Protocolos utilizados

RTMP (Real-Time Messaging Protocol)

RTMP (Real-Time Messaging Protocol) es un protocolo de transmisión en tiempo real desarrollado por Macromedia (ahora Adobe) para la entrega de contenido multimedia en la web. Proporciona una forma eficiente de transmitir audio y video en vivo a través de internet. RTMP se basa en una conexión persistente entre un servidor de medios y un cliente, lo que permite una comunicación bidireccional y una transmisión continua de datos. Utiliza el protocolo de transporte TCP (Transmission Control Protocol) para asegurar una entrega confiable de los paquetes de datos.

El funcionamiento de RTMP se basa en la división de los flujos de audio y video en pequeños fragmentos llamados "chunks". Estos chunks se envían desde el servidor al cliente, y el cliente los decodifica y reproduce en tiempo real. RTMP ofrece varios tipos de mensajes, como mensajes de control para establecer la conexión y controlar la reproducción, y mensajes de datos para transmitir los chunks de audio y video. Además, RTMP admite la compresión de datos para reducir la carga de ancho de banda y la sincronización de los datos multimedia para garantizar una reproducción fluida.

El funcionamiento del protocolo RTMP (Real-Time Messaging Protocol) para la transmisión en tiempo real de contenido multimedia a través de Internet se resume en dos pasos principales. Primero, la señal de audio o video codificada se transmite desde los codificadores de streaming al servidor RTMP utilizando TCP como protocolo de transporte. Durante el proceso de conexión, se realiza un handshake entre el cliente y el servidor para establecer la versión del protocolo y la marca de tiempo. Una vez establecida la conexión, el cliente envía una solicitud de conexión al servidor, quien responde para confirmarla.

Una vez establecida la conexión persistente, los datos de streaming se transmiten en bloques de diferentes tamaños, generalmente 128 bytes para video y 64 bytes para audio. La baja latencia de RTMP se logra gracias al uso de TCP. Desde el servidor RTMP también se puede transmitir el flujo directamente a los dispositivos finales mediante HLS.

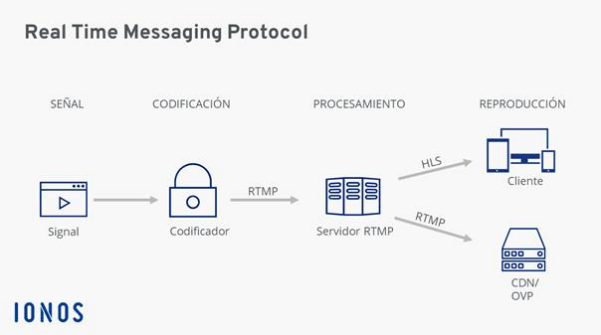


Figura 3.2.1: Funcionamiento RTMP [Ref13]

Capítulo 4. Desarrollo del proyecto

Para entender mejor el desarrollo del proyecto hay que tener en cuenta que ha habido 2 partes del desarrollo de este que se han llevado a cabo de forma paralela. Por un lado, el montaje del sistema móvil y su control en Arduino y por el otro lado, todo lo referente a la aplicación móvil y el uso de la cámara 360 para la visualización/grabación del contenido de vídeo 360.

Para el desarrollo de este proyecto no se ha empezado desde cero, sino que ya se contaba con una primera versión del circuito general del sistema móvil, realizada hacía varios años bajo el programa Makers de la UPV, aunque no estaba totalmente acabada y el conexionado no era el correcto. También se contaba con una versión mejorada del programa en Arduino para el control de los motores realizada el año anterior por un colaborador contratado en el grupo de I+D, que ha sido la base a partir de la cual se ha creado el programa finalmente utilizado, modificando la mayoría de las partes del código.

El código de todo el TFG se distribuye en la siguiente estructura de carpetas. Todo el código en Arduino utilizado para programar el microcontrolador ESP32 se encuentra en la carpeta Arduino_TFG, concretamente en el archivo Arduino_TFG.ino que se encuentra en su interior y que contienen programadas todas las funcionalidades del sistema móvil como tal.

A la misma altura que la carpeta Arduino_TFG, se encuentra la carpeta App_TFG, la cual contiene todos los archivos necesarios para compilar una aplicación Android estándar. Con una carpeta App, que contiene la carpeta Manifest, con el fichero Android_Manifest.xml (que contiene información importante de la aplicación, como su nombre, permisos requeridos, componentes y configuraciones), una carpeta Java (con las diferentes clases para ejecutar las actividades de la aplicación) y una carpeta Res (donde se guarda el contenido visual y de la aplicación como imágenes y pantallas, además de la traducción de la app al inglés y castellano). A la misma altura que la carpeta app, se encuentra la carpeta Gradle, que contiene los archivos y configuraciones relacionados con el sistema de construcción Gradle, que se encarga de compilar, construir y gestionar las dependencias de la aplicación.

La estructura de las carpetas anteriormente descrita se muestra a continuación:

- Arduino_TFG (Arduino)
    - Arduino_TFG.ino
- App_TFG (App)
    - App
        - Manifests
            - Android_Manifest.xml
        - Java
            - BLReceiver.java
            - ConnectionLostCallback.java
            - HomeActivity.java
            - MainActivity2.java
        - Res
            - Drawable
            - Layout
                - Activity_home.xml
                - Activity_home.xml (land)

        - Activity_main2.xml
        - Custom_control_view.xml
    - Values
        - Strings.xml
        - Themes
            - Themes.xml
- Gradle
    - Build.gradle (Project)
    - Build.gradle (Module)

Dado que el proyecto se compone de dos partes bien diferenciadas, es necesario dividir este capítulo en 2 apartados principales. Por un lado, el primero apartado incluye todo lo referente al montaje de todos los componentes de un vehículo móvil y la programación en Arduino del control del mismo. Por otro lado, el segundo está dedicado a explicar la programación de la aplicación para dispositivos móviles con SO Android y el control por Bluetooth, así como la conexión al servidor RTSP y la integración del reproductor del vídeo 360 siendo transmitido por la cámara mediante streaming vía IP.

4.1 Montaje y programación en Arduino

Montaje y electrónica del sistema móvil

Para el montaje del sistema móvil se parte del kit descrito en el capítulo anterior ya adquirido por el grupo de I+D en el pasado.

El diseño original del sistema móvil es el siguiente: 4 motores de 12V mueven los engranajes que, a su vez, moverán las cadenas que permiten el movimiento del vehículo.

Estos motores, a su vez, miden la velocidad a la que están girando mediante los encoder que llevan instalados que se alimentan a 5V. El control de la velocidad de los 4 motores se lleva a cabo mediante el microcontrolador ESP32 incluido en el sistema móvil.

Los encoder de los 4 motores se conectan directamente a la ESP32, sin embargo, la alimentación de estos motores pasa por los drivers en H L298N, se utiliza un driver en H para cada par de motores.

A las salidas de estos drivers se conectan los cables positivo y negativo de cada motor y las entradas de los drivers, de la 1 a la 4 se conectan a la alimentación, la 1 y 3 al positivo y la 3 y 4 al negativo.

Este control se realiza mediante un controlador PI implementado en la ESP32 recibiendo la señal de los encoder, midiendo la velocidad y aplicando el voltaje necesario a los pines de salida que alimentan a los motores en cada instante mediante un control proporcional integral para que el sistema móvil consiga la velocidad necesaria. Los parámetros de velocidad y aceleración son variables, ajustables por el usuario a través de la aplicación móvil desarrollada.

El montaje del sistema móvil ha pasado por diversas fases hasta su versión final.

En una fase inicial, todo el cableado estaba montado en una Protoboard. En esta fase se hicieron todas las pruebas de electrónica necesarias para entender el circuito, hasta que, finalmente, se consiguió hacer que funcionase. En la versión inicial se utilizaban 2 baterías distintas para alimentar el circuito: una batería de 12 Voltios alimentaba los 4 motores del sistema móvil; y otra batería de 5 Voltios se encargaba de alimentar los encoders de los motores y la ESP32. En esta versión se utilizó una ESP32-WROOM.

Cabe destacar que cuando se me entregó el sistema móvil, el cableado estaba a medio conectar, con cables rotos y estropeados, los cuales han sido cambiados. No obstante, en los documentos y esquemas creados anteriormente para el desarrollo de este sistema móvil había diversos errores en cuanto a correspondencia de pines y voltajes, dicha documentación se ha vuelto a hacer desde cero. Ahora en la fase final del proyecto el cableado del sistema móvil es mucho más limpio, con cables nuevos y crimpados con materiales nuevos. Sumado a esto se ha creado una PCB para que el cableado resulte mucho más sencillo y limpio, la cual ha sido desarrollada con el apoyo de profesores y técnicos del Campus utilizando el programa Altium.

Tras esta primera fase, para que la electrónica quedase mucho más limpia y profesional, se pasó a una segunda versión definitiva del cableado en la que hubo varios cambios. Entre ellos se cambiaron diversos componentes como los motores del sistema móvil y se cambiaron con L298N por unos nuevos del mismo modelo, además se diseñó una placa de circuito impreso o PCB para conectar todos los cables y la ESP32.

Para el diseño de la PCB, se utilizó un programa llamado Altium [Ref 17] para realizar el diseño de la forma más sencilla posible. El esquema de conexiones sufrió varios cambios, debido a que cambiamos la ESP32-WROOM por otro modelo ESP32-Feather, ya que este nuevo controlador tiene un diseño más compacto y además tiene mayor capacidad de almacenamiento flash. En este caso, se utiliza una única batería para alimentar a los motores, a la ESP32 y a los encoders, gracias a un conversor de tensión integrado en la propia PCB. A continuación, se muestra el esquemático del circuito definitivo de la PCB.

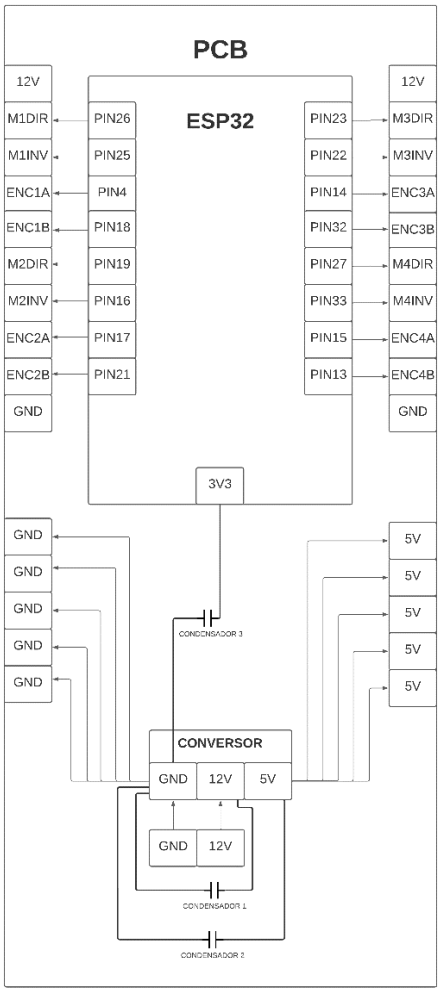


Figura 4.1.1: Esquema de conexiones PCB

A continuación, se muestra un esquema de la PCB con las conexiones en la capa top en blanco y las conexiones en la capa bottom en negro

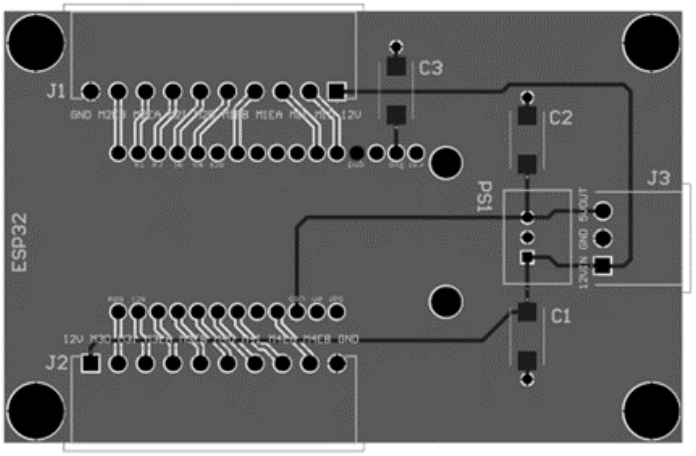


Figura 4.1.2: Esquema de capas TOP (blanco) y BOTTOM (negro) de la PCB

Para poder alimentar todo el circuito con una sola batería de 12V se instaló en la PCB un conversor de tensión que pasaba el voltaje de 12V a 5V para poder alimentar correctamente los encoders y la ESP32.

Para el cableado de esta segunda versión se utilizó cable multifilar flexible de 6 colores diferentes, crimpados a unos conectores de 6 vías (ver figura 4.1.4) para conectarse a los motores. Por el otro extremo, los cables se fueron crimpados con otro tipo de terminales para poder encajar en los conectores soldados en la PCB (figura 4.1.3).

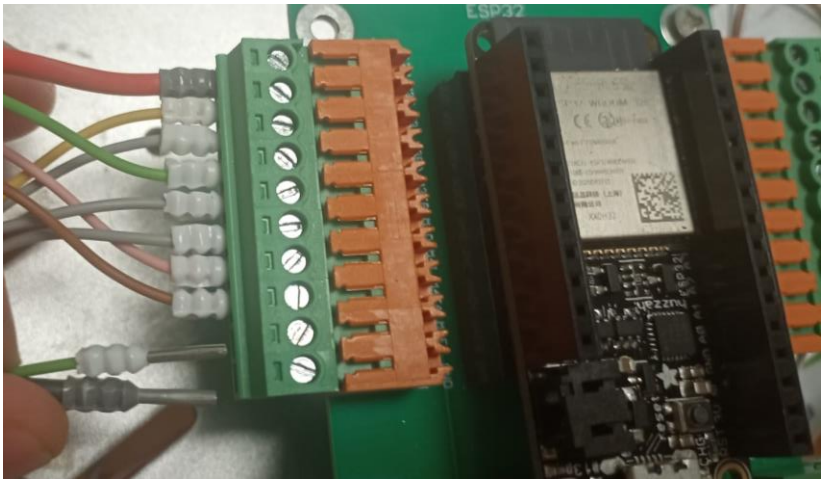


Figura 4.1.3: Terminales y cables crimpados de la PCB

En esta versión ya definitiva del cableado, las entradas de los drivers en H L298N [Figura 3.1.5] están conectadas a la PCB y las salidas de éstos están conectadas al positivo y negativo de cada motor de la siguiente forma: las entradas pares se conectan al negativo de cada motor y las entradas impares al positivo de cada motor Además, debido a fallos en uno de los encoders de los motores, los 4 motores fueron cambiados por unos nuevos, también con encoder y alimentados a 12 Voltios. El inconveniente de cambiar los motores fue que el orden de sus pines era distinto, por lo que se tuvo que reordenar los cables que conectaban cada motor con la PCB, lo que significó utilizar más cable y crimparlo para poder conectar los motores a la PCB de manera correcta. En la figura expuesta a continuación se muestra, a la izquierda, el orden de los cables de los motores nuevos y, a la derecha, el orden de los cables de los motores anteriormente utilizados:

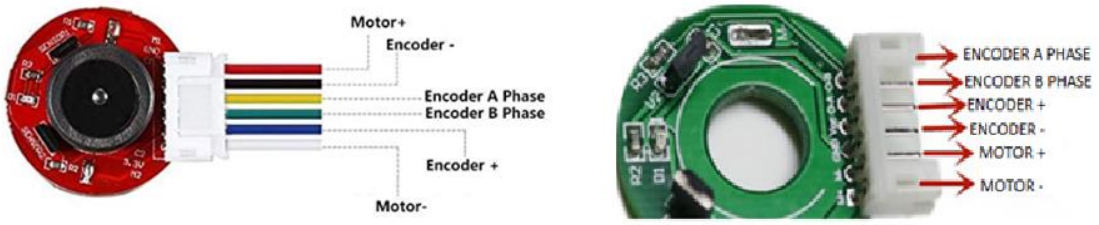
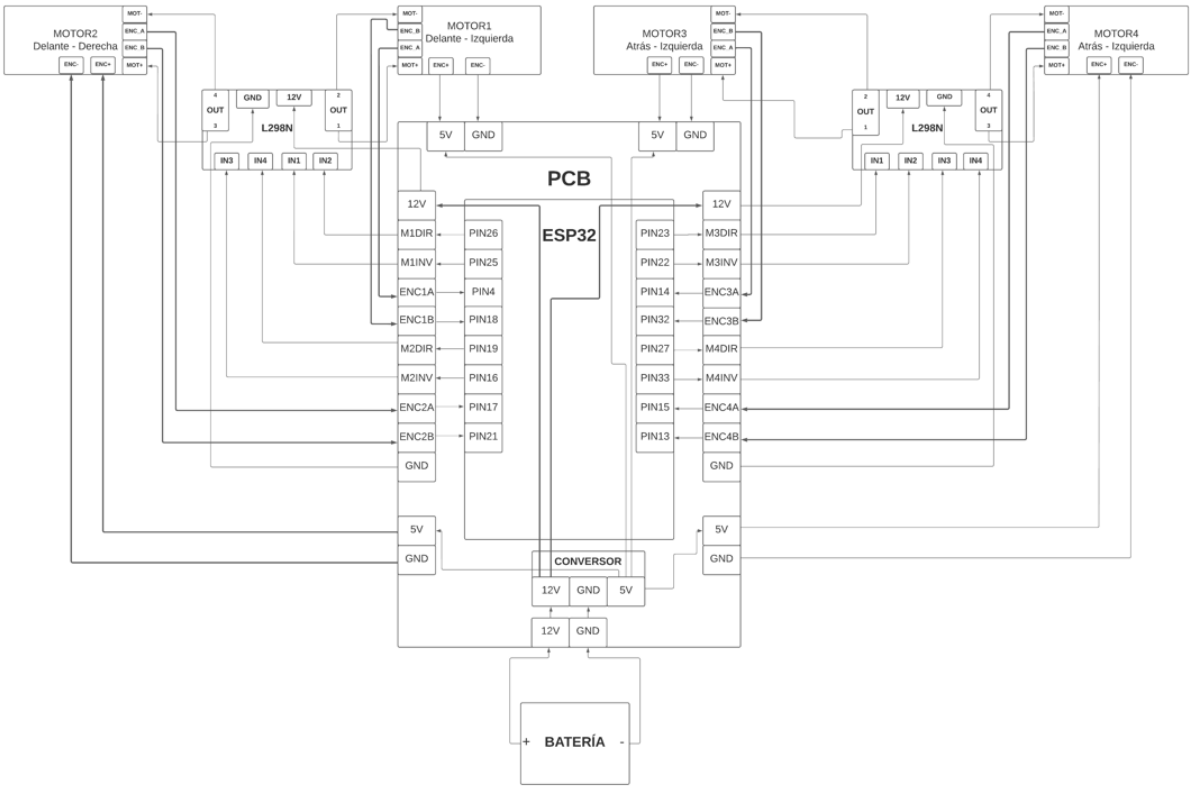


Figura 4.1.4: Comparativa de orden de los cables de los dos modelos de motores (izq: motores anteriores; der: motores definitivos)

Para esta versión ya definitiva se utilizó el esquema de conexiones del cableado del sistema móvil que se muestra a continuación:



ESP32: microcontrolador del sistema móvil
L296N: Driver en H para control de motores
Conversor: Conversor de tensión 12V a 5V
MOTOR1 a MOTOR4: Motores para mover las ruedas
PCB: Placa de Circuito impreso diseñada para el sistema
BATERÍA: Batería de litio de 12V

Figura 4.1.5: Esquema de conexiones del sistema móvil

Programación del control del sistema móvil

En este subapartado se describe cómo se ha realizado la programación en Arduino de las diferentes funcionalidades con las que cuenta el sistema móvil. Al igual que con el cableado del sistema móvil, el programa en Arduino no se hizo partiendo desde cero, sino que se contaba con una versión preliminar que había hecho otro compañero, colaborador anterior del grupo de I+D.

La primera versión del código en Arduino contenía una serie de funcionalidades, que fueron modificadas posteriormente y que se explican a continuación.

La primera versión del código ya leía la velocidad que llevaba cada motor, mediante el uso de sus encoders, contando la cantidad de vueltas por segundo que detectan, Sin embargo, al cambiar de motores hubo que cambiar la variable que definía los pulsos por vuelta que da un encoder, es decir, la resolución de los encoders.

También la primera versión podía recibir mensajes del tipo start/stop y mensajes para mover el sistema móvil hacia atrás o hacia delante. Sin embargo, el sistema móvil, en esta primera versión no podía girar.

La primera versión de la aplicación ya contaba con una función denominada *ActualizaMot*, que se explicará en detalle más adelante, que sirve para enviar la señal correspondiente a los motores en cada instante para que alcancen cierta velocidad, pasando por un controlador PI. El controlador PI en cuestión ha sido modificado y revisado múltiples veces para crear la versión final de esta función.

Además de esto, también se mantuvieron los nombres de las funciones ActualizaMot y loop2 (la función bucle que se encarga de recibir e interpretar los mensajes Bluetooth). Los nombres de las variables referentes a la velocidad y aceleración se mantuvieron, así como los nombres de las constantes que hacen referencia a números de pines de la ESP32. Sin embargo, el valor de todas ellas se tuvo que cambiar al cambiar de microcontrolador.

A continuación, se muestran las variables más importantes para entender el funcionamiento del código:

- *MX_encoder_A* (Siendo X un número entre 1 y 4): son 4 variables que definen el número del pin de la ESP32 que estará conectado al canal A de cada encoder. Son valores del tipo entero que definen el número del pin en cuestión.
- *MX_encoder_B* (Siendo X un número entre 1 y 4): son 4 variables que definen el número del pin de la ESP32 que estará conectado al canal B de cada encoder. Son valores del tipo entero que definen el número del pin en cuestión.
- *Cm_ref:* es un número de coma flotante (Float) que define el cambio de la señal de referencia por cada instante, es decir, al cambiar la señal de referencia de los motores, en cada instante en el que se pasa por la función *loop* (es decir, continuamente), el valor de esta variable se sumará o restará a las señales de velocidad de referencia (de los motores de las ruedas de la parte izquierda o derecha) hasta llegar a la referencia adecuada, es decir, que las variables *Ref_iz* y *Ref_der* alcancen el valor de *Ref_des*
- *Ref_iz*: es un número de coma flotante (Float) que define la señal de la velocidad de referencia de los motores situados en la parte izquierda del sistema móvil (Motores 2 y 4). Los motores de este lado deberán alcanzar esta velocidad, lo cual será registrado por los encoders.
- *Ref_der*; es un número de coma flotante (Float) que define la señal de la velocidad de referencia de los motores situados en la parte derecha del sistema móvil (Motores 1 y 3). Los motores de este lado deberán alcanzar esta velocidad, lo cual será registrado por los encoders.
- *Ref_des*: es un número de coma flotante (Float) que define la señal de la velocidad de referencia global de los motores. Al cambiar este número, las variables Ref_iz y Ref_der se deben ajustar para alcanzar el valor de esta variable, sumando o restando a las señales de velocidad de referencia (de los motores de las ruedas de la parte izquierda o derecha, respectivamente) cada vez que se llama a la función loop el valor de Cm_ref
- *Desv*: Es un número entero (Int) que define la dirección que va a tomar el sistema móvil. Si es positivo se moverá hacia delante y si es negativo se moverá hacia atrás. Su valor absoluto determina la dirección a tomar:
    - Si vale 1 el coche seguirá una trayectoria en línea recta
    - Si vale 2 el coche girará a la derecha
    - Si vale 3 el coche girará a la izquierda
    - Si vale 4 el coche hará un giro cerrado hacia la derecha
    - Si vale 5 el coche hará un giro cerrado hacia la izquierda

- *Ajuste_fino* es un número de coma flotante (Float) por el cual se multiplicará la velocidad de referencia de las ruedas del lado izquierdo del sistema móvil durante un periodo de tiempo de 0.75 segundos para que se muevan un poco más lento o un poco más rápido y así corregir su trayectoria en un sentido u otro
- *SesgoLento* es un número de coma flotante (Float) por el que multiplicará la velocidad de referencia de a las ruedas de la izquierda o derecha para que se muevan más lento y así poder girar en un sentido u otro. Es decir, al girar a la derecha, Ref_der será multiplicado por este número y el producto será la velocidad a alcanzar por las ruedas del lado derecho, para que su velocidad sea menor a las del lado contrario y así realizar el giro
- *SesgoRapido* es un número de coma flotante (Float) por el que multiplicará la velocidad de referencia de a las ruedas de la izquierda o derecha para que se muevan más rápido y así poder girar en un sentido u otro. Es decir, al girar a la derecha, *Ref_iz* será multiplicado por este número y el producto será la velocidad a alcanzar por las ruedas del lado izquierdo, para que su velocidad sea mayor a las del lado contrario y así realizar el giro
- *Stop* es un número entero (Int) que indica si los motores se deben moverse al recibir una orden o no. Si esta variable vale 1, los motores estarán apagados y si vale cero, los motores estarán encendidos y podrán actuar con normalidad.
- *Kp* es un número de coma flotante (Float) que representa la constante proporcional utilizada en el lazo de control implementado en el código.
- *Ki* es un número de coma flotante (Float) que representa la constante integral utilizada en el lazo de control implementado en el código.
- *Er* es un número de coma flotante (Float) que representa la diferencia entre la velocidad de referencia a alcanzar por cada motor y la velocidad medida por los encoders de cada motor.
- *FLAG_DISCONNECT* es un booleano (Bool) cuyo valor es igual a True cuando se detecta que el usuario se ha desconectado del Bluetooth del sistema móvil, o a False, cuando se vuelve a conectar. Si esta variable vale True durante más de 2 segundos, los motores se apagarán.
- *Tiempo_giro* es un número entero (Int) que indica el tiempo en milisegundos que el sistema móvil debe estar girando para realizar un desvío en su trayectoria.

El contenido del programa en Arduino fue modificado reiteradamente hasta llegar a la versión actual definitiva del programa incluido en este TFG. La función *ActualizaMot* es llamada cada vez que se desactiva la variable *Stop* , es decir, cuando el usuario enciende los motores mediante la aplicación móvil pulsando en el botón “ON”, como se explica más adelante. Esta función realiza el lazo de control para la señal de cada uno de los motores, recibiendo en cada instante la señal de la velocidad de referencia y la señal de la velocidad real medida que lleva el motor en ese momento, así como el número de los pines a los que está conectado el positivo y negativo de cada motor (para enviar la señal correspondiente en cada caso) y el error en el instante anterior.

Al pin que está conectado al positivo de cada motor se le llama “Canal directo” y al pin conectado al negativo de cada motor se le llama “Canal inverso”. De esta forma, se envía una señal al canal directo o inverso del motor cada vez que se desee que gire en un sentido u otro.

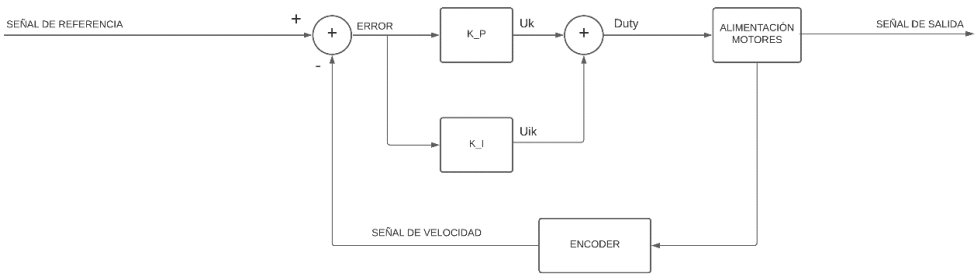


Figura 4.1.6: Lazo de control implementado en el proyecto

El lazo de control de esta función se muestra a continuación:

```
//Calculo de la accion utilizada
er=ref-vel;
Ik=T*er/1000.0+Ik_1;
Uik=Ik*ki;
Uk=kp*er ;
acc=Uk+Uik;

//Seguridad de la velocidad
    if (acc>v_max)
      acc=v_max;
    if (acc<v_min)
      acc=v_min;

//Calculo y seguridad del duty
duty=(uint32_t)(abs(acc)*CS2P)+DutyMin;
if (duty>DutyMax){
duty=DutyMax;
}else{
  if (duty<DutyMin){
    duty=DutyMin;}
}
```

Figura 4.1.7: Función con el lazo de control en Arduino

Como se puede apreciar, en esta función se calcula el error entre referencia obtenida y velocidad del sistema móvil, se calcula la parte integral (Uik) y proporcional (Uk) del controlador y se suman en la variable acc, que luego se convierte a la variable duty, la cual representa la señal que va a ser enviada a cada motor.

Si el valor de la señal es positivo, se enviará por el canal inverso la señal de 0V y por el directo la señal PWM, haciendo que, de esta forma, el motor gire hacia delante. Por otro lado, si el valor de la señal es negativo, significa que el sistema móvil debe moverse hacia atrás. En este caso, por el canal inverso del motor se envía la señal PWM de velocidad y por el canal directo se envía una señal de 0V.La función de callback *disconnectionCallback* se activa cuando el sistema móvil pierde la conexión bluetooth con la aplicación. Esta función callback activa un flag *FLAG_DISCONNECT* e inicia un contador. Si pasan más de 2 segundos y la conexión no se ha recuperado, los 4 motores se paran inmediatamente. Sin embargo, si esta función callback recibe un evento del tipo *ESP_SPP_SRV_OPEN_EVT*, significa que la conexión se ha recuperado. Si esto pasa en menos de 2 segundos, los motores no se detendrán.

Se realizó la programación de esta forma para que el sistema móvil no se detuviese en seco al cambiar la orientación de la pantalla del móvil o al cambiar de una pantalla a otra en la aplicación móvil, puesto que se detectó que se producían desconexiones y reconexiones del bluetooth cuando el usuario giraba el móvil y cambiaba la orientación.

Nada más empezar el programa, durante los primeros 2 segundos los motores quedan inactivos para dejar tiempo a que se inicialicen todas las variables. Este tiempo sirve para evitar errores con las variables que pueden ser modificadas, como, por ejemplo, no cambiar la velocidad de

referencia cuando todavía no se han iniciado las variables que hacen referencia a la velocidad actual del motor, lo cual podría producir que la cantidad de energía que se le ha de suministrar a los motores no sea la adecuada y el sistema móvil alcance velocidades superiores o inferiores a las deseadas. Durante este periodo no se activará ninguno de los cuatro motores.

Tras esto, el sistema móvil, al recibir un mensaje vía Bluetooth que le haga cambiar su velocidad, éste no cambiará su referencia de velocidad de manera instantánea, sino que esta variable cambiará progresivamente sumando o restando el valor de la variable *Cm_ref,* que haya establecido el usuario, hasta obtener una velocidad igual la velocidad de referencia deseada. La señal que se envía a cada motor debe proporcionar a éste la velocidad indicada en la señal de referencia, por medio de un control PI.

De esta forma, cuando se cambie de velocidad del sistema móvil, la velocidad de los motores variará de manera controlada desde la velocidad de partida hasta llegar a la velocidad final (nueva señal de referencia). Cada motor tiene un lazo de control distinto para llegar a la velocidad deseada, por lo que la señal de referencia de cada motor se gestiona individualmente.

A continuación, se muestra una gráfica obtenida gracias a la herramienta *Serial Plotter* de Arduino, que muestra la señal de referencia de velocidad que debe alcanzar el motor (en azul), la velocidad que lleva el motor en cada momento (en rojo) y el error entre estas 2 variables (en verde).

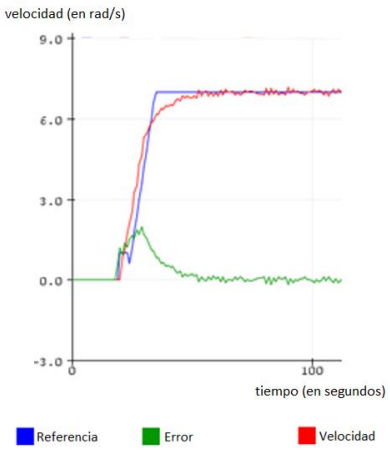


Figura 4.1.8: Gráfica de la señal de referencia y velocidad de un motor del sistema móvil

Como se puede apreciar, el valor de referencia no se cambia instantáneamente, sino que se aumenta de manera progresiva hasta llegar al valor indicado. Además, la velocidad del motor se adapta progresivamente al valor de la referencia por medio de un controlador PI.

Para que, al arrancar los motores, éstos alcancen la velocidad de manera progresiva, el valor de referencia no se cambia instantáneamente, sino que se aumenta poco a poco hasta llegar al valor indicado (sumando en cada instante el valor de *Cm_ref*), esto se puede apreciar en la figura. Se ha realizado de esta forma el cambio de velocidad para que no ocurra un aumento brusco en la velocidad de los motores, lo que podría afectar a la estabilidad del sistema móvil, que afecte al video en directo que retransmite la cámara sobre el mástil, debido a las vibraciones, y, por tanto, comprometer su estabilidad.

Al cambiar los 4 motores, la resolución de los encoder de éstos no era la misma que la de los encoders de los motores originales. La resolución de un encoder es la cantidad de pulsos que

emite por cada vuelta completa que da el eje del motor. Para calcular la velocidad de los motores se cuenta la cantidad de pulsos que emite el canal A o B de éste (A para cuando gira hacia delante y B para cuando gira hacia atrás) y al llegar a cierta cantidad se considera que ha dado una vuelta. Esta cantidad estaba definida como la resolución de cada encoder y era igual para los 4 motores. Al cambiar de motores hubo que volver a calcularla. Para ello, se contabilizaron la cantidad de pulsos que enviaba uno de los canales del encoder al mover uno de los motores. Tras ello se hizo que el motor diese una vuelta completa, y se obtuvo un resultado de 512 pulsos. Por tanto, la resolución del encoder es de 512 pulsos por cada vuelta. Este dato fue comprobado en los 4 nuevos motores y era el mismo.

Al conocer el instante en el que cada motor da una vuelta, solo se necesita contabilizar las vueltas que da por segundo y, de esta forma, se obtiene su velocidad en cada instante, medida en rad/s mediante la siguiente fórmula:

Velocidad_actual = NºPulsos*2*3.14 / (Tm*resolución)

, donde Tm es el periodo de muestreo en segundos, es decir, el tiempo que tarda en medir la velocidad que lleva el motor en cada instante.

Para que el sistema móvil pueda girar y tomar curvas se implementó la opción de girar, al recibir un mensaje por vía Bluetooth que indique al sistema móvil que debe girar a uno de los lados. En ese caso, la velocidad de los motores de ese lado disminuirá, multiplicando la referencia de velocidad por *sesgoLento,* y las de los motores del lado contrario aumentarán su velocidad, multiplicando la referencia de velocidad por *sesgoRapido*. De esta forma, el sistema móvil puede girar hacia un lado u otro aumentando o disminuyendo la velocidad de las ruedas de la izquierda o derecha.

Si el usuario desea arrancar el sistema móvil o cambiar su dirección de atrás hacia delante, la referencia de velocidad cambia de manera progresiva, como ya se ha explicado, para evitar movimientos bruscos. Sin embargo, a la hora de girar o volver a la trayectoria rectilínea, el cambio de referencia debe ser instantáneo para que el sistema móvil pueda tomar las curvas sin desviarse. Para programar esto, se hace un seguimiento del valor de la dirección que se le ha indicado al sistema móvil antes de tomar la nueva dirección, el valor de esta dirección anterior a la actual se guarda en una variable llamada "*AntDesv*", si el valor de esta variable es igual a la dirección actual, es decir, a la variable *Desv* o si una es negativa y la otra positiva, se realizará el cambio de referencia de manera progresiva. En el resto de los casos, se considera que el sistema móvil sigue moviéndose hacia el mismo sentido, pero realizando un giro o pasando de un giro a una trayectoria recta, por lo que se cambia el valor de referencia de velocidad de manera instantánea.

Un problema que se detectó en el sistema móvil estaba relacionado con el hecho de que los 4 motores deban cambiar su aceleración y velocidad a la vez, pero de manera independiente, por lo que, al cambiar el movimiento de atrás hacia delante, si el sistema móvil estaba realizando un giro, las velocidades de las 4 ruedas no son exactamente iguales, por lo que unas empezaban a girar marcha atrás antes que las otras, lo cual producía un giro del sistema móvil sobre su eje que no estaba controlado. Se ha podido arreglar controlando cuándo las 4 ruedas del sistema móvil están paradas. De esta forma, por ejemplo, si el sistema móvil está girando a la derecha y el usuario pulsa el botón de marcha atrás, el sistema móvil lo primero que hará es frenar sus ruedas de forma progresiva y sólo cuando todas las ruedas están paradas, realizará el movimiento marcha atrás.

Para tomar las curvas más cerradas o realizar un cambio de sentido en poco espacio, el giro descrito previamente no daba buenos resultados, por lo que se tuvo que implementar una opción para realizar un giro cerrado (giro de 180 grados en el mismo lugar). Cuando el sistema móvil recibe un mensaje, vía Bluetooth, indicando que debe realizar un giro cerrado en un sentido u otro, las ruedas de un lado aumentarán progresivamente la velocidad hasta alcanzar su valor original multiplicado por *sesgoRapido* y las del lado contrario disminuirán su velocidad progresivamente hasta llegar a 0. De esta forma, sólo giran las ruedas de un lado y se pueden tomar curvas muy cerradas y realizar cambios de sentido.

Al probar el sistema móvil, con los 4 motores que lo componen alcanzando la misma velocidad, se pudo notar que éste no seguía una trayectoria rectilínea, sino que sufría una desviación hacia la derecha, lo cual fue un problema que se trató de solucionar abarcando tanto el montaje del sistema móvil, como modificando el código de Arduino y el código de la aplicación móvil.

Para solucionar este problema y teniendo en cuenta que el sistema móvil se debe poder mover por diferentes tipos de suelo que provocarán resistencias diferentes a las ruedas y con diferentes grados de inclinación del terreno, se optó por utilizar una variable llamada "*ajuste_fino*", la cual es un número de coma flotante que se utiliza para que las ruedas del lado izquierdo vayan un poco más rápido o un poco más lento que las ruedas del otro lado durante un breve periodo de 0.75 segundos para que el sistema móvil pueda corregir su trayectoria cuando lo decida el usuario. De esta forma al pulsar uno de los botones con una flecha en diagonal, el sistema móvil cambiará su trayectoria un poco en un sentido u otro, de esta forma el usuario puede cambiar un poco la trayectoria del sistema móvil a la derecha si ve que se está desplazando hacia la izquierda o al revés.

La fórmula que se ha seguido en el código para implementar esta variable con el efecto deseado es la siguiente:

$$Ref\_izq = Ref\_izq \pm Ref\_izq * ajuste\_fino$$

El valor por defecto que se muestra en la aplicación es de 0.1, ya que este valor debe ser pequeño para que el sistema móvil no sufra giros muy bruscos. Sin embargo, el valor que toma esta variable *ajuste_fino* se puede cambiar desde la aplicación por el usuario. Sin embargo, este valor no puede ser mayor a 1, debido a que la velocidad de uno de los lados sería excesiva y podría comprometer la seguridad del sistema móvil.

Esta solución funcionaba de manera aceptable para corregir pequeñas desviaciones del sistema móvil. Sin embargo, si al girar hacia uno de los lados, el usuario orientaba de forma incorrecta el sistema móvil, corregir esa trayectoria resultaba algo difícil. Para solucionar este problema se optó por una idea algo diferente a la solución anterior. Se ha programado una funcionalidad en la que las ruedas de uno de los lados aumentan su velocidad mientras que las del otro lado la disminuyen, pero únicamente durante un breve periodo de tiempo configurable por el usuario, con un valor por defecto de 500 ms (aumentando o disminuyendo este periodo de tiempo el sistema móvil girará más o menos grados en el sentido indicado). La variable que indica el tiempo en milisegundos que debe girar el sistema móvil se llama "*tiempo_giro*". De esta forma, el sistema móvil, al recibir vía bluetooth el mensaje correspondiente, activará un contador de tiempo o temporizador (con un valor de *tiempo_giro*), durante el cual la velocidad de las ruedas de un lado tenderá a 0 progresivamente y la velocidad de las ruedas del otro lado aumentará de forma progresiva hasta alcanzar su valor original multiplicado por sesgoRapido.

Cuando el contador de tiempo alcance el valor de *tiempo_giro*, el valor de la velocidad de referencia de las ruedas volverá progresivamente a tomar el valor que tenían al inicio del proceso, haciendo que el sistema móvil continúe con su movimiento, pero en una nueva trayectoria.

Órdenes de control recibidas vía bluetooth desde la app móvil

Aprovechando que el microcontrolador ESP32 Feather utilizado tiene un módulo de comunicaciones Bluetooth para enviar y recibir mensajes mediante esta tecnología, se ha optado por usar mensajes bluetooth enviados desde la aplicación móvil a la ESP32 Feather para el control del sistema móvil. De esta forma se establece una comunicación simple y de alta velocidad entre el usuario y el sistema móvil.

Mediante estos mensajes desde la aplicación móvil el usuario puede indicar qué trayectoria y velocidad debe seguir en cada momento el sistema móvil, además de configurar el valor de algunas de las variables para un mejor control del sistema móvil.

A continuación, se detalla el formato de los diferentes tipos de mensajes que puede recibir el sistema móvil, vía Bluetooth, para cambiar su funcionamiento y variables internas anteriormente descritas.

Dichos mensajes tienen 2 partes: el encabezado y el valor. El encabezado sirve para diferenciar entre los tipos de mensajes que puede recibir. Dependiendo de cada tipo, se modificará la velocidad o la aceleración del sistema móvil, además de poder encender y apagar los motores.

A continuación, se explican los 3 tipos de mensajes que puede recibir el sistema móvil:

- *St*. Su valor puede ser 1 para encender (start) los motores o 0 para apagarlos. Ejemplos: “St0” o “St1”.
- *Ve*, para indicar la velocidad y dirección que debe tomar el sistema móvil. Se envían en el formato “Ve*vel:dir*”, donde
    - *vel* es un valor, de tipo float, que define los rpm a los que debe llegar el motor.
    - *dir* es un valor, de tipo entero, que puede ser 1, 2, 3 o -1, para indicar si el sistema móvil debe ir hacia delante, derecha, izquierda o atrás, respectivamente.

  Ejemplo: “Ve5:1”, indica que la velocidad es de 5rpm y se mueve en sentido hacia adelante.
- *Ac*, para indicar la Aceleración. Es un número, de tipo float, que indica cómo de rápido cambiará la señal de referencia. Básicamente se va sumando este valor cada cierto tiempo a la referencia de velocidad anterior hasta llegar a la deseada. Ejemplo: “Ac0.9” implica un aumento de la velocidad de referencia de 0.9 rpm cada segundo.
- *Se*, para indicar el valor de ajuste_fino. Es un número, de tipo float, poque se utilizará para aumentar o disminuir la referencia de velocidad de los motores de la izquierda para compensar el desvío del sistema móvil cuando se mueva hacia delante.
  Ejemplo: “Se0.01”, indica que el nuevo valor de *ajuste_fino* es 0.01

- *Sr*, para indicar el valor de sesgoRápido. Es un número, de tipo float, por el que se multiplicará el valor de referencia de velocidad de los motores correspondientes que se moverán más rápido al girar, para que el giro hacia un lado u otro siga la trayectoria deseada. Este valor ya no es posible modificarlo desde la app, ya que se usó para determinar el giro deseado y se estableció ese valor por defecto.
  Ejemplo: "Sr1.3", indica que el nuevo valor de *sesgoRápido* es 1.3
- *Sl*, para indicar el valor de *sesgoLento*. Es un número, de tipo float, por el que se multiplicará el valor de referencia de velocidad de los motores correspondientes que se moverán más lento al girar, para que el giro hacia un lado u otro siga la trayectoria deseada. Este valor ya no es posible modificarlo desde la app, ya que se usó para determinar el giro deseado y se estableció ese valor por defecto.
  Ejemplo: "Sl0.7", indica que el nuevo valor de *sesgoLento* es 0.7
- *Gi*, para indicar el tiempo que va a estar girando el sistema móvil para corregir su trayectoria y la dirección que va a tomar. Se mandan en formato "*Gidir:time*", donde:
  - *dir* es un valor, de tipo entero, que indica la dirección a la que se debe desviar el sistema móvil, puede ser 2 o 3, para indicar si el sistema móvil debe corregir hacia la derecha o izquierda, respectivamente.
  - *time* es un valor, de tipo float, que indica el tiempo que el sistema móvil va a estar girando.

  Ejemplo: "Gi2:500" indica que el sistema móvil se va a girar hacia la derecha durante 500 milisegundos

Comparativa entre la versión original y la versión actual

A continuación, se muestra una tabla resumen que compara la versión original heredada del por el grupo de I+D y la versión final del sistema móvil, tanto en lo referente al montaje como en todo lo referente a la programación del microcontrolador.

	Versión original	**Versión final**
Placa PCB	No, circuito montado en protoboard	Si
Cables Crimpados	No, uso de cables para protoboard	Si, cables crimpados a mano
Motores	Motores antiguos que venían incluidos con el chasis de tanque. 150 rpm de velocidad máxima	Motores nuevos de 280 rpm de velocidad máxima. El orden de los cables de los nuevos motores es distinto al de los antiguos
Baterías	Una batería de 12 voltios para los motores y una batería portátil de 5 voltios para el microcontrolador	Una sola batería de 12V
Microcontrolador	ESP32 WROOM	ESP32 Feather
Recepción de mensajes Bluetooth	Solo mensajes del tipo *"St"*, "*Vel"* y *"Ac"*	Puede recibir todos los tipos de mensajes descritos en la memoria
Velocidad y aceleración configurables	Si	Si

Lectura de velocidad por los encoders	Si	Si
Cambio progresivo de referencia	Si	Si
Controlador PI integrado	Si	Si, pero Ki y Kp han sido probadas y modificadas para mejor rendimiento
Movimiento	Hacia detrás y hacia delante	También realiza giros hacia izquierda y derecha.
Giro cerrado	No	Si
Corrección de trayectoria	No	Si
Ajuste Fino	No	Si
Detección de desconexiones Bluetooth	No	Si
Cámara Insta360Pro incorporada	No	Si
Aplicación móvil	No	Si

4.2 Aplicación móvil para dispositivos móviles con SO Android y uso de la cámara

La aplicación móvil para el control del sistema móvil y visualización del vídeo en directo ha sido desarrollada desde cero completamente. No se ha heredado ningún tipo de código para el desarrollo de esta aplicación en Android Studio.

La aplicación, desarrollada para ser ejecutada en un dispositivo móvil basado en el sistema Operativo Android, tiene 2 propósitos fundamentales: por un lado, controlar a distancia el movimiento del sistema móvil, y, por otro, mostrar el vídeo que capture la cámara de vídeo 360, tanto de forma apaisada como de forma estereoscópica. Para ello hace uso de tecnologías Bluetooth y WiFi. A continuación, se muestra un esquema general de las comunicaciones del sistema.

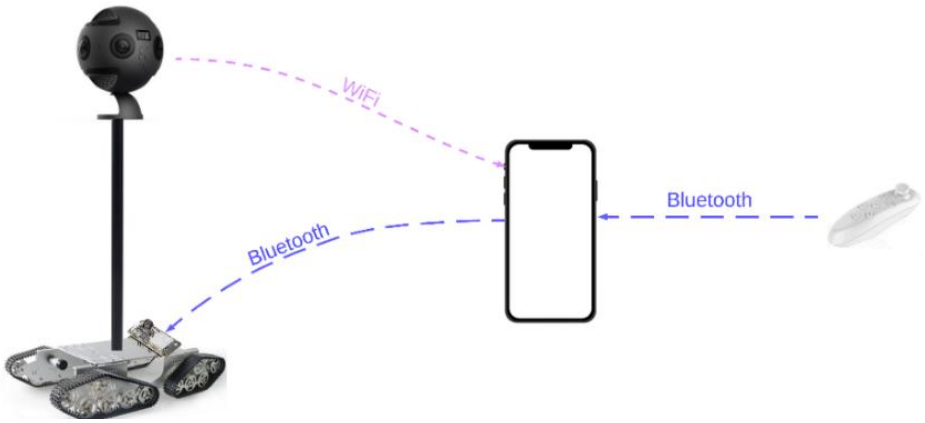


Figura 4.2.1: Esquema de comunicaciones

Para controlar el sistema móvil se hace uso de señales Bluetooth, por lo que, al iniciar la aplicación, se pedirán permisos de acceso a Bluetooth al usuario. También, si, al iniciar la aplicación, el Bluetooth está desactivado, un cuadro de diálogo pedirá al usuario poder activar el Bluetooth del dispositivo móvil automáticamente.

Figura 4.2.2: Cuadro de diálogo Bluetooth

Para poder visualizar la emisión de vídeo en directo de la cámara de vídeo 360, el dispositivo móvil debe estar conectado a la misma red WiFi que la cámara de vídeo 360, por lo que, antes de iniciar la aplicación, el usuario debe ir a ajustes de WiFi y conectarse a la red correspondiente. La cámara 360 utilizada cuenta con su propia red WiFi llamada Insta360-Pro-YCGQM8-OSC, si la cámara no está conectada a ningún punto de acceso WiFi, el usuario deberá conectarse esta red. En caso contrario, el usuario se deberá conectar a la red WiFi del punto de acceso al que se haya conectado la cámara.

A continuación, se muestra el diseño de la aplicación móvil desarrollada para este TFG.

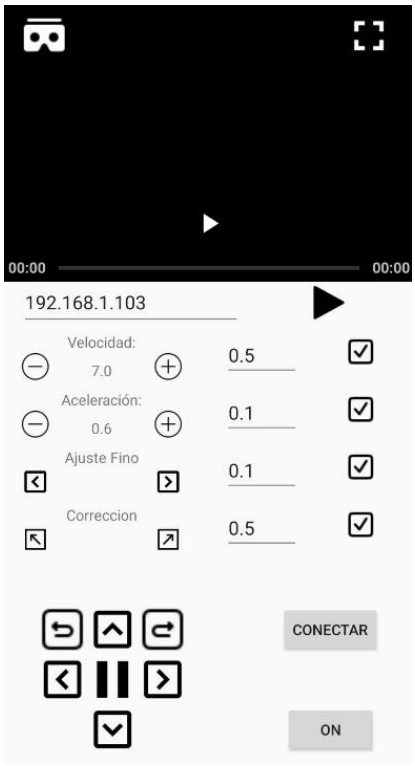


Figura 4.2.3: Imagen de Screenshot de la aplicación

En la actividad principal que se muestra al usuario (la pantalla principal para el control del sistema móvil) existe un visor que reproduce el vídeo en 360 que está emitiendo en directo la cámara. Este video puede ser visualizado en 360 grados moviendo el dispositivo móvil o desplazando con el dedo el punto de vista. En este reproductor de vídeo, en la parte superior derecha, hay un icono que se puede pulsar para mostrar el vídeo en directo en pantalla completa. Al hacerlo, desaparecen los controles (sobre todo las flechas) del sistema móvil y, por tanto, en ese caso será necesario controlar el sistema móvil mediante el uso del mando a distancia bluetooth descrito en el capítulo 3.

Debajo del visor de vídeo, se encuentra un cuadro de texto en el que el usuario debe introducir la dirección IP de la cámara. A continuación, para visualizar el vídeo se deberá pulsar el botón con el icono “PLAY” que está al lado.

Debajo del cuadro de texto se encuentran unos selectores que permiten al usuario regular tanto la aceleración como la velocidad de referencia del sistema móvil en cada momento.

En la parte inferior de la pantalla se encuentran una serie de botones que permiten al usuario indicar al sistema móvil la dirección que debe tomar. Tras pulsarse uno de éstos, su apariencia cambiará a un botón de color negro y el dispositivo móvil emitirá una vibración.

Al lado de estos botones direccionales se encuentran los botones para conectarse o desconectarse del sistema móvil mediante Bluetooth, para encender o apagar los motores del sistema móvil y para que el sistema móvil se frene o continúe su trayecto de forma progresiva.

Esta pantalla principal también se puede visualizar en vista Landscape (vista con el dispositivo móvil en horizontal) de la siguiente forma:

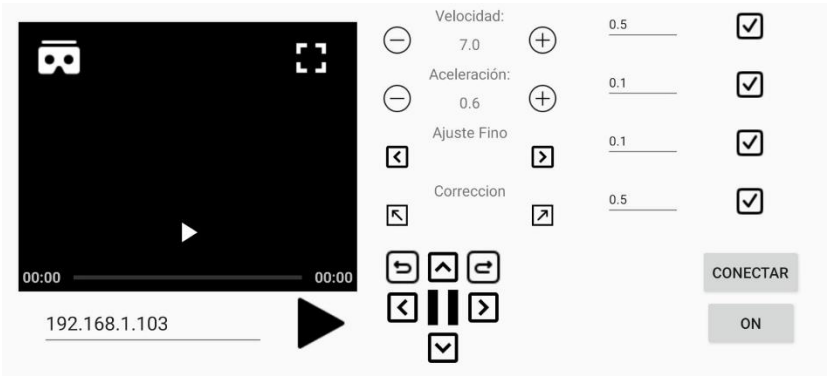


Figura 4.2.4: Imagen de Screenshot de la aplicación en vista Landscape

Control del sistema móvil mediante Bluetooth

Para poder enviar los mensajes de control al microcontrolador del sistema móvil, la aplicación utiliza un Socket Bluetooth, estableciendo una conexión con el mismo y enviando los diferentes mensajes que transmita el usuario. Las flechas direccionales sirven para indicar la dirección a tomar por el sistema móvil, por lo tanto envían un mensaje del tipo "*VeX:Y"* donde la X es la variable que hace referencia a la velocidad en m/s y la Y la dirección que debe tomar el sistema móvil.

Antes de poder enviar cualquiera de los mensajes, el dispositivo móvil (smartphone o tablet) tiene que registrar al sistema móvil como un dispositivo Bluetooth. Para ello, el usuario debe dirigirse a los Ajustes de Android del dispositivo y entrar en la pestaña de Bluetooth. Tras ello y con el Bluetooth activado, el usuario debe buscar en "Dispositivos Disponibles" un dispositivo con nombre "VideoCoche" y conectarse a él.

Una vez que el usuario se haya conectado mediante bluetooth al sistema móvil, éste podrá pulsar el botón de conexión bluetooth de la aplicación desarrollada, que tiene el texto "*Connect*", para establecer una conexión con el microcontrolador del sistema móvil (al pulsar este botón, el texto de este botón cambiará a *"Disconnect"*). Tras esto, el usuario pulsará el botón "ON" para encender o activar los motores. El texto del botón pasará a ser OFF. En el momento que quiera apagarlos, solo deberá pulsar el mismo botón. Esto se gestiona enviando mensajes del tipo "*StX*", siendo X=1 para apagar y X=0 para encender los motores. Tras volver a pulsar el botón de conexión bluetooth que ahora tiene el texto *"Disconnect"*, se finalizará la conexión Bluetooth con el microcontrolador del sistema móvil y el texto en este botón volverá a ser "*Connect*". Además, el resto de los botones tomarán la apariencia que tenían al iniciar la aplicación.

El usuario puede cambiar la velocidad que tomará el sistema móvil en tiempo real utilizando uno de los selectores de la parte superior de la pantalla, esto hará que cambie el valor de la variable Velocidad de referencia que se enviará en los mensajes Bluetooth, si ninguna de las flechas es pulsada. Sólo cambiará el valor de esta variable, pero si se cambia el valor mientras una de las flechas es pulsada, es decir, cuando el sistema móvil se encuentre en movimiento, se enviará un nuevo mensaje con la nueva velocidad a alcanzar y la dirección actual que está siguiendo el sistema móvil.

La aceleración del sistema móvil también puede cambiar su valor mediante otro selector, el cual enviará un mensaje del tipo "*Ac*" cuando el usuario deje de pulsarlo. Al hacer esto se modificará la variable *Cm_Ref* haciendo que los cambios de velocidad sean más o menos rápidos. El valor

por defecto de aceleración es de 0.6 rad/s. El usuario puede cambiar este valor utilizando el selector de aceleración, el cual aumenta o disminuye esta variable en saltos de 0,1 rad/s por defecto, aunque estos saltos también son modificables por el usuario mediante los editores de texto situados al lado de los selectores. El usuario debe establecer el valor deseado y luego pulsar el checkbox situado al lado del editor de texto para que el valor se actualice.

El sistema móvil no debe parar su movimiento de manera brusca, por lo tanto, al pulsar el botón de STOP (botón que se encuentra en el medio de los botones direccionales con un icono "Pause"), se enviará un mensaje del tipo Ve0:0, haciendo que el sistema móvil reduzca de forma progresiva la velocidad hasta un valor de referencia de 0m/s, según la aceleración que haya configurado el usuario. Al volver a pulsar en este botón (que ahora tendrá un icono "Play"), se retoma progresivamente la dirección anterior que llevaba el sistema móvil.

Al desconectar el usuario el dispositivo móvil del sistema móvil, en un primer momento, se mandaba una señal St0 mediante Bluetooth para apagarlo. Sin embargo, debido a la latencia que sufre el mensaje hasta llegar a la ESP32 del sistema móvil, el mensaje no se enviaba correctamente. Por ello, para solucionar este problema se optó por utilizar el callback de desconexión en el código de Arduino.

Para que el dispositivo móvil se actualice cuando se pierda la conexión de forma inesperada con el microcontrolador del sistema móvil, al estar muy lejos o al finalizar la carga de sus baterías, por ejemplo, se ha implementado una clase Java del tipo "*BroadcastReceiver*" llamada *"BLReceiver"*. Un *BroadcastReceiver* en Java escucha eventos o mensajes del sistema y realiza acciones en respuesta. En este caso, si *BLReceiver* recibe un mensaje del sistema indicando que el adaptador Bluetooth que gestionaba la conexión con el microcontrolador del sistema móvil ha sufrido una desconexión, cambiará el valor de la variable "*connection*" que indica si la conexión está establecida a *False*. De esta forma, si la variable "*connection*" es igual a *False,* el texto del botón de conexión Bluetooth volverá a ser "*Connect*". Al igual que todos los botones tomarán la apariencia que tenían al iniciar la aplicación.

Sin embargo, el problema de la latencia anteriormente comentado seguía afectando y provocaba que se sobrescribiesen mensajes y algunas órdenes no llegaran al sistema móvil si varios botones eran pulsados en un intervalo muy corto de tiempo. Para ello, en varios controladores de la aplicación, como los botones direccionales, el botón de ON/Off o los botones del mando a de control remoto, se implementó un contador de 1.5 segundos, tiempo en el cual no se puede volver a pulsar ninguno de los botones anteriormente mencionados. Esto es tiempo suficiente para que la ESP32 reciba cualquier mensaje y ejecute la orden correspondiente.

En ese tiempo muerto ninguno de dichos botones podrá ser pulsado para enviar órdenes. De esta forma los botones no cambiarán su apariencia ni el dispositivo vibrará para que el usuario sepa que el mensaje no ha sido enviado y que debe volver a pulsar el botón. Al ser un tiempo de espera tan corto no resulta molesto para controlar el sistema móvil y evita que la ESP32 se sobrecargue de mensajes y éstos se sobrescriban.

Para mejorara el giro del motor, se implementó en la aplicación móvil una forma de configurar el sesgo que se le aplica a los motores de cada lado del sistema móvil al girar, es decir, poder configurar por qué número se multiplica la señal de referencia de los motores 2 y 4 (izquierda) o 1 y 3 (derecha), para que el sistema móvil gire con el ángulo deseado.

Estos selectores permiten configurar 2 variables: sesgoLento y sesgoRapido. SesgoLento hace referencia al número por el que se multiplica la velocidad de referencia de los motores que

deben ir más lentos al hacer el giro, sesgoRapido hace referencia al número por el que se multiplica la velocidad de referencia de los otros dos motores. Gracias al uso de estos selectores se pudo encontrar un ángulo de giro adecuado para el sistema móvil con los valores, por defecto, de 1.125 para el sesgo rápido y 0.5 para el sesgo lento.

Para conseguir que el sistema móvil siga una trayectoria recta cuando lo indique el usuario y que no se desvíe hacia uno de los lados debido a la fragilidad del sistema, se implementaron otros 2 nuevos selectores. El primero de ellos, el selector "Corrección" sirve para realizar cambio de dirección de aproximadamente 10 grados en la trayectoria del sistema móvil. De esta forma podremos dirigirlo aproximadamente a la trayectoria deseada si éste se desvía demasiado o si, al realizar un giro, ha tomado una dirección no deseada por el usuario. El segundo de estos selectores, el selector "Ajuste fino" sirve para realizar pequeños ajustes de trayectoria y alinear de forma más precisa el sistema móvil, utilizando la variable *ajuste_fino*. De esta forma, el usuario puede corregir al instante las pequeñas desviaciones momentáneas que sufre el sistema móvil durante su trayectoria y así seguir una línea recta.

Para implementar el selector Corrección se utilizó la siguiente lógica, al pulsar en cualquiera de sus 2 botones (derecha o izquierda), se manda un mensaje vía Bluetooth del tipo "*Gidir:time*" indicando la dirección en la que se quiere realizar el desvío y el valor de *tiempo_giro*, un número entero que representa el tiempo en milisegundos en el que el sistema móvil va a estar girando. Durante ese breve periodo de tiempo, las ruedas de un lado acelerarán hasta tomar una velocidad igual a la variable *sesgoRapido* veces mayor a la que llevaban y las del otro lado disminuirán poco a poco su velocidad tendiendo a 0. Como este tiempo ha de ser corto, las ruedas de un lado no llegarán a frenar del todo (a menos que el usuario configure un tiempo más alto porque así lo desea) y las del otro acelerarán, provocando que el sistema móvil tome una nueva dirección apuntando más a la derecha o izquierda, según haya elegido el usuario.

El valor de *tiempo_giro,* por defecto, es de 500 milisegundos, pero el usuario puede modificarlo indicando su valor en el cuadro de texto al lado del selector y pulsando en el icono de "Aceptar". Para modificar este valor, el usuario deberá introducir, en segundos, el tiempo que deba estar girando el sistema móvil. Esta variable puede tomar valores a partir de 0.01 segundos.

El selector de Ajuste fino funciona de manera algo distinta al anterior. Para realizar un pequeño ajuste de la trayectoria del sistema móvil en el instante en el que se desvía en un sentido o en otro, el sistema móvil aumenta o disminuye un poco el valor de referencia de velocidad que deben tomar las ruedas del lado izquierdo para girar momentáneamente y así cambiar de dirección unos pocos grados y compensar las desviaciones que sufre el sistema móvil en su dirección cuando está en funcionamiento, esto se hace utilizando la variable *ajuste_fino*.

Al tocar en cualquiera de los 2 botones del selector (derecha o izquierda). Se envía un mensaje vía bluetooth del tipo "*Se*" indicando el valor que debe tomar la variable *ajuste_fino*, para ajustar la trayectoria en un sentido u otro. En cuanto el sistema móvil recibe este mensaje, aumenta o disminuye la referencia del lado izquierdo de manera instantánea y tras 0.75 segundos, vuelve a tomar el valor original. El valor por defecto de *ajuste_fino* es 0.1, pero puede ser modificado por el usuario indicando su valor en el cuadro de texto al lado del selector y pulsando en el icono de "Aceptar".

Control mediante mando a distancia

En el sistema desarrollado en este TFG, para poder controlar el movimiento del sistema móvil mediante un mando a distancia cuando así se requiera, se utiliza un mando de unas gafas de realidad virtual que tenía el grupo de I+D, el cual funciona vía Bluetooth. Tras varios intentos infructuosos de conectar el mando directamente al sistema móvil, se decidió conectar el mando al dispositivo móvil como un dispositivo de entrada y leer los mensajes del tipo *KeyEvents* que recibía el dispositivo móvil de éste.

De esta forma, el dispositivo móvil del usuario sirve como un intermediario entre el mando a distancia y el sistema móvil (ver la figura 4.2.1), pudiendo controlarlo sin necesidad de tocar la pantalla del dispositivo.

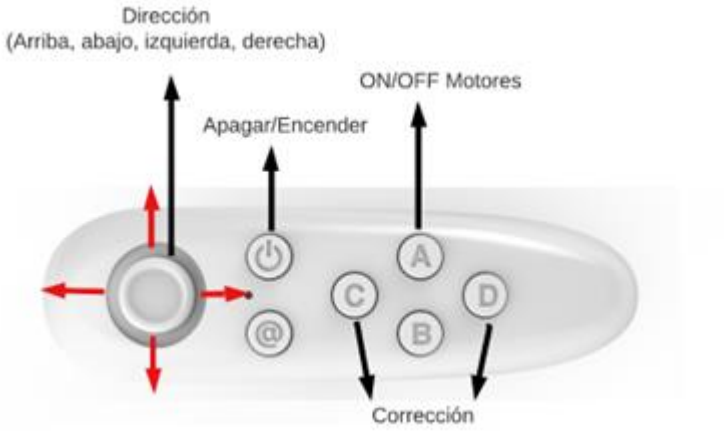


Figura 4.2.5: Controles del mando a distancia Bluetooth

De esta forma, debido a los escasos controles del mando, el usuario puede cambiar la dirección del sistema móvil, pararlo, apagar los motores y realizar correcciones a su trayectoria. Sin embargo, para configurar los valores de las variables tales como la velocidad y aceleración del sistema móvil o las variables *tiempo_giro* y *ajuste_fino* se deberá utilizar la aplicación móvil directamente.

Visualización del video 360

Lo primero que debe hacer el usuario para poder visualizar la emisión de vídeo en directo es obtener la aplicación de Insta 360 Pro [Ref 23] e instalarla en el smartphone o la tablet. Tras ello, se debe conectar a la red WiFi que emite la cámara o a la que esté conectada ésta en su defecto. Tras ello y dentro de la aplicación móvil de la cámara, el usuario puede comenzar la transmisión en directo, haciendo click en la opción “Record Livestream” si desea grabar el vídeo.

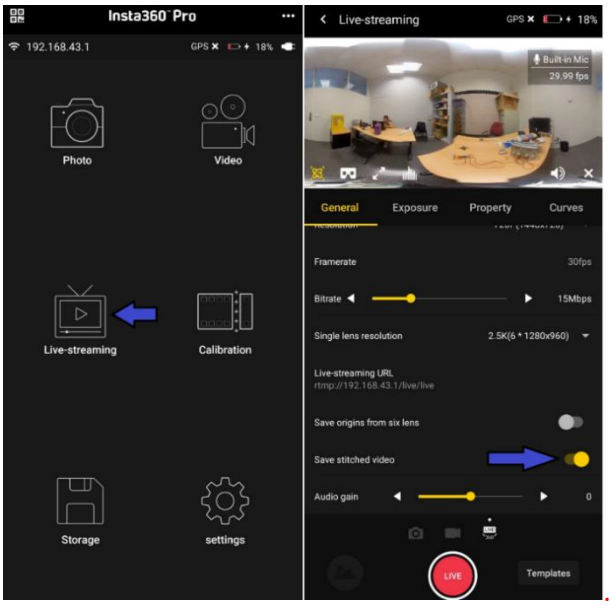


Figura 4.2.6: Grabación de video mediante la app de Insta360

Dentro de la aplicación desarrollada para este TFG, debajo del reproductor de vídeo se encuentra un cuadro de texto en el que el usuario debe introducir la dirección IP de la cámara y pulsar en el botón con el icono “Play” que se encuentra al lado. Tras ello, la aplicación móvil comenzará a recibir el flujo RTMP enviado a través de la cámara, el cual contiene la emisión del video en directo.

Para mostrar el vídeo de la cámara 360 se probó varias alternativas como el sdk de Google VR, la librería de PanormaView o el NDK de Google Cardboard. Sin embargo, la solución más práctica y que funcionaba de mejor manera con el protocolo de intercambio de datos RTMP fue utilizar la librería ExoPlayer [Ref 6].

Esta librería incluye un reproductor de vídeo que se puede conectar a servidores HLS, RTMP y RTSP de manera sencilla. También incluye una vista ‘*spherical_gl_surface_view*’ con la que se pueden visualizar vídeos y fotos en 360 grados, pudiendo ver todos los 360 grados de la imagen arrastrando con el dedo o mediante el uso de los sensores de aceleración y giroscopio incorporados en el dispositivo móvil del usuario.

Para utilizar el giroscopio y acelerómetro del dispositivo móvil para visualizar la emisión de vídeo en directo se debe preparar el reproductor de vídeo Exoplayer justo antes de lanzar la actividad, es decir, justo antes de que el usuario pueda interactuar con la pantalla de la aplicación. Para ello se debe llamar al método *onResume()* del reproductor de vídeo en la función *onResume()* de la propia actividad de la pantalla principal.

Para entender este funcionamiento del reproductor de vídeo se debe establecer primero qué es una actividad de Android y su ciclo de vida.

En Android, una actividad es un componente fundamental de una aplicación que representa una pantalla con una interfaz de usuario y una interacción con el usuario. Puede considerarse como una ventana en la cual se presentan los elementos visuales y se gestiona la lógica de interacción. Las actividades son responsables de recibir las entradas del usuario, mostrar información y realizar acciones específicas.

El ciclo de vida de una actividad se refiere al conjunto de estados por los que pasa una actividad, desde su creación hasta su destrucción. Este ciclo de vida es gestionado por el sistema operativo Android y permite a la actividad responder a eventos y realizar tareas en momentos clave. Los principales estados del ciclo de vida de una actividad incluyen:

- onCreate(): Es el primer método llamado cuando se crea una actividad. Aquí se realiza la primera inicialización, como la creación de la interfaz de usuario y la carga de datos.
- onStart(): Se llama cuando la actividad se hace visible para el usuario. En este punto, la actividad está en primer plano, pero aún no es interactiva.
- onResume(): Se llama cuando la actividad se vuelve interactiva y se inicia la interacción con el usuario. Aquí se realizan tareas como iniciar animaciones, reproducir música o iniciar servicios (en este caso, el reproductor de vídeo).
- onPause(): Se llama cuando la actividad pierde el foco, pero aún es visible para el usuario.
- onStop(): Se llama cuando la actividad ya no es visible para el usuario.
- onDestroy(): Se llama cuando la actividad es destruida y se elimina de la memoria. En este punto, se deben liberar todos los recursos utilizados por la actividad.

A continuación, se muestra el esquema del ciclo de vida de una actividad, desde que es creada, hasta que la misma se destruye al cerrar la aplicación:

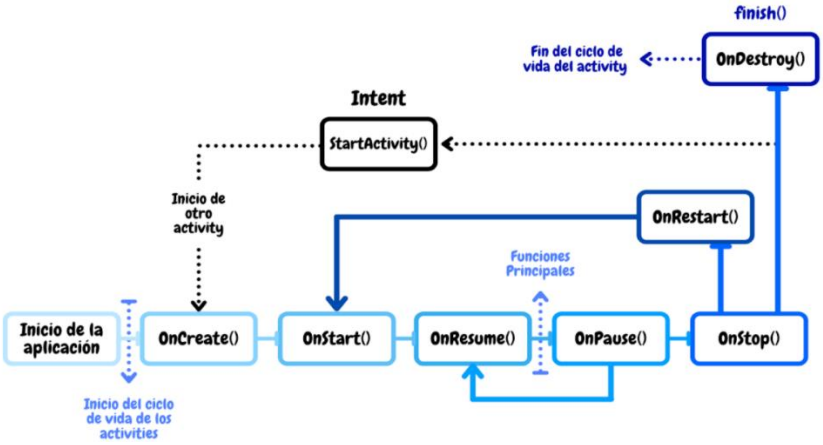


Figura 4.2.7: Ciclo de vida de una actividad [Ref 18]

De esta forma, al inicializar el reproductor de vídeo en la unción *onResume()*, cargamos todas sus configuraciones y la emisión de vídeo en directo de la cámara, cada vez que se muestra en primer plano. De esta forma, aunque se salga de la aplicación o actividad no se retrasará la emisión de vídeo pudiéndolo ver sin mayor latencia.

Un problema de la librería Exoplayer es que, a pesar de poder visualizar vídeos en 360 grados y soportar flujo RTMP, hay varias funciones que no están implementadas de base en la librería por defecto, las cuales se han tenido que montar de manera manual para ser integradas en la aplicación.

La librería de Exoplayer, por defecto, no soporta la funcionalidad de pantalla completa, es decir, no se puede visualizar el streaming de vídeo en pantalla completa en la misma actividad en la que gestionamos el movimiento del sistema móvil. Por lo tanto, se ha creado una nueva actividad de Android llamada "*MainActivity2*", que sólo se puede visualizar en vista *Landscape* y muestra un reproductor Exoplayer que ocupe toda la pantalla.

Exoplayer tampoco provee ningún plugin para visualizar el contenido del reproductor en un visor estereoscópico para poder visualizarlo mediante unas gafas de realidad virtual o un dispositivo HMD.

Generalmente, un visor estereoscópico es un dispositivo que permite experimentar imágenes en tres dimensiones (3D) de manera inmersiva. Consiste en un par de lentes separados que presentan una imagen ligeramente desplazada a cada ojo, lo que crea una ilusión de profundidad y percepción tridimensional. Al observar una imagen o video a través del visor, el cerebro fusiona las dos imágenes para crear una sensación de profundidad y realismo.

Exoplayer no provee esta solución, por lo tanto, se probaron varias alternativas. Una de ellas colocar un reproductor en la mitad de la pantalla y que un objeto TextureView copie el contenido que éste muestra, pero el contenido no se actualizaba a tiempo y era demasiada carga computacional para el dispositivo móvil. También se probó a hacer uso de librerías en Javascrip para montar una página web con un reproductor 360 y mostrarlo en la aplicación móvil, pero los reproductores de 360 actualmente en Javascript no soportan flujo de RTMP. Se probó a codificar el flujo RTMP y pasarlo a HLS para utilizar los reproductores Javascript, pero eso generaba un retraso de alrededor de 20 segundos en el reproductor, lo cual es inadmisible.

Como último intento y de implementación del visor estereoscópico en la actividad “MainActivity2” se añadieron 2 reproductores ExoPlayer dividiendo la pantalla, mostrando el mismo streaming en cada reproductor. De esta forma, cuando el usuario coloca el móvil en las gafas de realidad virtual y se las coloca, hará el efecto de un visor estereoscópico. Sin embargo al ser dos reproductores independientes que gestionan de manera separada el fujo RTMP, la vista de ambos ojos no está coordinada, lo que muchas veces provoca que el reproductor de un lado esté adelantado unos pocos instantes con respecto al otro, lo cual dificulta la experiencia de visualización en 360, sobre todo con el sistema móvil en movimiento.

Por lo tanto, no se ha llegado a desarrollar un visor estereoscópico satisfactorio que pueda gestionar el flujo de video en directo RTMP. Sin embargo, el visor estereoscópico desarrollado se encuentra disponible en la aplicación y se puede acceder a él mediante el un icono de unas gafas de realidad virtual situado en el reproductor de vídeo.

Sin embargo, para ello, muchas variables de “HomeActivity” deben ser enviadas a “MainActivity”, entre ellas la variable que define si se va a reproducir el vídeo en un visor estereoscópico o sólo en pantalla completa, además de todas las variables necesarias para poder seguir controlando el sistema móvil de la misma forma utilizando el mando a distancia Bluetooth y poder seguir visualizando el streaming sin introducir la dirección IP. Estas variables, como la velocidad, la aceleración o la dirección IP de la cámara, son enviadas a la nueva actividad mediante los "extras" de un Intent, que es un objeto utilizado para iniciar una nueva actividad.

El proceso utilizado para pasar variables entre las 2 actividades se puede resumir en los siguientes pasos:

En la actividad origen “HomeActivity” (la pantalla principal donde se controla el sistema móvil), se crea un objeto Intent para iniciar la actividad destino “MainActivity2”.

Se utiliza el método putExtra() del Intent para agregar la variable a pasar. A esta función se le pasan 2 parámetros, el primero es una clave única que se utiliza para identificar la variable, y el segundo es el valor dicha variable.

Se inicia la actividad destino utilizando el método startActivity() y se pasa el Intent como argumento.

En la actividad destino “MainActivity2”, se obtiene el Intent utilizando el método getIntent(). Luego, se utiliza el método getExtra() del Intent para recuperar la variable pasada, utilizando la misma clave utilizada en la actividad origen.

De esta forma, al iniciar la nueva actividad, se crea una nueva sesión Bluetooth para controlar el sistema móvil, se inicializa el reproductor de vídeo ExoPlayer con la dirección IP utilizada en la anterior actividad y se rescatan los valores de velocidad, aceleración y dirección. Al volver a la actividad original (salir de pantalla completa), los datos son enviados de vuelta junto al estado del sistema móvil, es decir, si está parado o no y su nueva dirección.

Capítulo 5. Conclusiones

En este TFG se ha realizado el diseño, montaje y programación de un sistema móvil que permite realizar grabaciones y visualización de video en directo omnidireccional mediante la cámara Insta360 Pro. El sistema móvil es controlado a través de una aplicación móvil, haciendo uso de los controladores integrados en la aplicación o a través de un mando de control remoto vía Bluetooth.

La aplicación permite tanto el control del sistema móvil como la visualización del vídeo en tiempo real en 360. El flujo de vídeo en directo 360 captado por la cámara puede ser visualizado en un reproductor de vídeo 360, pudiendo visualizar el vídeo en directo moviendo el dispositivo móvil o arrastrando con el dedo.

Los objetivos del proyecto que se han completado son los siguientes:

- El sistema puede adaptar su velocidad y dirección de manera progresiva y controlada
- El sistema móvil detiene su funcionamiento si pierde la conexión Bluetooth.
- El sistema móvil se controla tanto con los mandos de la aplicación como con el mando a distancia Bluetooth.
- Los temblores de la cámara se han reducido todo lo posible por medio de ajustar los tensores al colocar la cámara y el uso de gomas en el mástil para más estabilidad.
- El sistema móvil sigue la dirección indicada por el usuario en línea recta de la manera más exacta posible
- La aplicación permite al usuario conectarse y desconectarse del sistema móvil vía Bluetooth
- La aplicación móvil permite al usuario controlar la velocidad y aceleración del sistema móvil
- La aplicación móvil muestra el video en directo con lo que está captando la cámara Insta 360 en diversos formatos

Uno de los objetivos de este TFG, el cual que consistía en visualizar el vídeo en directo mediante un reproductor estereoscópico y en unas gafas de RV o HMD no se ha completado con éxito, debido a que la librería Exoplayer, utilizada para reproducir el vídeo en directo en 360 mediante flujo RTMP no contiene una solución para poder visualizar el contenido en este formato. Tras muchos intentos con otras librerías tanto de Java, como de Javascript utilizando páginas web para implementarlas en la aplicación móvil se llegó a la conclusión de que no existe en el mercado un reproductor de vídeo 360 estereoscópico que trabaje con el protocolo RTMP, RTSP o similares.

Para realizar este TFG he tenido que ampliar mis conocimientos en el desarrollo de aplicaciones en Android Studio, implementando librerías que nunca había utilizado, como puede ser *ExoPlayer* y gestionando el flujo de vídeo en directo en Android, así como el uso de sensores de aceleración y giroscopios, como el envío de datos mediante Bluetooth.

Ha sido un proyecto muy interesante desde el punto de vista de combinar diferentes tecnologías, tanto WiFi mediante RTMP para captar el flujo de video en directo 360, como Bluetooth para el envío de mensajes y control del sistema móvil.

También he aprendido mucho de la programación en Arduino y he ampliado conocimientos adquiridos tanto en la asignatura de Microprocesadores y acondicionadores de señal como en

la asignatura de Control. He aprendido a implementar un control PI en un sistema que gestiona 4 motores de manera simultánea, a interceptar desconexiones mediante el uso de *callbacks* y a leer datos de la señal de los pines de la ESP32 Feather y saber interpretarlos.

El cableado del sistema móvil me ha permitido afrontar un reto de electrónica más complejo que ningún otro que haya montado hasta la fecha y su correcta comprensión me fue de gran ayuda para diseñar la PCB utilizada.

5.1 Trabajo futuro

- Desarrollar un reproductor de vídeo 360 con visor estereoscópico compatible con la tecnología RTMP
- Añadir estabilidad al sistema móvil para evitar vibraciones presentes en el vídeo de la cámara

Bibliografía

- [1] Abel Martínez Martínez, *Programacion de robot móvil para vigilancia con visión artificial*, TFG, Universitat Politècnica de València, 2014. https://riunet.upv.es/bitstream/handle/10251/186491/Martinez%20Martinez%2c%20A._Programacion%20de%20robot%20movil%20para%20vigilancia%20con%20vision%20artificial.pdf?sequence=1&isAllowed=y (último acceso: junio 2023)
- [2] Javier Salvador Marco y Joan Martínez Recasens, TFG, *Montaje, calibración y programación del robot SR1*, Universitat Politècnica de Catalunya, 2007. https://upcommons.upc.edu/bitstream/handle/2099.1/4295/Mem%c3%b2ria.pdf?sequence=22&isAllowed=y (último acceso: junio 2023)
- [3] Rubén Cabalgante Ballesta, D*iseño de un robot móvil para estudio de la ocupación en habitaciones*, TFG, Universitat Carlos III de Madrid, 2007. https://e-archivo.uc3m.es/bitstream/handle/10016/23010/TFG_Ruben_Cabalgante_Ballesta.pdf?sequence=1&isAllowed=y (último acceso: junio 2023)
- [4] Viacheslav Brydko., *Diseño e implementación de múltiples interfaces para el control remoto de un vehículo y del sistema de visión embarcado*, TFG, Universitat Politècnica de València, 2014-2015. https://riunet.upv.es/bitstream/handle/10251/54222/BRYDKO%20-%20Dise%c3%b1o%20e%20implementaci%c3%b3n%20de%20m%c3%baltiples%20interfaces%20para%20el%20control%20remoto%20de%20un%20veh%c3%adculo%20y%20....pdf?sequence=2&isAllowed=y (último acceso: junio 2023)
- [5] Iván Alejandro Fernández Pacheco, *Diseño Y Desarrollo De Un Sistema De Vídeo-Vigilancia basado en tecnología Android,* TFG, Universitat Carlos III de Madrid, 2010. https://e-archivo.uc3m.es/bitstream/handle/10016/11209/PFC_Ivan_Alejandro_Fernandez_Pacheco.pdf?sequence=1&isAllowed=y (último acceso: junio 2023)
- [6] ExoPlayer Documentation. Artículo, Jekyll y TeXt Theme https://exoplayer.dev/ (último acceso: mayo 2023)
- [7] Insta360 Pro Documentation. https://www.insta360.com/es/product/insta360-pro/ (último acceso: abril 2023)
- [8] Google Cardboard NDK. Artículo, Google for Developers https://developers.google.com/cardboard/develop/c/quickstart?hl=es-419 (último acceso: junio 2023)
- [9] Raquel Nerea Bueno Bueno., *Experimentando con cámaras 360, realidad virtual y el guion interactivo para contar historias*, TFG, Badajoz, 2018. https://dehesa.unex.es/bitstream/10662/9880/1/TFGUEX_2019_Bueno_Bueno.pdf (último acceso: junio 2023)
- [10] Qué es y cómo funciona una cámara 360. Artículo, GrupoAudioVisual.com https://grupoaudiovisual.com/camara-360/ (último acceso: junio 2023)
- [11] Motor DC con Encoder – Velocidad – Posición. Artículo, Sergio Andrés Castaño Giraldo https://controlautomaticoeducacion.com/arduino/motor-dc-encoder/ (último acceso: mayo 2023)
- [12] Jon Goñi Amatriain., *IPTV. Protocolos empleados y QoS*. Artículo. https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/trabajos/resumenes/gr16-QoSEnIPTV.pdf (último acceso: mayo 2023)

- [13] Valentina Espinoza, Gastón Quevedo, Alejandro Romero, Pablo Troncoso., Protocolo RTMP, Artículo, Universidad Técnica Federico Santa María, 2019. http://profesores.elo.utfsm.cl/~agv/elo322/1s19/projects/reports/Protocolo_RTMP.pdf (ultimo acceso: junio 2023)
- [14] Cómo interpretar el ciclo de vida de una actividad. Artículo, Google for Developers https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es-419 (ultimo acceso: mayo 2023)
- [15] Teoría del control. Obtenida el 30/05/2023, Artículo, Wikipedia, La enciclopedia libre https://es.wikipedia.org/wiki/Teor%C3%ADa_del_control (ultimo acceso: junio 2023)
- [16] Javier Valls Coquillat. *Apuntes de la asignatura de Control*, Universitat Politècnica de València, 2019.
- [17] Altium Documentation. Altium LLC https://www.altium.com/es (último acceso: abril 2023)
- [18] Android Studio: Ciclo de Vida de los Activities. Artículo, Santi Pérez Gómez. https://forum.huawei.com/enterprise/es/android-studio-ciclo-de-vida-de-los-activities/thread/667226358524297216-667212895009779712 (último acceso: julio 2023)
- [19] Presentación de 360ViSi en la UCV. Artículo, Quasar Dynamics. https://quasardynamics.com/360visi-video-inmersivo/ (último acceso: julio 2023)
- [20] Antoni Juan Lozano, *SPHERISION*, TFG, Universitat Politècnica de Mataró, 2018. https://upcommons.upc.edu/bitstream/handle/2117/102742/TFG%20-%20Spherision.pdf?sequence=1&isAllowed=y (último acceso: junio 2023)
- [21] Jair Daynor López Gutiérrez, *Diseño, implementación y evaluación de un reproductor basado en web para vídeo omnidireccional, siguiendo las especificaciones de sincronización del estándar HbbTV v2.0.1,* TFG, Universitat Politècnica de València, 2019. https://riunet.upv.es/bitstream/handle/10251/128458/L%c3%b3pez%20-%20Dise%c3%b1o%2c%20implementaci%c3%b3n%20y%20evaluaci%c3%b3n%20de%20un%20reproductor%20basado%20en%20web%20para%20v%c3%addeo%20omnidirecc....pdf?sequence=1&isAllowed=y (último acceso: junio 2023)
- [22] Videos en 360 grados: una innovadora forma de empatizar con las audiencias. Artículo, Think with Google. https://www.thinkwithgoogle.com/intl/es-419/estrategias-de-marketing/video/videos-360-innovacion-audiencias/ (último acceso: junio 2023)
- [23] Insta360 Pro Download. https://www.insta360.com/es/download/insta360-pro (último acceso: julio 2023)
- [24] Review Meta Quest 2. Artículo, Hardzone https://hardzone.es/noticias/equipos/impresiones-meta-quest-2/ (último acceso: junio 2023)
- [25] Rendering 3D video/image for google cardboard. Artículo, Blender StackExchange https://blender.stackexchange.com/questions/39190/rendering-3d-video-image-for-google-cardboard (último acceso: junio 2023)