SISTEMA DE DETECCIÓN DEL ESTADO DE LA CALZADA BASADO EN SENSORES DE VIBRACIÓN Y SISTEMAS DE CÓMPUTO DE BAJO COSTE

TRABAJO DE FIN DE TÍTULO

Grado en Ingeniería Informática
Universidad de Las Palmas de Gran Canaria
Instituto Universitario de Ciencias y Tecnologías Cibernéticas

Tutores:
Alexis Quesada Arencibia
Carmelo Rubén García Rodríguez

ORLANDO BELLOCH DÍAZ

Octubre 2019

ÍNDICE

ÍNDICE DE ILUSTRACIONES

AGRADECIMIENTOS

Este proyecto ha sido posible gracias al apoyo de mi tutor Alexis Quesada Arencibia, que con sus conocimientos y experiencia ha actuado de guía durante todo el desarrollo. Quiero agradecer también a los miembros del Instituto Universitario de Ciencias y Tecnologías Cibernéticas su compañerismo y ayuda desinteresada. A todas aquellas personas que me han acompañado durante la carrera y con los que he compartido momentos inolvidables. A mis amistades, por todo el cariño y por darme fuerzas cuando más lo he necesitado. Y a mi familia, que siempre me ha dado lo mejor y que ha sido la mayor de las motivaciones.

1. INTRODUCCIÓN

El estado de las carreteras es un problema que nos afecta a todos, según un estudio de la Asociación Española de la Carretera “uno de cada trece kilómetros de la red de carreteras española presenta deterioros relevantes en más del 50% de la superficie del pavimento, acumulando baches, roderas y grietas longitudinales y transversales” [1]. Esto puede provocar que algunos de los sistemas del vehículo, como son el sistema de suspensión o el sistema de dirección, se vean afectados puesto que tienen que absorber un mayor número de irregularidades [2]. Además, este tipo de desperfectos en la vía suponen una pérdida de comodidad, eficiencia y seguridad para el conductor. Es por esto por lo que es necesario tener sistemas que permitan detectar problemas en el estado de la calzada. Actualmente existen sistemas de supervisión basados en procesamiento de imágenes que permiten identificar y localizar este tipo de daños, pero requieren de una alta capacidad de cómputo y, por tanto, son sistemas de elevado coste.

Este proyecto pretende dar una nueva solución usando sistemas de cómputo de bajo coste, como la placa Arduino UNO Rev3. Con ella, un acelerómetro, un GPS y una tarjeta microSD se ha creado un prototipo que es capaz de detectar el estado de la calzada a partir de la vibración del vehículo.

De manera paralela al proyecto se ha desarrollado una aplicación con las mismas funcionalidades que este dispositivo con la intención de comparar los resultados de ambos sistemas y poder determinar con ello cuál es la mejor solución al problema.

2. ESTADO DEL ARTE

Un pavimento en mal estado, es decir, que posee baches, grietas, deformaciones, agua o hielo, entre otros, puede provocar averías o incluso un accidente. Es por ello por lo que se han ido desarrollando distintos sistemas que detectan estos problemas. Actualmente existen varios sistemas que se encargan de determinar el estado de la calzada. Estos sistemas se pueden agrupar en dos tipos en función del método que utilicen [3].

Por un lado, están los sistemas basados en la causa "cause-based", es decir, que intentan identificar los factores que afectan al coeficiente de fricción y a partir de esto determinar el estado de la calzada. Para estos métodos son necesarios sensores ópticos, radares o cámaras. Por ejemplo, existen medidores ópticos no intrusivos, es decir, que no se embeben en la calzada, que detectan el estado de la calzada y el coeficiente de fricción a partir de la altura de la película de agua o hielo que se encuentra presente en el pavimento. También existen sistemas que utilizan medidores de temperatura colocados a nivel de suelo, que detectan la temperatura de congelación mediante un sensor. Otros dispositivos más elaborados embebidos en la carretera pueden detectar la altura de la película de agua hasta 4 milímetros, la temperatura de congelación para varios materiales de descongelación como pueden ser NaCl, MgCl o CaCl, el coeficiente de fricción y el porcentaje de hielo. También son capaces de detectar si la calzada está seca, húmeda, si hay hielo o nieve, sal residual o lluvia helada.

Por otro lado, se encuentran los sistemas basados en el efecto "effect-based". Estos sistemas utilizan la información generada en tiempo real por los sensores del vehículo en circulación. Estos detectan la deformación, el ruido o la vibración de la rueda para saber cuál es el estado de la calzada. También existen sistemas que controlan los cambios de altura en la suspensión del vehículo, logrando con ello detectar baches y su profundidad. Hay sistemas que envían esta información, junto con la posición GPS, a otros vehículos, consiguiendo que estos ajusten la suspensión evitando el daño que pueda provocar el bache o llegando incluso a esquivarlo. La marca que ha implementado este sistema también ha desarrollado un sistema que permite detectar los baches antes de pasar por encima usando una cámara frontal. Estos sistemas permiten a los vehículos esquivar los baches o reducir la velocidad para minimizar el impacto de la colisión.

El dispositivo desarrollado en este trabajo entraría dentro del segundo grupo ya que detecta en tiempo real el estado de la calzada a partir de la vibración del vehículo en circulación. La principal diferencia con los ejemplos mencionados anteriormente es que se trata de un sistema independiente, por lo que puede ser utilizado en cualquier vehículo a un coste muy bajo.

Aunque no están directamente relacionados con el problema a tratar en este trabajo, también existen sistemas que detectan el peso de vehículos en movimiento a partir de la vibración del pavimento cuando estos circulan sobre él. Para obtener esta información se utilizan varios sensores, entre ellos sensores de aceleración como el utilizado en este proyecto. Todos estos sensores se controlan mediante un punto de acceso (Access Point, AP) que sincroniza todos los sensores y almacena la información.

A continuación, se muestra una tabla resumen con algunos de los dispositivos y sistemas encontrados:

Dispositivo	Características
Sensor de Calzada TCS [4]	Sensor que se embebe en la calzada y detecta la temperatura superficial y el espesor de la película de agua y de hielo. Además, se caracteriza por tener un consumo bajo de energía.
Sensor de Asfalto WSS [5]	Sensor que se embebe en la calzada y detecta la temperatura superficial, el espesor de la película de agua y de hielo, detección del punto de congelación y la presencia de agentes químicos. Además, se caracteriza por tener un consumo bajo de energía.
Medidor de Estado de la Calzada: RCM411 [6]	Sensor óptico que se coloca en vehículos y que permite medir el estado de la superficie de la calzada, el coeficiente de fricción y la altura de la película de agua o hielo
Sensor de Estado de la Calzada: M201 [7]	Sensor que se embebe en la calzada y es capaz de medir el coeficiente de fricción, la temperatura, el punto de congelación, la concentración de sal y la altura de la película de agua o hielo.
Sensor Óptico de Estado de la Calzada: M251 [8]	Sensor óptico que se instala sobre un pórtico de carretera y es capaz de medir el coeficiente de fricción, la temperatura, el punto de congelación, la concentración de sal, la altura de la película de agua o hielo y la altura de la nieve presente.
Sensor de Carretera Inteligente y Activo ARS31Pro-UMB [9]	Sensor que se embebe en la calzada y es capaz de determinar la temperatura de congelación.
Sensor inteligente de carretera IRS31Pro-UMB [9]	Sensor que se embebe en la calzada y registra la temperatura, la altura de la película de agua, la temperatura de congelación, el estado de la carretera, la fricción y el porcentaje de hielo.
Sensor de carretera sin contacto NIRS31-UMB [9]	Sensor óptico que detecta el estado de la calzada, el grosor de la película de agua, la temperatura de congelación y la fricción.
Active Sensor for Measurement of Freezing Point Temperature [10]	Sensor que se embebe en la calzada y es capaz de detectar la temperatura de congelación.
Intelligent Passive Sensor for Transportation Systems: IT-Sens [11]	Sensores que se embeben en la calzada y que son capaces de detectar el estado de la carretera, la temperatura de la superficie, la película de agua, la temperatura de congelación y la sal residual.
In-pavement wireless Weigh-In-Motion [12]	Sensor que detecta la vibración del pavimento para determinar el peso de los vehículos que circulan por un tramo de carretera. También es capaz de registrar las llegadas y salidas de los vehículos.
Pavement pothole detection and severity measurement using laser imaging [13]	Sensor que utiliza un láser para obtener imágenes de la calzada para después analizarlas utilizando una red neuronal. Con esto se puede determinar las dimensiones y la profundidad de los baches.
Pothole detection and tracking in car video sequence [14]	Sistema para detectar baches mediante el uso de grabaciones realizadas desde un vehículo en circulación. Los videos son tratados mediante algoritmos que detectan el tamaño y la forma de los baches.

Pothole Detection Using Computer Vision and Learning [15]	Sistema para la detección de baches en el asfalto utilizando visión estereoscópica en un vehículo en circulación. La información obtenida se trata utilizando dos modelos de redes neuronales que son capaces de detectar baches.
Vehicle Vibration Signal Processing for Road Surface Monitoring [16]	Sistema que utiliza el sensor acelerómetro de un teléfono inteligente para detectar los cambios de aceleración en el eje vertical de un vehículo en circulación y, mediante un algoritmo, clasificar los baches y valora su gravedad.

Como se mencionó en la introducción, se ha desarrollado una aplicación móvil que tiene la capacidad de detectar la vibración del dispositivo usando un acelerómetro y de obtener la posición utilizando el sensor GPS. Actualmente, en el mercado existen multitud de aplicaciones que realizan alguna de estas funciones por separado. A continuación, se muestra una tabla con las aplicaciones encontradas en la *PlayStore* de Google:

Aplicación	Características
Accelerometer Meter [17]	Aplicación que detecta los cambios de aceleración en los ejes utilizando el acelerómetro del teléfono móvil. Esta información la muestra en tiempo real en una gráfica y la guarda en un archivo de texto.
Full system info [18]	Aplicación que muestra información detallada del hardware y del software del teléfono móvil. (No permite obtener datos de los sensores)
Device Info [19]	Aplicación que muestra información detallada del hardware y del software del teléfono móvil. (No permite obtener datos de los sensores)
Sensores Multiherramienta [20]	Aplicación que muestra información detallada del hardware y del software del teléfono móvil. Permite utilizar los sensores del dispositivo, mostrando la información en gráficas. Además, permite realizar tomas de datos solamente en un instante de tiempo concreto, no durante un periodo de tiempo.
Sensores Multiherramienta [21]	Aplicación que muestra información detallada del hardware y del software del teléfono móvil. Permite utilizar los sensores del dispositivo, mostrando la información en gráficas. Sin embargo, no permite almacenar la información obtenida.

3. OBJETIVOS

El objetivo de este trabajo es analizar, diseñar e implementar un prototipo de sistema capaz de medir la calidad del asfalto a partir de la vibración de un vehículo en circulación utilizando sistemas de cómputo de bajo coste. Además, el dispositivo debe ser capaz de obtener la posición en la que se encuentra gracias a un GPS y guardar la información obtenida de los sensores en un dispositivo de almacenamiento externo como puede ser una tarjeta microSD. Esta información debe tener un formato que facilite su posterior tratamiento.

4. JUSTIFICACIÓN DE LAS COMPETENCIAS ESPECÍFICAS CUBIERTAS

La competencia especifica que ha sido trabajada durante el desarrollo de este proyecto es la *TI02*. Esta competencia pertenece a la mención de tecnologías de la información y fomenta lo siguiente: "Capacidad para seleccionar, diseñar, desplegar, integrar, evaluar, construir, gestionar, explorar y mantener las tecnologías de hardware, software y redes, dentro de los parámetros de coste y calidad adecuados".

Se considera que esta competencia ha sido cubierta puesto que, durante el desarrollo de este trabajo, se han seleccionado, integrado, evaluado y explorado varias tecnologías hardware y software, como por ejemplo la utilización de un acelerómetro y de un GPS. También se ha cubierto el diseño y posterior construcción de un sistema que usa tecnologías hardware y software. Esto se ha demostrado al combinar exitosamente todos los componentes hardware para la creación del prototipo, y su control mediante librerías escritas en C++. Todo esto se ha desarrollado teniendo en cuenta que debe ser un sistema de bajo coste.

Del total de las competencias comunes de la Ingeniería Informática se considera que han sido cubiertas las siguientes:

- *CII01:* Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
    - Esta competencia se ve reflejada en el apartado de "Legislación y normativa", en el que se detalla la influencia en este trabajo de las leyes y normativas relacionadas con el tratamiento de los datos personales.

- *CII06:* Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.
    - Tanto el desarrollo del dispositivo Arduino como el de la aplicación para Android requieren de la aplicación de algoritmos para el control de las funciones implementadas. En particular, el dispositivo Arduino requiere de algoritmos bien analizados y optimizados puesto que los recursos son limitados.

- *CII08:* Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
    - En el proceso de creación de la aplicación para Android se decide utilizar el framework IONIC que permite la creación de software robusto, seguro y eficiente.

- *CII016:* Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.
    - Al comienzo de este proyecto se valoran diversas metodologías y ciclos de vida buscando aquella que mejor se adapte al desarrollo de la solución. Finalmente, se decide utilizar una metodología iterativa e incremental.

- *CII018:* Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.
    - Al igual la competencia "CII01" descrita anteriormente, en el proceso de desarrollo de este proyecto fue necesario consultar y entender las normativas relacionadas con el tratamiento de datos personales. Dos de estas son la LOPD (Ley Orgánica de Protección de Datos) y la RGPD (Reglamento General de Protección de Datos).

5. APORTACIONES AL ENTORNO SOCIO ECONÓMICO

Como se ha mencionado anteriormente, el estado de las carreteras es un problema que afecta a todos los ciudadanos, por ello este proyecto pretende facilitar la tarea de comprobación del estado del pavimento gracias a un sistema de bajo coste que es capaz de tomar datos del asfalto en tiempo real mientras se circula. La intención es que esta información sea facilitada a los organismos encargados del mantenimiento de las carreteras y que puedan saber con mayor exactitud cuales son las vías más deterioradas y que por lo tanto requieren de una mayor atención. Además, esto permitiría una mejor distribución del dinero y de los recursos para las tareas de mantenimiento de las carreteras.

Las posibles mejoras de las condiciones del asfalto conllevarían una menor inversión económica por parte de los usuarios en el mantenimiento de su vehículo y una disminución del estrés y la fatiga a la hora de conducir. También se podría llegar a reducir el número de accidentes, ya que las probabilidades de pérdida de control del vehículo disminuyen en carreteras donde no hay grietas, baches y deformaciones.

Además, tanto el dispositivo como la aplicación Android son sistemas fáciles de utilizar. En el caso del dispositivo solo hay que alimentarlo con corriente eléctrica y esperar a que comience a tomar datos y en la aplicación solo es necesario presionar un botón. Además, ambos sistemas son portables lo que permite monitorizar todo tipo de vías con cualquier vehículo. A esto hay que sumarle que son sistemas de bajo coste y que pueden hacer sinergias con otros sistemas gracias a que la información generada se almacena siguiendo un estándar mundialmente conocido, como es JSON. También hay que tener en cuenta que se utiliza, en el caso del dispositivo, un sistema de almacenamiento externo lo que permite mover la información entre distintos dispositivos.

6. LEGISLACIÓN Y NORMATIVA

En todo proyecto se debe velar por el cumplimiento de la legislación y la normativa vigente y, aunque en este proyecto no se tratan datos personales de nivel alto (como son las creencias, la religión, la ideología, etc), se debe tener en cuenta que la geolocalización, utilizada en este proyecto tanto en el dispositivo como en la aplicación móvil, está contemplada por la LOPD (Ley Orgánica de Protección de Datos). En ella se especifica que el desarrollador de la aplicación, cuando recoge y trata datos de carácter personal, debe cumplir con los siguientes puntos [22]:

- Debe informar al usuario del uso y del tratamiento de los datos de carácter personal.
- Se debe obtener el consentimiento por parte del usuario antes de que este pueda utilizar la aplicación
- El usuario debe ser informado de sus derechos ARCO, es decir, de su derecho al acceso, rectificación, cancelación y oposición de sus datos personales.
- Se debe tener unas medidas adecuadas de seguridad
- Se debe definir el periodo de conservación de los datos.

Otro aspecto de este trabajo que está controlado por la legislación, es la grabación “onboard”, es decir, aquellas grabaciones realizadas desde dentro de un vehículo en las que aparece la vía por la que se circula y los alrededores. Aunque normalmente estas grabaciones se utilizan como prueba en caso de accidente o para denunciar acciones irregulares por parte de otros vehículos, en este proyecto se ha utilizado como método para contrastar la información generada por el dispositivo y la aplicación. Según la AEPD (Agencia Española de Protección de Datos)[23], este tipo de grabaciones son legales y no merecen de la aplicación de ningún precepto de las normas vigentes en materia de protección de datos, siempre y cuando tengan una finalidad “doméstica” o íntima, es decir, que no sean publicadas y compartidas en internet. Sin

embargo, en el caso de que en ellas apareciera un individuo o algún elemento (una matrícula) que pudiera identificar a una persona, debería actuarse siguiendo la normativa.

Además, la RGPD indica que el tratamiento de datos personales en el ámbito científico está sujeto al principio de minimización de los datos personales [24]. Este principio tiene como objetivo evitar la identificación de los individuos. La normativa pide también que las grabaciones no se hagan de manera indiscriminada, sino que solo se realice la grabación cuando sea necesario. Es por ello por lo que se han tapado las matrículas en los videos y solo se han realizado las grabaciones en recorridos concretos.

7. METODOLOGÍA Y RECURSOS UTILIZADOS

7.1. METODOLOGÍA

Para llevar a cabo este proyecto se ha utilizado una metodología iterativa e incremental, que divide el proceso de desarrollo en bloques temporales denominados iteraciones [25]. Estos bloques están compuestos de cuatro fases. En primer lugar, una fase de análisis del problema que se quiere abarcar, valorando todos los requisitos que están relacionados. En segundo lugar, una fase de diseño de la solución que se va a llevar a cabo. En tercer lugar, una fase de desarrollo en la que se implementa lo diseñado en la fase anterior. Por último, el resultado se debe probar y para ello está la cuarta fase denominada "fase de pruebas". En esta fase se prueba el resultado en un entorno real, comprobando con ello que lo desarrollado funciona correctamente.

Al ser una metodología incremental permite añadir funcionalidades al producto siguiendo siempre las cuatro fases descritas anteriormente, logrando al finalizar la iteración un producto funcional con mejoras. Para añadir estas funcionalidades es necesario partir de una base que se desarrolla en la primera iteración.

Este modelo se caracteriza principalmente por facilitar la integración de cambios en el proyecto durante el desarrollo de este. También destaca por reducir los riesgos desde el comienzo del desarrollo, ya que desde un primer momento se pueden detectar los problemas, pudiendo solucionarlos en las siguientes iteraciones. Además, se minimiza el número de errores que se produce a lo largo del desarrollo y por lo tanto aumenta la calidad del producto.

7.2. PLANIFICACIÓN

Al comienzo del desarrollo de este proyecto se propone, en el documento de propuesta de trabajo de fin de título, un posible plan de trabajo dividido en cuatro fases con el número de horas estimadas de duración. Cada una de estas fases posee a su vez varias tareas que la componen.

- *Estudio previo / análisis (60 horas):*
  - Estado del arte
  - Búsqueda del hardware
  - Búsqueda de lenguajes de programación y tecnologías
- *Diseño / desarrollo / implementación (170 horas):*
  - Diseño del sistema
  - Diseño e implementación del dispositivo
- *Evaluación / validación / prueba (30 horas):*
  - Prueba del dispositivo en un entorno real
  - Comprobación de los datos obtenidos

- *Documentación / presentación (40 horas):*
    - Redacción de la memoria
    - Preparación de la defensa

Como se puede observar en la representación gráfica que se muestra a continuación (Ilustración 1), esta planificación conlleva que el desarrollo se realice de manera secuencial sin posibilidad de repetir alguna de las fases. Sin embargo, la metodología de trabajo elegida es iterativa e incremental, por lo que es necesario actualizar el plan de trabajo.

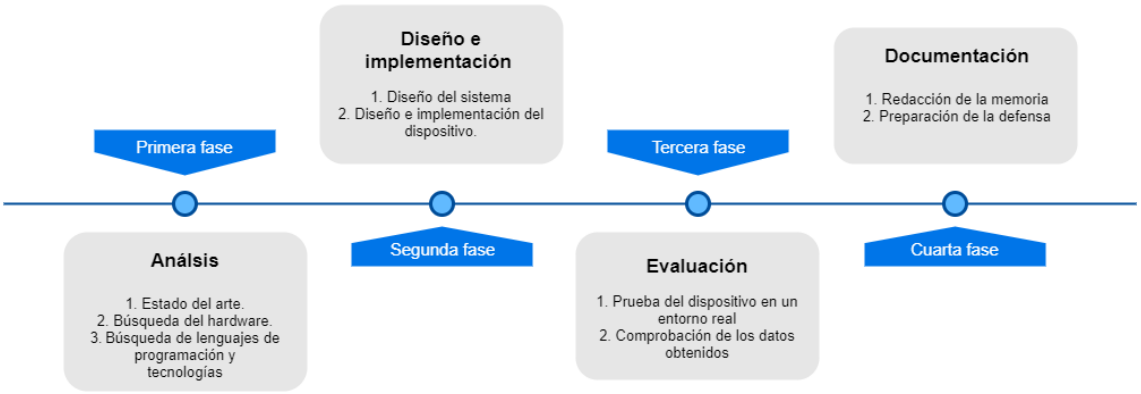


Ilustración 1 – Plan de trabajo estimado.

A continuación, se muestra la planificación real (Ilustración 2) que se llevó a cabo durante el desarrollo del proyecto. Como se puede observar, la segunda fase, correspondiente con el diseño e implementación, se bifurca para representar el desarrollo en paralelo del dispositivo Arduino y la aplicación Android. También se puede observar que desde la tercera fase (la fase de evaluación) se puede volver a la primera fase. Esto es debido a la metodología utilizada, al finalizar una iteración se debe volver la fase de análisis.

Ilustración 2 – Plan de trabajo real.

Llevar el diagrama anterior a la práctica provoca que las tareas planteadas en el documento de propuesta de trabajo de fin de título también se modifiquen. A continuación, se muestra de manera resumida los pasos que se realizaron a lo largo del desarrollo del proyecto:

1. Investigación del estado del arte y búsqueda del hardware
2. Iterativamente se ponen a prueba cada uno de los componentes hardware del dispositivo.
    - Al adquirir un componente nuevo se realizan con él las fases de análisis, diseño, implementación y prueba.
3. De manera paralela, se desarrolla una aplicación para Android.
    - El desarrollo de esta aplicación también se realiza de manera iterativa e incremental, añadiendo en cada iteración nuevas funciones.
4. Cuando todos los componentes hardware han sido testeados, se procede a la unión de todos ellos utilizando la placa Arduino Uno Rev3.
5. Con la aplicación para Android y el dispositivo Arduino terminados, se procede a la prueba de ambos sistemas en un entorno real.
6. Finalmente, con el resultado de ambos dispositivos se comienza el desarrollo de esta memoria y la preparación de la defensa.

7.3. RECURSOS SOFTWARE UTILIZADOS

Desarrollar un proyecto en el campo de la Informática requiere utilizar, casi de manera obligatoria, todo tipo de software. A continuación, se detallan los recursos software utilizados para el desarrollo de este trabajo:

7.3.1. GIT

Git es un programa de control de versiones distribuido, escalable y rápido, desarrollado en *C*, *Bourne Shell* y *Perl*. Se distribuye bajo la licencia *GNU GPL v2*, por lo que se considera que es software libre. Permite llevar un registro de los cambios en los archivos de un proyecto, con el objetivo de controlar las distintas versiones de un código fuente [26], [27].

Git fue creado por Linus Torvalds, con la colaboración de Junio Hamano y la organización Software Freedom Conservancy. Su lanzamiento inicial fue el 7 de abril de 2005 y actualmente está siendo supervisado por J. Hamano.

Esta herramienta de control de versiones se ha utilizado en este trabajo para gestionar el código creado para el dispositivo y para la aplicación móvil.

7.3.2. GITKRAKEN

GitKraken es una interfaz de usuario para *Git* creada por la empresa Electron. Permite tener una representación visual del repositorio en el que se está trabajando, evitando así usar la línea de órdenes [28].

Además, permite la integración con plataformas como *GitHub*, *Bitbucket*, *Gitlab*, etc. Lo que convierte a esta herramienta en idónea para el desarrollo de programas informáticos.

7.3.3. BITBUCKET

Bitbucket es una plataforma web que permite alojar y gestionar, de manera centralizada, repositorios *Git* y *Mercurial*. Además, posee características como control de acceso a los archivos y control del flujo de trabajo. Esta herramienta ha sido desarrollada con *Python* por la empresa Atlassian y fue lanzada al mercado en el año 2008 [29], [30].

En esta plataforma, junto con los programas *Git* y *GitKraken*, se ha llevado a cabo la gestión de los distintos repositorios usados en el desarrollo de este trabajo.

7.3.4. ARDUINO IDE

Arduino IDE es un entorno de desarrollo integrado que proporciona una serie de herramientas para facilitar el trabajo a desarrolladores y programadores en la creación de software para *Arduino*. Este *IDE* ha sido escrito en el lenguaje de programación *Java* por la empresa *Arduino Software* [31].

7.3.5. LIBRERÍAS PARA ARDUINO

Para poder controlar todos los componentes utilizados en el dispositivo *Arduino* fue necesario hacer uso de librerías. Una librería es un fragmento de código que permite añadir nuevas funcionalidades a un proyecto y/o solucionar un problema en concreto. Esto permite hacerlo de manera rápida ya que todo el código ya está implementado y solo es necesario integrarlo en el proyecto que está en desarrollo. Además, si la librería es de reconocido prestigio, es una forma fiable de añadir una nueva funcionalidad, ya que ya ha sido probada por otros desarrolladores.

Con el entorno de desarrollo de *Arduino* (*Arduino IDE*) vienen instaladas varias librerías estándar [32]. Algunas de estas se han usado en este proyecto:

- *La librería SD:* para la lectura y escritura en tarjetas SD.
- *La librería Wire:* para la transmisión de información utilizando los protocolos TWI (*Two Wire Interface*) o I2C (*Inter-Integrated Circuit*).
- *La librería SoftwareSerial:* para poder utilizar comunicaciones seriales en cualquier puerto digital.

Del mismo modo se han utilizado librerías externas, creadas por particulares y organizaciones. Estas son las siguientes:

- *La librería TinyGPS*: Creada por Mikal Hart, está diseñada para proveer al usuario de la posición, fecha, hora, altitud, velocidad y rumbo de su dispositivo GPS, manteniendo un consumo de energía bajo. Para poder obtener esta información la librería decodifica los datos obtenidos del GPS que se encuentran en formato NMEA.
- *La librería LiquidCrystal_I2C:* Al usar un adaptador para transformar el bus serie de la pantalla LCD en un bus I2C, fue necesario buscar una librería que controlara la pantalla usando este bus. La opción elegida fue la librería *LiquidCrystal_I2C* [33]. Esta posee las mismas funcionalidades que la librería *LiquidCrystal* de *Arduino*, es decir, la librería puede inicializar la pantalla, mover el cursor, imprimir un texto, moverlo, controlar la luz del fondo y crear un carácter personalizado.

7.3.6. WEBSTORM

WebStorm es un entorno de desarrollo integrado de la empresa JetBrains pensado para *frameworks* como *Angular*, *Ionic*, *Node.js*, entre otros. Se caracteriza por su sistema de detección de errores automático, su sistema de navegación y su herramienta de refactorización [34].

7.3.7. NPM

npm (*Node Package Manager*) está formado por tres componentes [35]–[37]. En primer lugar, posee el mayor registro del mundo. En él se encuentran más de 800000 paquetes de código abierto para *Node.js*, compartidos por toda una comunidad de desarrolladores. En segundo lugar, *npm* es un *CLI*, es decir, una interfaz de línea de comando. Con ella se realiza la gestión e instalación de paquetes de *Node*. Y, en tercer

lugar, *npm* posee un sitio web usado para descubrir paquetes, crear y configurar una cuenta de usuario y crear Orgs (Organizaciones) para controlar el acceso a paquetes públicos y privados.

npm ha sido desarrollado por Isaac Z. Schlueter y publicado en enero del 2010. El lenguaje de programación usado para su creación es *JavaScript*.

7.3.8. NVM FOR WINDOWS

nvm (*Node Version Manager*) for *Windows* es un gestor de versiones de *Node* desarrollado para el sistema operativo *Windows*, basado en el *nvm* [38] para *Linux/Mac*. Permite instalar varias versiones de *node.js* en un ordenador [39].

Ha sido desarrollado con el lenguaje de programación *Go* y se distribuye bajo la licencia *MIT*.

7.3.9. FRITZING

Fritzing es un programa de código abierto para crear esquemas eléctricos en proyectos con *Arduino*. Dispone de bibliotecas con componentes propios de *Arduino* y de otras empresas. Además, permite crear una placa de circuito impreso (*Printed Circuit Board*, *PCB*) a partir del esquema diseñado [40], [41].

El desarrollo de *Fritzing* fue iniciado por la empresa FH Potsdam y actualmente lo dirige la fundación Friends of Fritzing. El código fuente del programa está escrito en *C++* usando el *framework Qt*.

7.3.10. MENDELEY

Mendeley es un gestor de referencias y red social académica que tiene como objetivo facilitar la organización de las investigaciones. Su desarrollo comenzó en el año 2007 por Elsevier y fue publicada en el año 2008.

Entre sus características principales destaca su generador automático de citas y bibliografías, su aplicación multiplataforma y el acceso a documentos en cualquier momento vía online. Además de la aplicación, *Mendeley* posee un plugin para *Microsoft Office Word*, que permite integrar la información almacenada en el documento que se está escribiendo, y una extensión para navegadores que permite guardar automáticamente en la aplicación la web que se está visitando.

7.3.11. MICROSOFT OFFICE WORD

Microsoft office Word es el software utilizado para redactar esta memoria. Se trata de un procesador de textos creado por la compañía Microsoft. Su desarrollo comenzó en el año 1983 con la primera versión para el sistema operativo *MS-DOS*. En el año 1989 fue cuando se lanzó esta herramienta para el sistema operativo *Windows* [42], [43].

Permite al usuario crear documentos en los que se puede redactar textos, añadir imágenes e insertar tablas entre otras funciones. Además, se caracteriza por tener un formato de texto propio, el formato DOC (“Document”). Más tarde surgiría un nuevo formato llamado “DOCX”, que usa el lenguaje de marcado XML y que permite una mayor comprensión del archivo [44], [45].

Este software se comercializa normalmente en un paquete de programas informáticos llamado “Office 365”, compuesto de aplicaciones como *Excel* o *PowerPoint*.

7.3.12. TIMESTAMP CAMERA FREE

Se trata de una aplicación [46] para teléfonos inteligentes con el sistema operativo Android que permite la grabación de vídeos añadiendo, en la propia imagen, la fecha y hora, la posición GPS y la zona en la que se encuentra. Esto es muy útil para la fase de pruebas del dispositivo y de la aplicación porque permite saber el momento y el lugar en el que se circuló sobre un bache, pudiendo así contrastar la información obtenida de los sistemas con lo que realmente ocurrió.

7.4. RECURSOS HARDWARE UTILIZADOS

El desarrollo de este proyecto requirió de la utilización de recursos hardware. A continuación, se describen todos ellos:

7.4.1. ARDUINO UNO REV3

Arduino Uno Rev3 es una placa de prototipado, cuyas dimensiones son 8 x 5.5 x 2.5 cm, basada en el microcontrolador AVR ATmega328P de la empresa ATMEL. Posee 14 pines digitales de entrada/salida, 6 pines de entrada analógicos, una velocidad de reloj de 16 MHz, un conector USB, una clavija de alimentación, conectores ICSP y un botón de reinicio [47].

La placa *Arduino Uno Rev3* puede ser programada gracias al protocolo SKT500 y al *bootloader* (*firmware*) que permite cargar nuevos códigos sin tener que recurrir a herramientas hardware externas. Todo el código puede ser desarrollado en el entorno de desarrollo integrado de *Arduino* (*Arduino IDE*), teniendo en cuenta que la placa dispone solo de 32 KB de memoria *Flash* en el microcontrolador, además de 2 KB en una memoria *SRAM* y 1 KB en una memoria *EEPROM*.

Existen varias formas de alimentar a la placa *Arduino Uno Rev3* con corriente eléctrica. Normalmente se usa el conector USB o una fuente externa, como un transformador AC-DC o una batería, mediante el conector de alimentación. También se pueden usar los pines de alimentación GND y Vin. Independientemente del método utilizado, la placa soporta un rango de 6 a 20 voltios, siendo lo recomendable entre 7 y 12 voltios. Fuera de este rango la placa es inestable y puede sufrir daños, debido a que el regulador de voltaje se sobrecalienta.

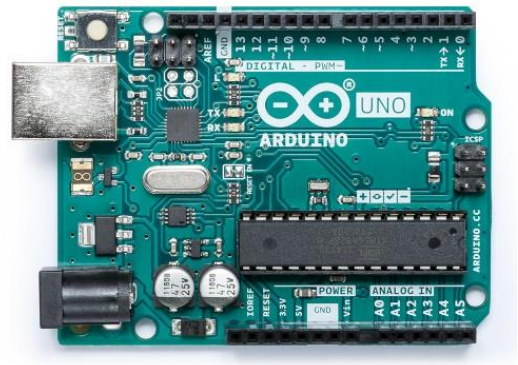


Ilustración 4 – Parte frontal Arduino UNO Rev3.

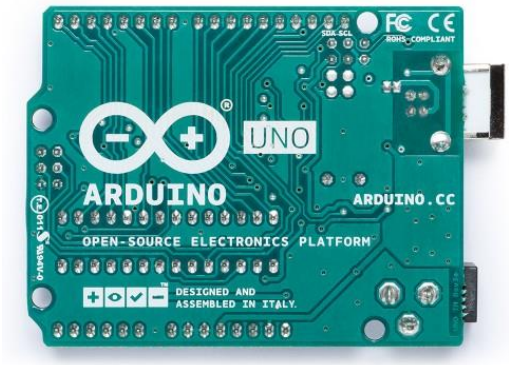


Ilustración 3 – Parte trasera Arduino UNO Rev3.

7.4.2. PLACA DE PROTOTIPADO

Una placa de prototipado (en inglés *protoboard*) es una tabla de plástico con orificios conectados eléctricamente entre sí, pensada para construir prototipos de circuitos electrónicos. Estas placas están formadas por tableros de plástico perforados y láminas de una aleación de cobre, estaño y fósforo que unen los orificios siguiendo el patrón de la Ilustración 6 [48]:

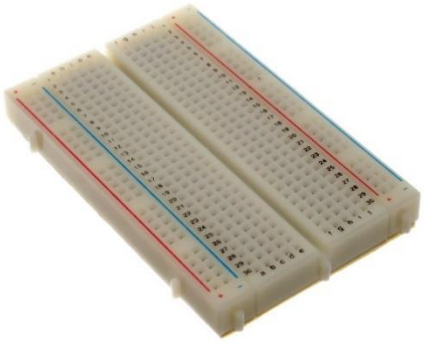


Ilustración 5 – Placa de prototipado.

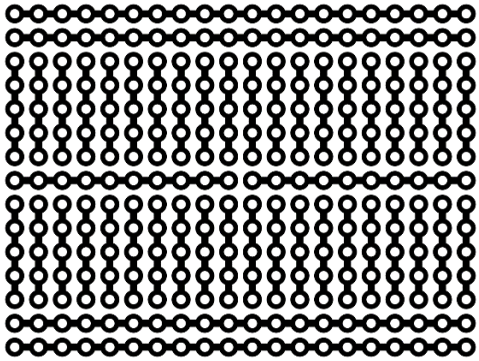


Ilustración 6 – Esquema de placa de prototipado.

7.4.3. MPU6050

El MPU6050 es una Unidad de Medición Inercial (IMU) compuesta por un Procesador Digital de Movimiento (DPM), un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Además, este IMU posee un conversor de señal analógica a digital (ADC) para los valores obtenidos del giroscopio y del acelerómetro. La comunicación se realiza mediante el protocolo de comunicación I^2C lo que facilita la obtención de los datos y la integración con otros dispositivos. Para su correcto funcionamiento necesita una tensión de alimentación entre 2.4V a 3.6V [49]–[52].

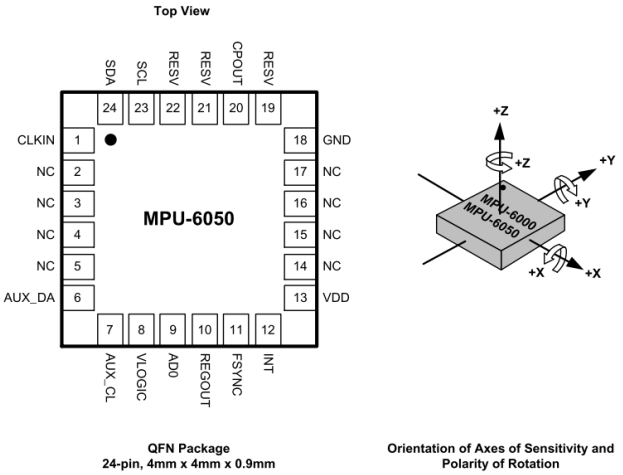


Ilustración 7 – MPU6050.

Esta unidad se puede encontrar embebida en el módulo GY-521, lo que permite que sea alimentada con corrientes de 5 Voltios, además de facilitar su conexión con otros dispositivos.

7.4.4. GPS: NEO 6M

El GPS o *Global Positioning System* es un sistema, como su propio nombre indica, de posicionamiento y navegación global. El sistema está formado por tres elementos o segmentos. En primer lugar, está el segmento del usuario, es decir, los receptores GPS. Estos se pueden embeber en teléfonos móvil, vehículos, naves, personas, etc y son los encargados de recibir la señal GPS. En segundo lugar, están los segmentos de control, utilizados principalmente como apoyo a los satélites. Están compuestos de estaciones de monitorización que se encargan de captar la señal GPS de los satélites y datos atmosféricos, y pasar esta información a la estación máster de control. Esta estación es la encargada de procesar la información, controlar los satélites y de enviar la información GPS a las antenas de tierra, encargadas de comunicarse con los satélites. Por último, están los satélites, el segmento espacial, conformado por 24 satélites situados a 20.200 km de altitud, distribuidos en 6 órbitas equidistantes. En cada una de estas órbitas hay 4 satélites que transmiten la información mediante radiofrecuencia a los usuarios y a los segmentos de control [53].

Para este proyecto se ha utilizado el módulo GPS NEO-6M fabricado por la empresa U-Blox. Puede obtener los datos mediante los protocolos NMEA, UBX binary y RTCM. Además, posee cuatro interfaces de comunicación UART, USB, SPI y DDC (I^2C). Para poder funcionar el módulo necesita una tensión de alimentación entre 2.7V y 3.6V.

Los datos obtenidos utilizando el protocolo NMEA pueden seguir 19 formatos distintos. Para este proyecto se ha utilizado el formato $GPRMC, ya que es uno de los recomendados por su bajo tránsito de datos. Este formato utiliza la secuencia que aparece a continuación:

```
                                                            12
       1         2 3       4 5        6 7   8   9    10  11|
       |         | |       | |        | |   |   |    |   | |
$--RMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,xxxx,x.x,a*hh

 1) Time (UTC)
 2) Status, V = Navigation receiver warning
 3) Latitude
 4) N or S
 5) Longitude
 6) E or W
 7) Speed over ground, knots
 8) Track made good, degrees true
 9) Date, ddmmyy
10) Magnetic Variation, degrees
11) E or W
12) Checksum
```

Ilustración 8 – Formato $GPRMC del protocolo NMEA.

(Fuente: https://www.tronico.fi/OH6NT/docs/NMEA0183.pdf)

Este módulo GPS se puede encontrar embebido en el módulo GY-GPS6M [54], [55] que permite alimentar al receptor GPS con una tensión de 5V y realizar la conexión con otros dispositivos mediante el bus UART. Además, este módulo viene con una antena externa.

7.4.5. LCD 1602

Para que el usuario del dispositivo pueda comprobar los datos que se están tomando, se ha instalado una pantalla LCD (*Liquid Crystal Display*). Concretamente el módulo LCD1602, una pantalla de cristal líquido, que puede representar caracteres alfanuméricos. El LCD tiene dos filas de caracteres, habiendo 16 en cada hilera. Cada uno de estos caracteres está formado por 5 x 7 pixeles [56].

El módulo posee 16 pines que conforman un puerto paralelo. Ocho de estos pines son utilizados para mantener una comunicación bidireccional con otro dispositivo, y el resto de los pines permiten la conexión a una fuente de alimentación, controlar la luz de la pantalla y gestionar las lecturas y escrituras.

7.4.6. ADAPTADOR LCD SERIE A BUS I2C

Debido a que la pantalla LCD utiliza 16 pines para conectarse a la placa *Arduino UNO Rev3*, impidiendo poder añadir otros dispositivos, se optó por utilizar un módulo adaptador del bus paralelo utilizado por el módulo LCD1602 al bus I^2C, logrando con ello que la cantidad de pines consumidos por la pantalla pasen de 16 a 4, contando con los pines de alimentación.

Este adaptador utiliza el controlador I^2C-bus PCF8574 de la empresa Philips Semiconductors y está formado por dos conjuntos de pines. Por un lado, los 16 pines que se conectan directamente al bus paralelo de la pantalla LCD, puesto que tienen el mismo orden y, por otro lado, posee los cuatro pines del bus I^2C, dos para alimentación y otros dos para los datos. Además, dispone también de un potenciómetro que permite variar la luz de fondo de la pantalla [57].

7.4.7. MICROSD Y ADAPTADOR MICROSD

Una *microSD* o *Micro Secure Digital* es un tipo de tarjeta de memoria flash, cuyas dimensiones son 11 mm x 15 mm x 1 mm. Para este proyecto se ha utilizado una tarjeta *microSD* Canvas Select SDHC (High Capacity) de 16 GB de capacidad de la marca Kingston Technology [58], caracterizada por tener el sistema de archivos FAT32 (File Allocation Table) y una velocidad de lectura de 80 MB/s y de escritura de 10 MB/s.

La tarjeta *microSD* se ha utilizado en este proyecto para almacenar toda la información generada por los módulos MPU6050 y NEO-6M, ya que es un sistema de almacenamiento compacto, que permite transportar la información con facilidad y es de bajo coste.

Para poder conectar la tarjeta a la placa *Arduino* es necesario disponer de un módulo adaptador [59] con una clavija para este tipo de tarjeta y un adaptador de voltaje, puesto que las *microSD* trabajan a 3.3V y en este proyecto se usa 5V. Para poder comunicarse con otros dispositivos, este módulo utiliza el bus SPI (*Serial Peripherical Interface*). En la Ilustración 9 y la Ilustración 10 se puede observar el módulo adaptador utilizado.

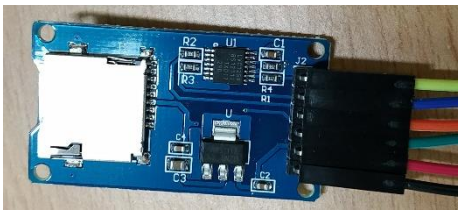


Ilustración 9 – Parte frontal adaptador microSD.

Ilustración 10 – Parte trasera adaptador microSD.

7.4.8. FUENTE DE ALIMENTACIÓN

Para alimentar la placa *Arduino UNO Rev3* y todos los dispositivos conectados a ella, se ha utilizado un adaptador de corriente alterna a corriente continua (AC/DC) de 9 voltios y 1 Amperio. Este se conecta al "*Jack*" de alimentación presente en la placa Arduino.

También existe la posibilidad de alimentar el dispositivo mediante el puerto USB que viene incluido en la placa *Arduino UNO Rev3*. Esto es especialmente útil durante la fase de desarrollo ya que este puerto

permite también la comunicación con un ordenador, por lo que el control del software de la placa y la alimentación se producen por el mismo sitio.

7.4.9. BOTÓN DE APAGADO

Para evitar daños en los componentes del dispositivo se ha implementado un botón que permite realizar un apagado controlado de todos ellos. Esto es especialmente importante para la tarjeta *microSD* puesto que si el archivo en el que se están guardando los datos no se cierra correctamente puede quedar corrupto o, incluso, se puede llegar a dañar la tarjeta físicamente, lo que provocaría que esta quedara inservible.

El botón utilizado está compuesto por cuatro pines que forman el circuito [60] que se observa a continuación:

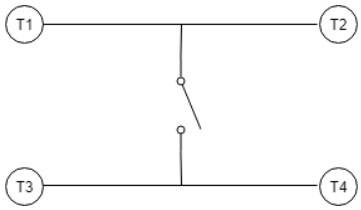


Ilustración 11 – Esquema de un botón.

Como se puede apreciar en la imagen, los terminales 1 y 2 están conectados entre sí, al igual que los terminales 3 y 4. Cuando el pulsador es accionado ambos pares de pines se conectan, permitiendo el paso de la corriente.

Además, este botón se encuentra conectado a la placa Arduino UNO junto con una resistencia. El circuito creado por ambos elementos se conoce como resistencia *pull down* [61]. Cuando el botón está en reposo manda una señal *LOW* (0 voltios), y cuando es pulsado se manda un *HIGH* (5 voltios). Estas señales son detectadas por la placa que actúa en consecuencia.

7.4.10. XIAOMI REDMI NOTE 6 PRO

En el desarrollo de este trabajo ha sido necesario contar con un teléfono móvil para poder realizar la grabación de las pruebas del dispositivo y de la aplicación móvil en un entorno real y para la toma de datos del estado de la calzada utilizando la aplicación desarrollada. También ha sido utilizado durante la creación de esta aplicación, puesto que era necesario testear el código desarrollado.

El teléfono móvil en cuestión es el *Redmi Note 6 Pro* de la marca *Xiaomi*. Cabe destacar, de todo el hardware que viene en un teléfono inteligente, los siguientes elementos utilizados para este trabajo:

- Sensor acelerómetro: BMI120 Bosch
- Cámara de 12 + 5 MP
- Sensor GPS
- Puerto USB 2.0 de tipo Micro-B

Parte de esta información ha sido obtenida utilizando las aplicaciones “Full system info” [18] y “Device Info” [19] que permiten extraer información del sistema con un alto nivel de detalle.

7.5. TECNOLOGÍAS UTILIZADAS

7.5.1. BUS I²C (INTER-INTEGRATED CIRCUIT)

Para realizar la comunicación entre algunos de los elementos del dispositivo, como son la placa Arduino, el adaptador de la pantalla LCD1602 y el módulo GY-521 (acelerómetro), es necesario utilizar el bus I^2C (*Inter-Integrated Circuit*). Se trata de un bus serie, síncrono, que permite varios maestros y esclavos y que fue creado por la empresa Philips Semiconductors en 1982 con el objetivo de simplificar la conexión entre los distintos componentes de sus circuitos [62].

Este bus está diseñado siguiendo el modelo bus maestro-esclavo, es decir, un dispositivo maestro será el encargado de iniciar las transferencias de datos (en nuestro caso la placa Arduino), mientras que los otros dispositivos toman el papel de esclavos y responden a las peticiones del maestro. Cada uno de los elementos esclavos conectados al bus deben tener una dirección única formada por 7 bits [63].

Un aspecto positivo de este protocolo es que solo necesita dos señales bidireccionales para transmitir los datos entre los distintos dispositivos conectados al bus. Una es la señal SDA o *Serial Data Line*, por la que se envían y se reciben los datos. Y la otra es la señal SCL o *Serial Clock Line*, por la que se transmite la señal de reloj generada por el maestro.

7.5.2. UART

UART o *Universal Asynchronous Receiver-Transmitter* es un dispositivo para realizar comunicaciones en serie de manera asíncrona [64]. El circuito UART hace de intermediario entre una interfaz paralela y una en serie, siendo esta última la utilizada para la comunicación con otros dispositivos. Además, este dispositivo puede ser programado para controlar la velocidad, la paridad, la longitud y los bits de parada. En la imagen que aparece a continuación se puede observar un esquema de un UART [65]:

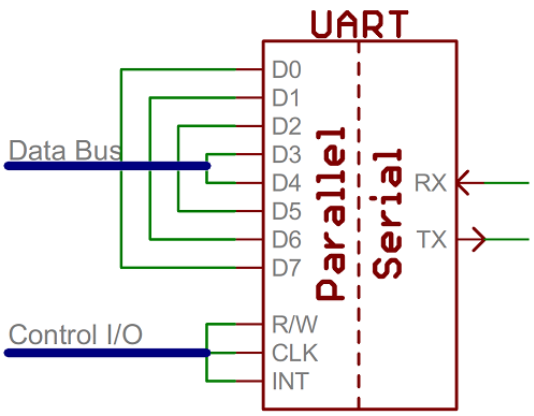


Ilustración 12 – Esquema UART.

Para llevar a cabo la transmisión de los datos entre dos dispositivos es necesario que ambos dispongan de un microcontrolador con el circuito UART integrado. Esta transmisión se realiza directamente, sin necesidad de una señal de reloj, mediante dos pines. Uno es el pin Tx, que es el encargado de enviar los datos al otro dispositivo. Y el otro pin es el Rx, encargado de recibir la información. Como es lógico, la conexión entre dos microcontroladores con UART se realiza conectando el pin Tx de uno con el pin Rx del otro dispositivo y viceversa.

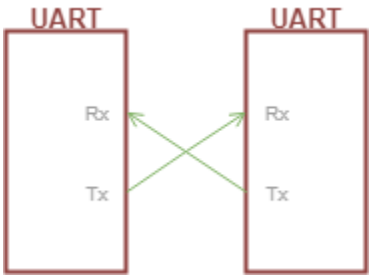


Ilustración 13 – Esquema de conexión entre UARTs.

7.5.3. SPI

SPI o *Serial Peripheral Interface* es una interfaz de comunicación serial y síncrona [66], desarrollada por la compañía Motorola a mediados de la década de los 80. Usa una arquitectura maestro-esclavo con un único maestro y múltiples esclavos. Este es el encargado de controlar las escrituras y lecturas realizadas en el bus. Como se puede observar en la imagen [67] que aparece a continuación, la interfaz utiliza cuatro señales lógicas:

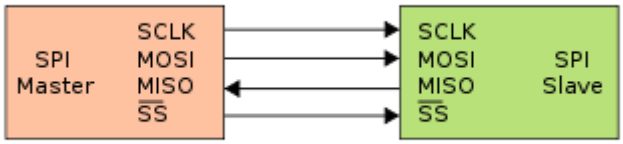


Ilustración 14 – Esquema de conexión entre dos interfaces SPI.

Las señales MOSI (*Master Output Slave Input*) y MISO (*Mater Input Slave Output*) son las encargas de transmitir los datos entre los dispositivos. Mediante la señal MOSI el maestro envía los datos al esclavo, mientras que con la señal MISO ocurre lo contrario, el esclavo envía los datos al maestro. Esto está sincronizado gracias a la señal de reloj generada por el maestro y transmitida al esclavo con la señal SCLK. Para que todo lo anterior ocurra, se tiene que indicar primero al esclavo, la señal SS (*Slave Selector*), que tiene que conectarse al bus. Esta señal también es utilizada cuando existen varios esclavos conectados al bus.

7.5.4. NODE.JS

Node.js es un entorno de ejecución de JavaScript basado en eventos, multiplataforma, asíncrono, de código abierto y altamente escalable. Su autor es Ryan Dahl y está desarrollado con C, C++ y JavaScript. Usa el motor V8 de Google para compilar el código fuente de JavaScript, logrando con ello un aumento de la velocidad de ejecución [68].

Este entorno permite la creación de aplicaciones webs usando *JavaScript* y diversos módulos que proveen de múltiples funcionalidades, como pueden ser gestor de sistemas de archivos, control de protocolos de red como DNS, HTTP o TCP, funciones criptográficas, etc.

7.5.5. ANDROID ESTUDIO

Para el desarrollo de aplicaciones para Android es necesario disponer del entorno de desarrollo integrado (IDE) Android Studio, basado en IntelliJ IDEA. Este IDE fue desarrollado usando el lenguaje de programación Java y está disponible para varias plataformas. El desarrollo de esta herramienta está a cargo de la empresa Google, que lanzó la primera versión en diciembre del 2014 [69], [70].

7.5.6. JAVA DEVELOPMENT KIT 8

Java Development Kit o Java JDK es un paquete de herramientas software multiplataforma para el desarrollo de programas en el lenguaje de programación Java. Ha sido desarrollado por la compañía Oracle Corporation y fue publicado en noviembre del 2006 [71].

7.5.7. JSON

JSON o *JavaScript Object Notation* es un formato ligero para el intercambio de datos. Se caracteriza principalmente por la facilidad para los seres humanos y ordenadores de leer y escribir información siguiendo este formato. Los datos pueden estar estructurados de dos formas. Por un lado, pueden estar agrupados en parejas clave/valor o en listas ordenadas. A continuación, se puede observar un ejemplo:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Fuente: https://json.org/example.html

7.5.8. LENGUAJE DE PROGRAMACIÓN DE ARDUINO

Para poder programar la placa *Arduino UNO Rev3* se ha utilizado el *IDE* de Arduino junto con el lenguaje propio de Arduino. Este lenguaje está basado en C y C++. Los programas escritos con el lenguaje de Arduino se denominan "Sketch" y se caracterizan principalmente por estar compuestos de las funciones "`setup()`" y "`loop()`". La primera es llamada una vez al comienzo de la ejecución del programa. En esta función se deben realizar las inicializaciones de librerías y la configuración de pines. La segunda función, como su propio nombre indica, es llamada en bucle. En ella se realizan las tareas más importantes [72].

7.5.9. IONIC FRAMEWORK

IONIC Framework es una herramienta de código abierto para la creación de aplicaciones móviles y de escritorio usando tecnologías webs como son HTML, CSS y JavaScript. Se caracteriza por su potencial en la creación de interfaces de usuario. Además, se puede integrar con otros marcos de trabajo como puede ser Angular [73].

7.5.9.1. IONIC NATIVE C.E.

Ionic Native Community Editon es un conjunto de librerías de Cordova que permiten añadir nuevas funcionalidades a las aplicaciones desarrolladas con el IONIC Framework. Estas librerías son creadas y mantenidas por la comunidad de IONIC [74].

Las librerías utilizadas en este proyecto son las siguientes:

- *File:* Permite la lectura y escritura de los archivos que se encuentran el dispositivo.
- *FileOpener:* Permite abrir un archivo desde la aplicación móvil usando la aplicación por defecto de lectura de documentos.
- *Google Maps:* Permite el uso de la API de Google Maps en la aplicación.
- *Sensor:* Permite el uso de los sensores del teléfono móvil.

7.5.10. APACHE CORDOVA

Apache Cordova es un marco de trabajo de código abierto para el desarrollo de aplicaciones para móvil. Está compuesto por un conjunto de APIs que permiten el acceso a las funciones nativas del teléfono móvil, como el acelerómetro o la cámara, usando JavaScript [75], [76].

7.5.11. ANGULAR

Angular es un marco de trabajo de código abierto para el desarrollo de aplicaciones webs para móviles y para escritorio usando los lenguajes de programación *TypeScript* y *JavaScript*. Es desarrollado por la compañía Google, y fue lanzado en el año 2016 [77].

Entre las características principales de Angular se encuentra el uso del patrón de diseño *Modelo Vista Controlador* (MVC) y el cuadro de órdenes "*Angular CLI*", con el que se puede añadir código, realizar pruebas y previsualizar la aplicación [78], [79].

8. DESARROLLO

Para dar solución al problema planteado se ha desarrollado un dispositivo capaz de detectar la vibración y la posición en la que se encuentra. Estos datos se obtienen gracias a un acelerómetro y un GPS, son almacenados en una microSD y mostrados en una pantalla LCD. Todo esto se controla mediante una placa Arduino UNO Rev3.

Para poder valorar los datos obtenidos, se ha desarrollado también una aplicación móvil para Android que tiene las mismas funciones que el dispositivo y, además, muestra los datos en una gráfica. El objetivo de este segundo desarrollo es poder comparar la información recogida por ambos sistemas y así determinar cuál podría ser una solución adecuada al problema.

A continuación, se describe en mayor detalle ambos desarrollos.

8.1. ANÁLISIS

En la fase de análisis se han realizado tres pasos [80] que nos permiten tener una visión global del problema que queremos resolver y de las herramientas que debemos desarrollar para lograr una solución.

El primer paso es el de identificar el problema al que nos enfrentamos. Como se mencionó en el apartado de introducción, el estado de las carreteras es algo que nos afecta a todos y por ello se debe realizar un control periódico. Existen múltiples formas de realizar este control, pero la gran mayoría son de un alto coste económico o no se centran en el deterioro del asfalto sino en detectar la temperatura, la cantidad de hielo o nieve, etc. Es por ello por lo que se requiere de un dispositivo que sea capaz de detectar el estado de la calzada, utilizando componentes de bajo coste.

El segundo paso que se debe realizar es el de definir como son las características y requisitos de este dispositivo. Principalmente, el dispositivo debe ser capaz de detectar el estado de la calzada y de proveer de esa información al usuario. Además, esta información debe ser comprensible por otros sistemas, es decir, se debe utilizar un estándar. Físicamente el dispositivo debe ser de pequeño tamaño y portátil, además, los materiales utilizados deben ser económicos y los componentes software deben ser de código abierto. En el apartado estético no se hace hincapié puesto que se trata de un prototipo. Sin embargo, se debe tener en cuenta que la colocación de los elementos hardware debe ser minimalista, evitando enredos de cables. Por último, el dispositivo debe ser seguro en todo momento para el usuario y no debe contaminar el medio ambiente.

El tercer y último paso de la fase de análisis, es el de fraccionar el problema para poder abordarlo mejor. Inevitablemente, el proyecto se subdivide en la parte hardware y la parte software.

Por un lado, la parte hardware se divide a su vez en el análisis, diseño e implementación de cada uno de los elementos que componen al dispositivo. Mientras que la parte software se divide en el desarrollo de un script inicial y en el de una librería para cada componente. Estas divisiones permiten abordar mejor el problema y, además, permite que el proyecto sea modular, pudiendo cambiar o modificar cualquiera de los elementos sin que el resto se vea afectado.

Tras finalizar el análisis del dispositivo se identificó una nueva problemática. Era necesario disponer de otro dispositivo con las mismas funcionalidades con el que comparar los datos obtenidos. Este nuevo dispositivo debía cumplir con las mismas características definidas anteriormente. Por ello, se decidió utilizar un teléfono móvil, ya que posee un acelerómetro y un GPS, y es capaz de guardar la información en su memoria interna. Este segundo desarrollo se dividió, al igual que con el dispositivo, en los distintos sensores y elementos utilizados, con la diferencia de que no es necesario entrar en el apartado hardware.

Profundizando en el segundo paso descrito anteriormente, en el que se habla de las características y requisitos, se plantean una seria de cuestiones que establecen una base sobre la que desarrollar este trabajo.

Al comienzo de este proyecto nos enfrentamos a una gran incógnita que, una vez resuelta, nos sirve de guía en el desarrollo de este trabajo. La pregunta en cuestión es *¿Cómo podemos **detectar** el estado de la calzada?* Existen varios indicadores que nos son útiles para determinarlo:

- Indicadores visuales: Una carretera está en mal estado si en la superficie se pueden observar grietas, hundimientos, baches, cúmulos de agua, etc.
- Indicadores físicos: Si al circular por una carretera notamos que la carrocería y los ocupantes del vehículo vibran notablemente, es una señal de que la calzada está en mal estado. Lo mismo ocurre con la dirección del vehículo. Si esta realiza movimientos bruscos, pueden estar producidos por irregularidades en la vía.
- Indicador temporal: Otra forma de averiguar cómo está la calzada es teniendo en cuenta el tiempo que lleva sin ser revisada y sin recibir mantenimiento. El paso del tiempo permite que los materiales se degraden y aparezcan desperfectos en la carretera.
- Indicador de uso: Se puede intuir como está el estado de una carretera en función del uso que se le dé y el tipo de vehículos que circulen sobre ella. No es lo mismo que por una carretera en un día circulen 1000 turismos que 1000 camiones. El paso de los camiones somete al asfalto a situaciones de mayor estrés provocando un mayor deterioro.

Tras valorar estos posibles indicadores, se llega a la conclusión de que la mejor forma de determinar el estado de la calzada, teniendo en cuenta las limitaciones económicas y materiales de este proyecto, es utilizando la vibración del vehículo. Los otros indicadores quedan descartados, puesto que la información del uso y del mantenimiento no son accesibles fácilmente y el uso de los indicadores visuales implica la utilización de cámaras y/o láseres con un coste económico alto. Además, los indicadores visuales requieren de una alta capacidad de cómputo y la complejidad del proyecto aumentaría ya que es probable que se requiera de la utilización de inteligencia artificial.

Por tanto, la respuesta a la pregunta, de cómo detectar el estado de la calzada, sería utilizando un indicador físico ya que son sencillos de tratar y económicos. Tras concluir esto, lo siguiente es aclarar cuál es el indicador físico ideal. Por un lado, está la vibración del vehículo con la que se puede determinar cuándo se pasa por encima de un bache. Esta vibración puede ser captada mediante un sensor de vibración o un acelerómetro. Por otro lado, existe la posibilidad de capturar movimientos bruscos en la dirección del vehículo. Sin embargo, esta opción tiene el problema añadido de que estos movimientos pueden ser provocados por el conductor del vehículo y no por una irregularidad en la calzada, lo que llevaría a tener datos inexactos.

Finalmente, se decide utilizar un acelerómetro para detectar las vibraciones del vehículo, en concreto el acelerómetro MPU6050 descrito en el punto 7.3.3. de esta memoria. Este sensor permite medir los cambios de aceleración en los ejes, por lo que se puede determinar cuándo se pasa por un bache. La otra opción es utilizar un sensor de vibración como el que aparece a continuación (Ilustración 15 e Ilustración 16) [81]. Sin embargo, este sensor no es útil puesto que solo proporciona dos estados, es decir, solo indica si hay o no vibración. Esto es un problema ya que cualquier situación, incluso subir una pendiente, activaría el sensor. Además, este sensor tampoco indica la intensidad de la vibración.

Ilustración 15 – Sensores de vibración SW-520D

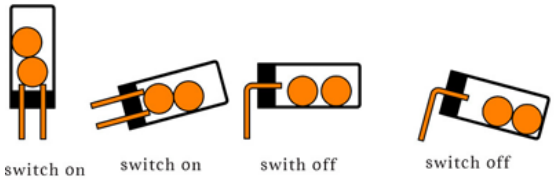


Ilustración 16 – Esquema del sensor de vibración SW-520D

Tras resolver la primera cuestión surge la siguiente pregunta: *¿Cómo podemos saber **dónde** se ha detectado un bache en concreto?*

Si solo se almacena la información del sensor acelerómetro surge el problema de que no se puede saber dónde ha ocurrido un bache en concreto, por lo que se tiene que recurrir a un método adicional que permita obtener dicha información.

- Una opción puede ser definir una ruta por la que circular con el vehículo, sabiendo así que si se detecta un bache con el acelerómetro estará obligatoriamente en ese trayecto. Además, si el bache se produce al principio de la toma de datos, significará que este está al comienzo del trayecto. Lo mismo ocurre si está a mitad o a final del archivo de datos. No obstante, este método es muy poco preciso y existen muchos factores que eliminarían esta relación entre el archivo de datos y la ruta establecida.
- Otra opción es recurrir a un sensor GPS, como el explicado en el punto 7.3.4., logrando con ello obtener la latitud, la longitud y la altitud. Si la consulta al sensor GPS se realiza en cada toma de datos se puede posicionar con mucha presión cada uno de los baches detectados.

Como es de suponer, la opción elegida para posicionar las tomas de datos es el sensor GPS. La información generada por el dispositivo se almacena, junto con los datos del acelerómetro, en el archivo de datos.

Aunque no es totalmente necesario, almacenar el momento temporal en el que se realiza la toma de datos es importante. La solución a esta nueva cuestión es relativamente sencilla puesto que el dispositivo GPS permite obtener la fecha y hora de su sistema. Esta marca temporal, al igual que la información anterior, se almacena en el archivo de datos.

Otra duda que surge durante el análisis del problema es *¿dónde podemos **almacenar la información** generada por los sensores?* Como se ha podido observar en los párrafos anteriores, la información se almacena en un archivo de datos. Sin embargo, no se especifica el soporte tecnológico utilizado para guardar esta información. A continuación, se listan las opciones que se han tenido en cuenta:

- Conectar el dispositivo mediante una antena wifi y mandar la información a un servidor externo. Este servidor podría encargarse también de su tratamiento y visualización de los datos.
- Conectar el dispositivo a un teléfono inteligente o un ordenador portátil utilizando un sensor bluetooth y mandar la información a través de él. Además, se puede crear una aplicación que reciba los datos y los trate.
- Guardar la información en una tarjeta SD o microSD. Es una forma sencilla de guardar información sin tener que depender de otros sistemas. Además, usando una tarjeta microSD se puede pasar la información a otros dispositivos que tengan un puerto para tarjetas, como teléfonos móviles y ordenadores (utilizando un adaptador).

De estas tres opciones se elige la tarjeta microSD, la cual se explica en el apartado 7.3.7.

Sabiendo como detectar el estado de la calzada, como posicionar los baches encontrados y donde almacenar la información, queda quizás la pregunta más importante *¿cómo **controlamos** todos los componentes hardware?* Existen multitud de placas con microcontroladores que realizarían perfectamente esta función. Para este proyecto se utilizó la placa Arduino UNO Rev3, descrita en el apartado 7.3.1., dado que ya se había trabajado con una igual en proyectos anteriores y no era necesario realizar un estudio del funcionamiento de la placa.

Tras resolver estas preguntas, las siguientes cuestiones que surgen tratan temas menores. Una de estas dudas es la siguiente: *¿Cuál es el **intervalo de tiempo** que se debe dejar entre cada toma de datos?*

Suponiendo que el vehículo en el que se está realizando la toma de datos va a una velocidad media de 40 kilometro/hora, en un solo segundo recorrerá 11'1 metros. Esto implica que el tiempo entre cada consulta a los sensores debe ser el menor posible, dado que puede que el dispositivo no detecte algún bache. Sin embargo, esto no siempre es factible debido a que los componentes hardware tienen sus propias limitaciones. Un ejemplo de ello es el sensor GPS que depende de la conexión con los satélites. Por esa razón, el intervalo de tiempo variará en función de las características del hardware utilizado.

Finalizado el análisis del dispositivo, solamente queda abarcar los interrogantes relacionados con el usuario y con las pruebas de funcionamiento. En primer lugar, surge la pregunta de *¿cómo podemos indicar al usuario que el dispositivo está **funcionando correctamente**?*

Se puede indicar al usuario que los componentes y el software se ha inicializado correctamente y que está funcionando según lo previsto mediante alguna de las siguientes opciones:

- Usando diodos leds: Se puede colocar un led por componente e indicar con él si funciona correctamente o no. Con esto se consigue saber el estado de cada dispositivo de manera sencilla y visual. El problema es que la cantidad de información o mensajes es limitada.
- Usando una pantalla LCD: Permite mostrar más información de los componentes y de los datos que se están tomando. El principal problema es su complejidad y otros aspectos relacionados con el hardware, como son problemas de alimentación (ya que consume mucha energía), problemas de espacio y de número de pines utilizados (posee un bus serie y requiere de 16 pines para realizar la conexión).
- Usando un zumbador (sonido): Se puede crear un sistema de sonidos y pitidos que informe al usuario de lo que está ocurriendo en el dispositivo. Por ejemplo, un pitido corto es que el proceso de inicio se ha completado correctamente, un pitido intermitente es que algo no ha ido bien, un pitido largo es que la toma de datos ha terminado y el dispositivo se va a apagar.

Finalmente, aunque la complejidad de su instalación y de su configuración es mayor, se decide utilizar la pantalla LCD (apartado 7.3.5.) dado que permite mostrar información al usuario sobre el estado de los componentes y, durante la toma de datos, mostrar la información de los sensores en tiempo real.

Para solucionar el problema del número de pines se optó por utilizar un adaptador (apartado 7.3.6.) que transforma el bus serie de 16 pines al bus I2C con solo cuatro pines. Esto elimina uno de los principales problemas de esta opción.

Este apartado se termina con una última cuestión *¿cómo se puede **demostrar** que el dispositivo funciona correctamente?* Al completar todo el análisis surge la problemática de como demostrar que los datos que se encuentran en el archivo de datos, tomados desde un vehículo en circulación y que almacenan los ejes del acelerómetro y la posición GPS, son reales y correctos. Existen varias formas de conseguir esto:

- Se puede recurrir a una tercera persona ajena al proyecto que verifique que los datos tomados son reales y correctos. Para ello sería necesario que la persona estuviera presente en una toma de datos real.
- Se puede utilizar una grabación, realizada a la vez que la toma de datos, en la que se muestre cual es el trayecto que el vehículo realiza. Este formato permite no tener que depender de una tercera persona. Además, sobre esa grabación se puede mostrar una gráfica con los datos tomados, comparando de manera muy visual lo que se observa en el video con los datos tomados.
- Se puede tomar fotografías de la calzada en los puntos críticos, es decir, aquellos puntos en los que hay un gran cúmulo de baches. Estas fotografías, junto con una posición GPS, pueden usarse para contrastar los datos del archivo de datos (si en el archivo de datos aparece un bache en una posición 'X', entonces debería haber una foto de la posición 'X' en la que se muestre ese bache).

Tras valorar las opciones se ha optado por realizar una grabación ya que es una forma válida y sencilla de demostrar el buen funcionamiento del dispositivo. Además, el vídeo puede ser visionado en cualquier plataforma y, junto con un gráfico, es una forma completa de demostrar cómo trabaja el aparato y cuáles son los resultados que obtiene. Esto se puede complementar con otra de las opciones valoradas, la toma de fotografías.

También surge la necesidad, desde el punto de vista del desarrollo del proyecto, de poder comparar la información obtenida con otro sistema que realice las mismas funciones. Con esto se pretende poder corroborar los datos y poder determinar que solución es la idónea para el problema que se plantea en este trabajo. A continuación, se mencionan algunos de los sistemas valorados:

- Acelerómetro de gran precisión: Se realizarían tomas de datos junto con el dispositivo desarrollado y se compararían los datos a posteriori. El principal problema de esta opción es el elevado precio.
- Aplicaciones para dispositivos móviles ya publicadas: Tras un análisis de las aplicaciones disponibles se llegó a la conclusión de que ninguna cumple al 100% con las funcionalidades del dispositivo, lo que complica la comparación entre ambos.
- Crear una aplicación utilizando un framework: Se trata de un desarrollo complejo, pero con el que se logrará un producto que se adapte totalmente al problema.

Aunque de las tres opciones es la más compleja, se decide crear una aplicación para dispositivos móviles con sistema operativo Android utilizando el framework IONIC (explicado en el punto 7.4.9.). Esta aplicación realiza las mismas funciones que el dispositivo desarrollado con Arduino.

En el apartado que viene a continuación se explica cuáles son los requisitos funcionales y no funcionales del dispositivo y de la aplicación para Android. Este punto es una parte fundamental de la fase de análisis debido a que describe el comportamiento de ambos sistemas, permite acotar el trabajo y sirve de guía durante todo el desarrollo.

8.1.1. REQUISITOS

En este apartado se enumeran todos los requisitos, tanto funcionales como no funcionales, presentes en este proyecto [82], [83]. Pero antes de continuar es importante responder a las siguientes cuestiones:

- *¿Qué es un requisito funcional?*

  Un requisito funcional permite describir los servicios y funciones que se esperan del sistema, como debe su comportamiento ante ciertas situaciones y como debe ser su interacción con el entorno.

- *¿Qué es un requisito no funcional?*

  Los requisitos no funcionales son restricciones, cualidades y atributos de las funciones y servicios del sistema.

Este proyecto se compone de dos desarrollos paralelos que poseen requisitos distintos. Por un lado, está el dispositivo Arduino y por otro la aplicación para móvil. Es por ello por lo que a continuación hay dos listas de requisitos funcionales y dos listas de requisitos no funcionales.

El *dispositivo Arduino* tiene los siguientes requisitos:

REQUISITOS FUNCIONALES:

- El dispositivo detectará la vibración del vehículo.
- El dispositivo detectará la posición en la que se encuentra. Para ello hará uso de un sensor GPS.
- El dispositivo almacenará la información generada en un sistema de almacenamiento que permita su portabilidad a otros dispositivos.
- El dispositivo generará un archivo de datos con la información generada por los sensores.
- El dispositivo mostrará al usuario la información capturada por los sensores en tiempo real durante su funcionamiento.
- El dispositivo podrá ser apagado por el usuario de manera controlada, evitando la pérdida de información.
- El dispositivo mostrará un mensaje de bienvenida y de despedida al usuario.

REQUISITOS NO FUNCIONALES:

- El tamaño del dispositivo no puede superar los 15 centímetros en ninguna de sus dimensiones.
- El dispositivo debe poder apagarse en cualquier momento.
- El dispositivo debe mantener la integridad de los datos.
- El dispositivo usará un estándar para estructurar la información generada.
- El código del dispositivo debe estar estructurado por librerías.
- El código del dispositivo debe desarrollarse en el lenguaje de Arduino y en C.
- El código del dispositivo debe tener un tamaño inferior a 23 Kilobytes.
- El dispositivo debe ser seguro para el usuario.
- El dispositivo no debe contaminar el medio ambiente.
- El dispositivo debe ser modular, permitiendo el cambio de un componente sin afectar al resto.
- El coste total de todos los componentes del dispositivo no debe superar los cien euros.

La *aplicación para Android* tiene los siguientes requisitos:

REQUISITOS FUNCIONALES:

- La aplicación debe guardar la información generada por los sensores en un archivo.
- La aplicación detectará la vibración del vehículo usando los sensores del teléfono móvil.
- La aplicación detectará la posición en la que se encuentra usando el sensor GPS del teléfono móvil.
- La aplicación mostrará en tiempo real la información que obtiene de los sensores.
- La aplicación guardará la información en la memoria interna del teléfono móvil.
- La aplicación generará un archivo por cada toma de datos y le dará como nombre la fecha y hora siguiendo el formato Unix Epoch.
- La aplicación listará todas las tomas de datos, mostrando en cada elemento de la lista: la fecha, la hora y el tamaño del archivo.

- Se podrá abrir un archivo de datos desde la lista de archivos dentro de la propia aplicación.
- Se podrá visualizar una gráfica en la que se muestre la información de un archivo de datos.
- La aplicación mostrará un mensaje de error en caso de que el sensor GPS no se encuentre encendido.
- La aplicación mostrará un mensaje al usuario informándole sobre sus derechos con relación al tratamiento de datos de carácter personal.

REQUISITOS NO FUNCIONALES

- La aplicación no tardará más de 10 segundos en listar todos los archivos de datos
- La aplicación deberá tener el logotipo de la Universidad de Las Palmas de Gran Canaria y el Instituto de Ciencias y Tecnologías Cibernéticas.
- La aplicación no tardará más de 5 segundos en iniciarse.
- La toma de datos no tendrá un límite de duración.
- La información generada en la toma de datos deberá ser almacenada siguiendo un estándar.

8.1.2. ACTORES

En el campo de la Ingeniería del Software, un actor es algo o alguien que interactúa con el sistema, pero que es externo a él. Este actor es capaz de enviar o recibir mensajes e intercambiar información con dicho sistema. Además, se debe diferenciar entre dos tipos de actores. Por un lado, están los actores principales cuyos objetivos se satisfacen al utilizar el sistema. Por otro lado, están los actores secundarios, encargados principalmente de proporcionar un servicio al sistema [84], [85].

En este proyecto, los actores son los mismos para el dispositivo como para la aplicación. A continuación, se hará una descripción de todos ellos, teniendo en cuenta que un actor es todo aquello que interactúe con el sistema, ya sea una persona, un dispositivo hardware u otro sistema.

- *Usuario* (actor principal). Su función principal es la de controlar el dispositivo para tomar datos del estado de la calzada.
    - En el dispositivo puede:
        - Encender y apagar el dispositivo
        - Consultar la información en la pantalla LCD
        - Insertar o extraer la tarjeta de memoria
    - En la aplicación puede:
        - Ver la lista de los archivos de tomas de datos
        - Abrir un archivo
        - Ver el gráfico de una toma de datos
        - Comenzar una toma de datos
        - Finalizar una toma de datos
- *Sistema GPS* (actor secundario). El sensor GPS utilizado tanto en la aplicación móvil como en el dispositivo interactúan con la infraestructura del sistema GPS para poder obtener la posición en la que se encuentran.
- *Sensor acelerómetro* (actor secundario). El acelerómetro detecta los cambios de aceleración en los ejes del dispositivo y del teléfono móvil y los transmite al sistema.

8.1.3. CASOS DE USO

La técnica de los casos de uso fue creada por Ivar Jacobson. Esta técnica tiene como objetivo especificar el comportamiento del sistema desde el punto de vista del usuario. Además, define como interactúa el sistema con el entorno [84], [85]. A continuación, se muestran los casos de uso del dispositivo Arduino (Ilustración 17) y de la aplicación para Android (Ilustración 18):

En el dispositivo Arduino interviene, por un lado, el usuario (que sería un actor principal) que tiene como objetivo principal tomar datos del estado de la calzada. Por otro lado, el sensor GPS y el sensor acelerómetro (que serían actores secundarios o pasivos) tienen como meta alimentar al sistema con información.

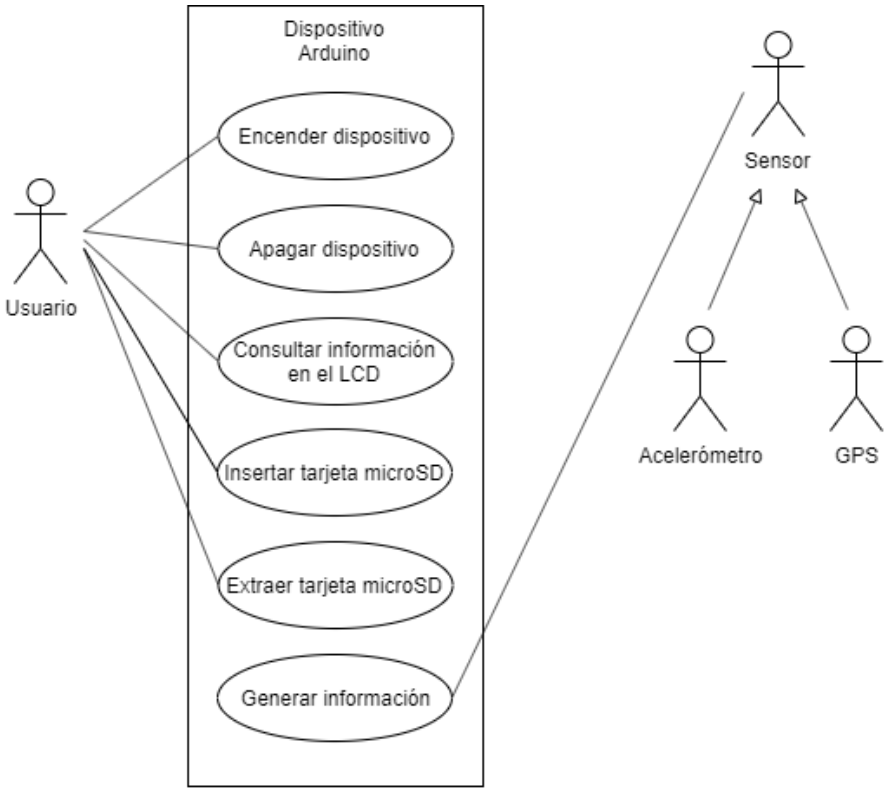


Ilustración 17 – Casos de uso del dispositivo Arduino.

8.1.3.1. CASOS DE USO DEL DISPOSITIVO ARDUINO

NOMBRE	Encender el dispositivo.
DESCRIPCIÓN	El usuario enciende el dispositivo alimentándolo con corriente eléctrica.
ACTORES	Usuario
PRECONDICIONES	El usuario debe disponer de un transformador AC/DC de 9 Voltios y una toma de corriente o de un cable USB (de tipo A en un extremo y de tipo B en el otro) y un ordenador.
POSTCONDICIONES	El dispositivo se encenderá y mostrará durante un segundo un mensaje de bienvenida en la pantalla LCD. Tras ello el dispositivo inicializará todos los componentes y comenzará la toma de datos.
FLUJO	1. El usuario proporciona corriente eléctrica al dispositivo utilizando:

	1.1. Un transformador AC/DC y conectándolo al Jack de alimentación de la placa Arduino UNO Rev3. 1.2. Un cable USB de tipo A/B y conectándolo al puerto USB tipo B de la placa Arduino UNO Rev3. 2. El dispositivo se enciende y muestra un mensaje de bienvenida en la pantalla LCD.
FLUJO ALTERNATIVO	1. El usuario alimenta el dispositivo con un voltaje superior a los 12 voltios. 2. La placa Arduino UNO Rev3 o alguno de sus componentes pueden sufrir daños.

NOMBRE	Apagar el dispositivo.
DESCRIPCIÓN	El usuario apaga el dispositivo.
ACTORES	Usuario
PRECONDICIONES	El dispositivo debe estar encendido.
POSTCONDICIONES	El dispositivo se apaga y se puede extraer la tarjeta microSD.
FLUJO	1. El usuario pulsa el botón de apagado que se encuentra en tabla de prototipado. 2. El dispositivo realiza las tareas de apagado. 3. El dispositivo muestra en la pantalla LCD un mensaje de apagado. 4. El usuario puede desconectar el dispositivo de la corriente eléctrica.
FLUJO ALTERNATIVO	1. El usuario desconecta el dispositivo de la corriente sin haber presionado previamente el botón de apagado. 2. La placa Arduino UNO Rev3, los componentes, la tarjeta microSD y/o los datos almacenados pueden sufrir daños.

NOMBRE	Consultar la información en la pantalla LCD.
DESCRIPCIÓN	El usuario consulta la información de los sensores en tiempo real gracias a la pantalla LCD. Si existe algún problema en el dispositivo este se verá reflejado en la pantalla LCD.
ACTORES	Usuario
PRECONDICIONES	El dispositivo debe estar encendido.
POSTCONDICIONES	El usuario obtiene información del dispositivo.
FLUJO	1. El usuario observa la pantalla LCD y comprueba el funcionamiento del dispositivo. 1.1. Si se muestran datos coherentes es un buen indicador del correcto funcionamiento del dispositivo. 1.2. Si no se muestran datos, se muestran caracteres aleatorios o la intensidad lumínica de la pantalla varía, es símbolo de un posible problema. 2. El usuario actúa en consecuencia de lo que ve en la pantalla LCD.
FLUJO ALTERNATIVO	

NOMBRE	Insertar la tarjeta microSD.
DESCRIPCIÓN	El usuario inserta la tarjeta microSD en el adaptador para tarjetas.

ACTORES	Usuario.
PRECONDICIONES	• El usuario debe estar en posesión de una tarjeta microSD. • El dispositivo debe estar apagado. • La tarjeta microSD debe estar formateada en el sistema de archivos FAT16 o FAT32.
POSTCONDICIONES	La tarjeta microSD está conectada al dispositivo y lista para almacenar los datos de los sensores.
FLUJO	1. El usuario comprueba que el dispositivo está apagado y que está en posesión de una tarjeta microSD. 2. El usuario conecta la tarjeta microSD en el adaptador situado en la tabla de prototipado. 3. El usuario puede encender el dispositivo y hacer uso de él.
FLUJO ALTERNATIVO	1. El usuario conecta la tarjeta microSD mientras el dispositivo está encendido. 2. El adaptador, la tarjeta microSD y/o el contenido de esta pueden sufrir daños.

NOMBRE	Extraer la tarjeta microSD
DESCRIPCIÓN	El usuario extrae la tarjeta microSD del dispositivo.
ACTORES	Usuario
PRECONDICIONES	El dispositivo está apagado y la tarjeta se encuentra alojada en el adaptador para tarjetas.
POSTCONDICIONES	La tarjeta microSD se encuentra desconectada fuera del dispositivo, lista para ser utilizada en otro sistema.
FLUJO	1. El usuario comprueba que el dispositivo está apagado y que la tarjeta se encuentra alojada en el adaptador para tarjetas. 2. El usuario extraer la tarjeta microSD. 3. El usuario puede utilizar la tarjeta en otro sistema.
FLUJO ALTERNATIVO	1. El usuario desconecta la tarjeta microSD mientras el dispositivo está encendido. 2. El adaptador, la tarjeta microSD y/o el contenido de esta pueden sufrir daños.

NOMBRE	Generar información
DESCRIPCIÓN	El sensor acelerómetro y el sensor GPS proporcionan información al sistema.
ACTORES	Sensor acelerómetro y sistema GPS.
PRECONDICIONES	El dispositivo debe estar encendido y el sensor GPS debe haber realizado la conexión con el sistema GPS.
POSTCONDICIONES	La información proporcionada se muestra en la pantalla LCD y se almacena en la tarjeta microSD dentro de un archivo.
FLUJO	1. El sistema se enciende y ambos sensores realizan las tareas de inicialización. 2. El sensor GPS se conecta con el sistema GPS 3. Ambos sensores vuelcan la información obtenida en el dispositivo

	3.1. En la pantalla LCD se muestra en tiempo real información de ambos sensores. 3.2. En la tarjeta microSD se almacena la información de ambos sensores.
FLUJO ALTERNATIVO	1. El sensor GPS no logra establecer la conexión con el sistema GPS 2. El sensor vuelca en el sistema datos con valor 0. 3. En la pantalla LCD y en la tarjeta microSD se observa que el valor de las variables del GPS es igual a cero.

8.1.3.2. CASOS DE USO DE LA APLICACIÓN ANDROID

Por otra parte, aunque la aplicación para Android tiene el mismo objetivo que el dispositivo Arduino, posee más funcionalidades relacionadas con el tratamiento de los datos obtenidos. A continuación, se puede observar el diagrama de los casos de usos y las tablas de cada uno de ellos:

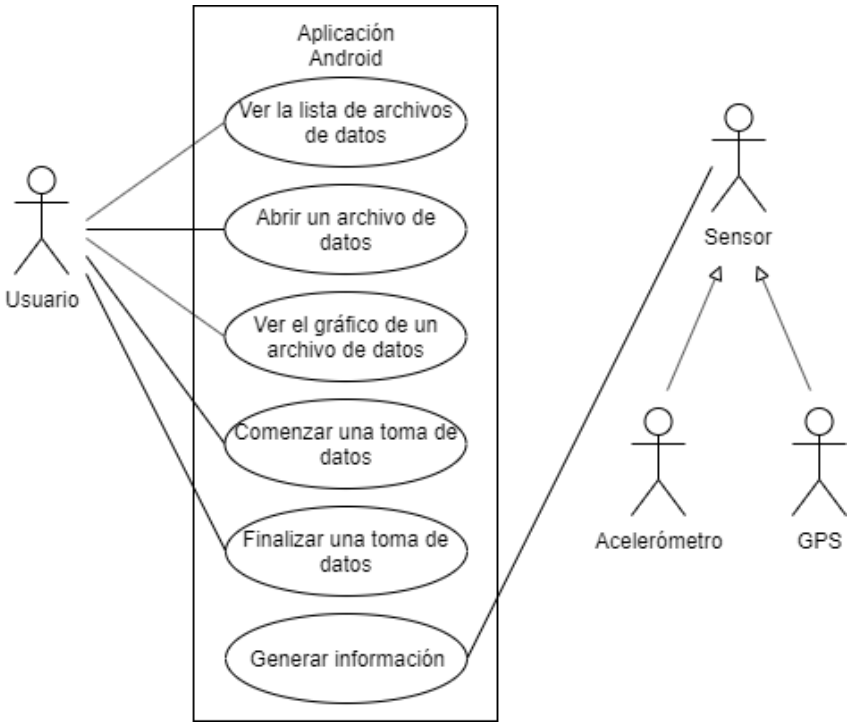


Ilustración 18 – Casos de uso de la aplicación Android.

NOMBRE	Ver la lista de archivos de datos
DESCRIPCIÓN	El usuario visiona la lista de todos los archivos de datos que se encuentran en el teléfono móvil.
ACTORES	Usuario
PRECONDICIONES	Debe haberse realizado una toma de datos para que la lista tenga al menos un elemento.
POSTCONDICIONES	El usuario visualiza la lista de archivos de datos.
FLUJO	1. El usuario accede a la aplicación. 2. El usuario visualiza la lista de archivos de datos.
FLUJO ALTERNATIVO	1. El usuario accede a la aplicación. 2. El usuario no visualiza ningún archivo de datos puesto que no se ha realizado nunca una toma de datos.

NOMBRE	Abrir un archivo de datos.
DESCRIPCIÓN	El usuario accede y visualiza el contenido de un archivo de datos.
ACTORES	Usuario.
PRECONDICIONES	El usuario debe haber realizado una toma de datos previamente.
POSTCONDICIONES	El usuario visualiza el contenido de un archivo de datos.
FLUJO	1. El usuario accede a la aplicación y ve la lista de archivos de datos. 2. El usuario toca un elemento de la lista (un archivo de datos). 3. La aplicación muestra al usuario un menú con varias aplicaciones para visualizar el archivo. 4. El usuario selecciona una aplicación y seguidamente esta se abre mostrando el contenido del archivo de datos.
FLUJO ALTERNATIVO	1. El usuario accede a la aplicación. 2. El usuario no visualiza ningún archivo de datos puesto que no se ha realizado nunca una toma de datos.

NOMBRE	Ver el gráfico de un archivo de datos.
DESCRIPCIÓN	El usuario visualiza el contenido del archivo de datos en forma de gráfico. Concretamente se muestran los datos del acelerómetro.
ACTORES	Usuario
PRECONDICIONES	El usuario debe haber realizado una toma de datos previamente.
POSTCONDICIONES	El usuario visualiza una gráfica con los datos del acelerómetro de una toma de datos en concreto.
FLUJO	1. El usuario accede a la aplicación y visualiza la lista de archivos de datos. 2. El usuario selecciona un elemento de la lista (un archivo de datos) y lo desplaza hacia la izquierda, apareciendo un nuevo botón. 3. El usuario presiona el botón y se abre una nueva vista en la que se muestra el gráfico. 4. Para visualizar correctamente el gráfico, el usuario debe girar 90º el teléfono móvil.
FLUJO ALTERNATIVO	1. El usuario accede a la aplicación. 2. El usuario no visualiza ningún archivo de datos puesto que no se ha realizado nunca una toma de datos.

NOMBRE	Comenzar una toma de datos.
DESCRIPCIÓN	El usuario comienza una toma de datos.
ACTORES	Usuario
PRECONDICIONES	El sensor GPS del teléfono móvil debe estar activado.
POSTCONDICIONES	La aplicación comienza la toma de datos y muestra al usuario la información que está tomando en tiempo real.
FLUJO	1. El usuario activa el sensor GPS del teléfono móvil 2. El usuario entra en la aplicación 3. El usuario presiona el botón con el icono de 'empezar' ▶.

	4. La aplicación cambia de vista, comienza la toma de datos y muestra al usuario la información que obtiene de los sensores en tiempo real.
FLUJO ALTERNATIVO	1. El usuario NO activa el sensor GPS, entra en la aplicación y presiona el botón con el icono de 'empezar' ▶. 2. La aplicación muestra un mensaje de error indicando al usuario que debe activar el GPS. Aun así, la aplicación comienza la toma de datos almacenando en los datos del GPS un cero. 3. El usuario puede realizar la toma de datos de esta manera o encender el GPS y reiniciar la aplicación.

NOMBRE	Finalizar una toma de datos.
DESCRIPCIÓN	El usuario finaliza una toma de datos.
ACTORES	Usuario
PRECONDICIONES	La aplicación está realizando una toma de datos.
POSTCONDICIONES	La aplicación termina una toma de datos.
FLUJO	1. La aplicación se encuentra realizando una toma de datos. 2. El usuario presiona el botón con el icono de 'pausar' ▮▮. 3. La aplicación finaliza la toma de datos y cambia a la vista principal.
FLUJO ALTERNATIVO	

NOMBRE	Generar información
DESCRIPCIÓN	El sensor acelerómetro y el sensor GPS proporcionan información al sistema.
ACTORES	Sensor acelerómetro y sistema GPS.
PRECONDICIONES	La aplicación debe estar realizando una toma de datos.
POSTCONDICIONES	La información proporcionada por ambos sensores se muestra en la pantalla y se almacena en la memoria interna del dispositivo móvil.
FLUJO	1. El usuario comienza la toma de datos presionando el botón de comenzar. 2. Ambos sensores vuelcan la información generada en la aplicación. 2.1. Esta información se muestra en la vista en tiempo real. 2.2. Esta información se guarda en un archivo que se almacena en la memoria interna del teléfono móvil.
FLUJO ALTERNATIVO	

Tras resolver todas las incógnitas que surgieron al inicio del proyecto se pasa a la fase de diseño en la que se define con detalle cómo funciona el sistema, como interactúan todas sus partes entre sí y con elementos externos, y como es usado por el usuario.

8.2. DISEÑO

8.2.1. ESQUEMAS DEL DISPOSITIVO

En este apartado se mostrarán tres esquemas sobre la distribución de los componentes del dispositivo Arduino. La razón de que existan tres bocetos es debido a la metodología incremental e iterativa utilizada en este proyecto. Esta metodología conlleva que en cada iteración el producto se modifique y se le añada nuevas funcionalidades. A continuación, se muestra uno de los primeros diagramas realizados:

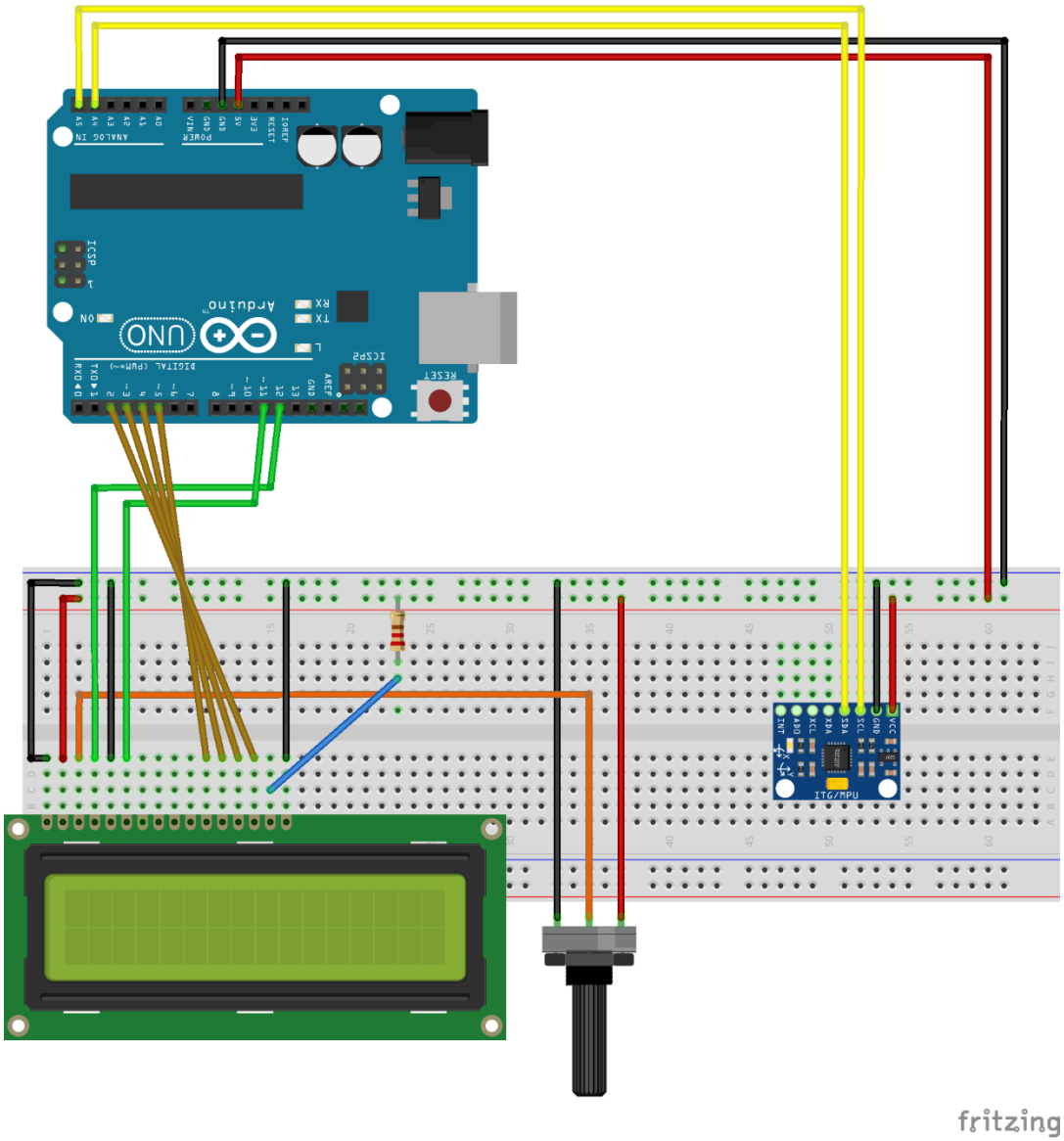


Ilustración 19 – Primer esquema del dispositivo Arduino.

En este esquema se puede observar que hay cuatro componentes (en sentido de las agujas de un reloj): la placa Arduino UNO Rev3, el acelerómetro MPU6050, un potenciómetro (para cambiar la intensidad lumínica de la pantalla LCD) y la pantalla LCD. Esto es debido a que en el comienzo del proyecto se implementaron en el dispositivo solamente el acelerómetro y la pantalla LCD con la intención de probar ambos componentes y ver su comportamiento juntos. A partir de esto se añadieron nuevos componentes como son el sensor GPS, el adaptador para la tarjeta microSD y el adaptador I2C para la pantalla LCD.

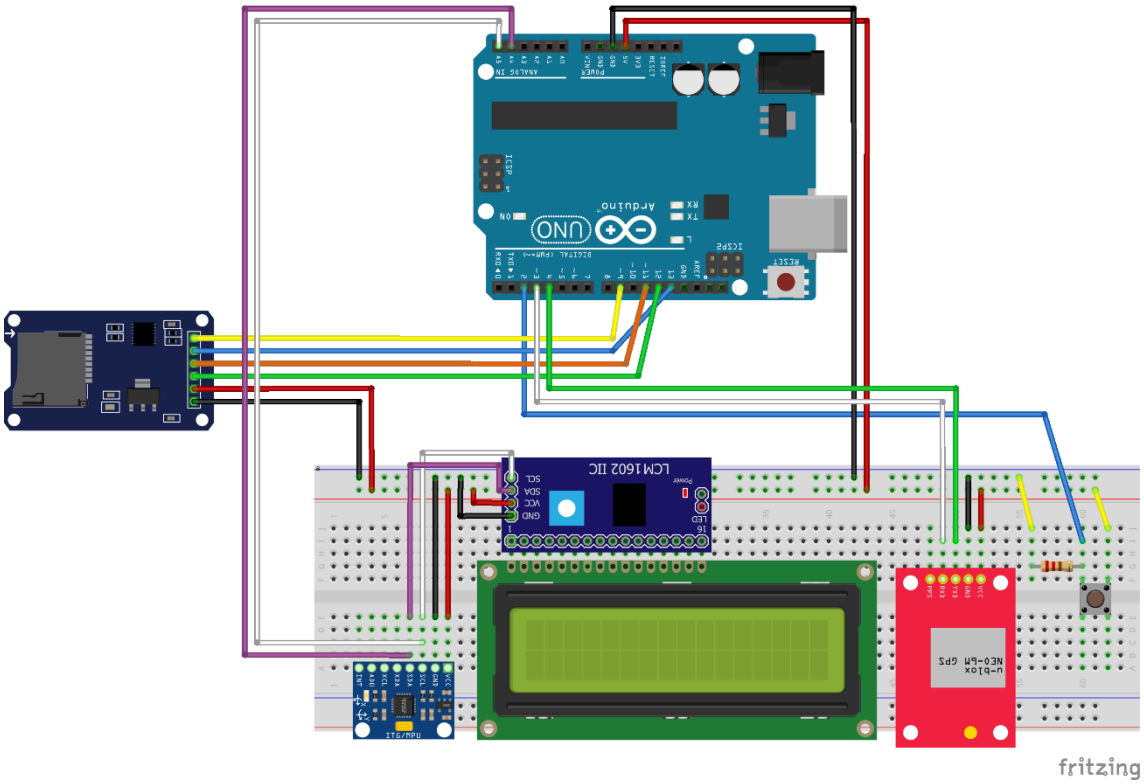


Ilustración 20 – Segundo esquema del dispositivo Arduino.

En este segundo esquema se puede observar cómo se han añadido los tres componentes mencionados en el párrafo anterior. En la imagen, siguiendo el sentido de las agujas del reloj, se encuentra la placa Arduino UNO Rev3, el botón de apagado, el sensor GPS, la pantalla LCD junto con su adaptador I2C, el sensor acelerómetro y el adaptador para tarjetas microSD.

De esta distribución de los componentes cabe destacar que el adaptador para la pantalla LCD permite reducir el número de cables conectados a la placa Arduino de seis a dos. Además, como el sensor acelerómetro y el adaptador utilizan el protocolo de comunicación I2C se conectan a los mismos pines en la placa Arduino, por lo que se reduce aún más el número de conexiones.

Sin embargo, esta colocación de los componentes sigue siendo mejorable. El adaptador para la tarjeta microSD no se encuentra anclado a la placa de prototipado, provocando que el traslado del dispositivo y su utilización sean tediosos. Además, se pueden producir daños en el dispositivo si el adaptador se engancha en algún objeto. Es por esto por lo que se optó por la distribución de los componentes que aparece a continuación:

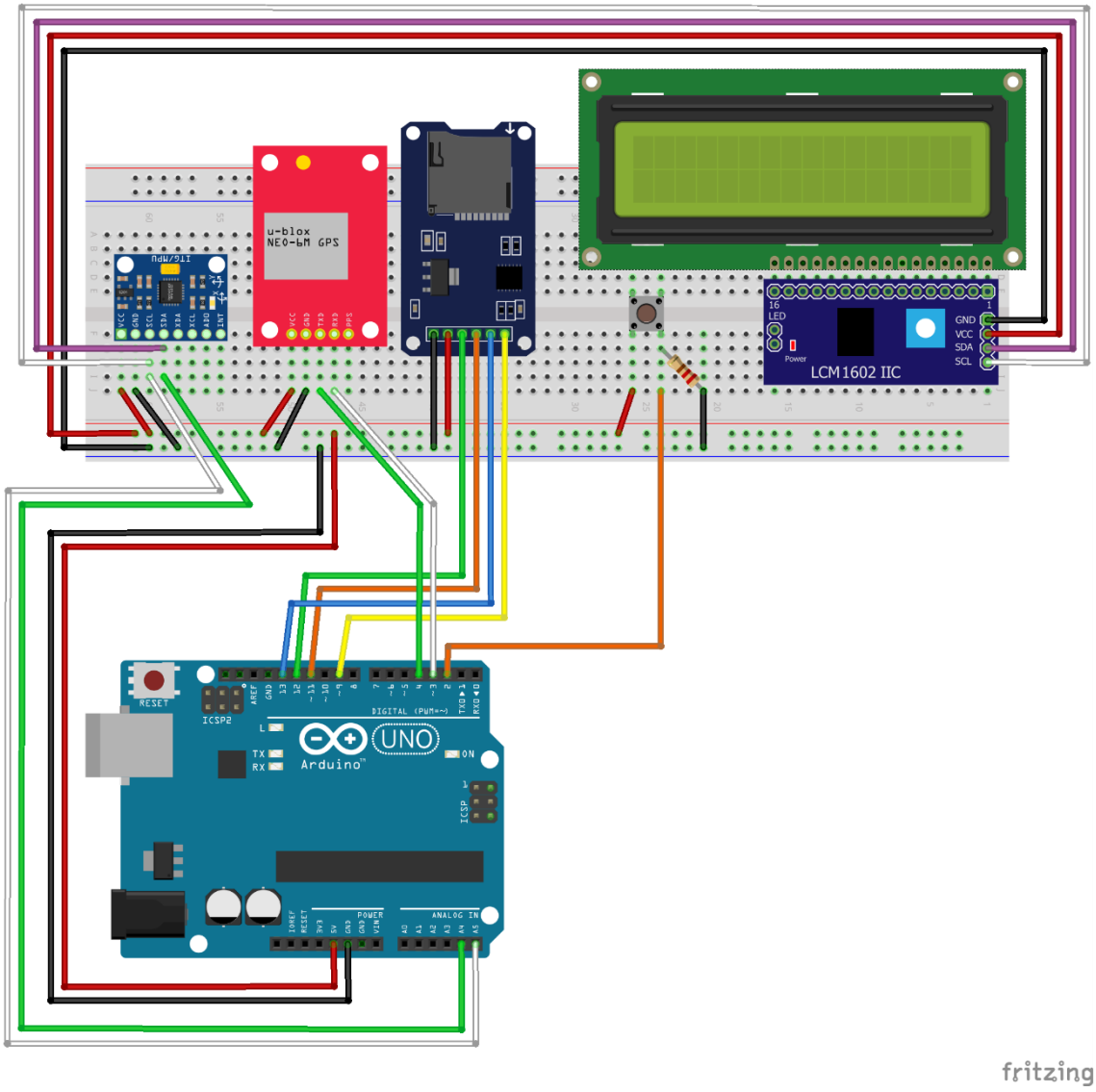


Ilustración 21 – Tercer esquema del dispositivo Arduino.

En esta nueva distribución los componentes son los mismos pero su colocación ha sido modificada. Ahora el adaptador para la tarjeta microSD se encuentra entre la pantalla LCD y el sensor GPS. Esto mejora la movilidad del dispositivo y facilita su utilización.

8.2.2. DIAGRAMAS DE FLUJO

Para obtener una visión global del funcionamiento del dispositivo se ha creado una serie de diagramas de flujo que muestran de manera visual todos los procesos que se deben llevar a cabo mientras el dispositivo está encendido.

La ejecución del programa comienza en el archivo "`main.ino`" que hace de script principal. El diagrama de flujo que aparece a continuación muestra el comportamiento de este archivo:

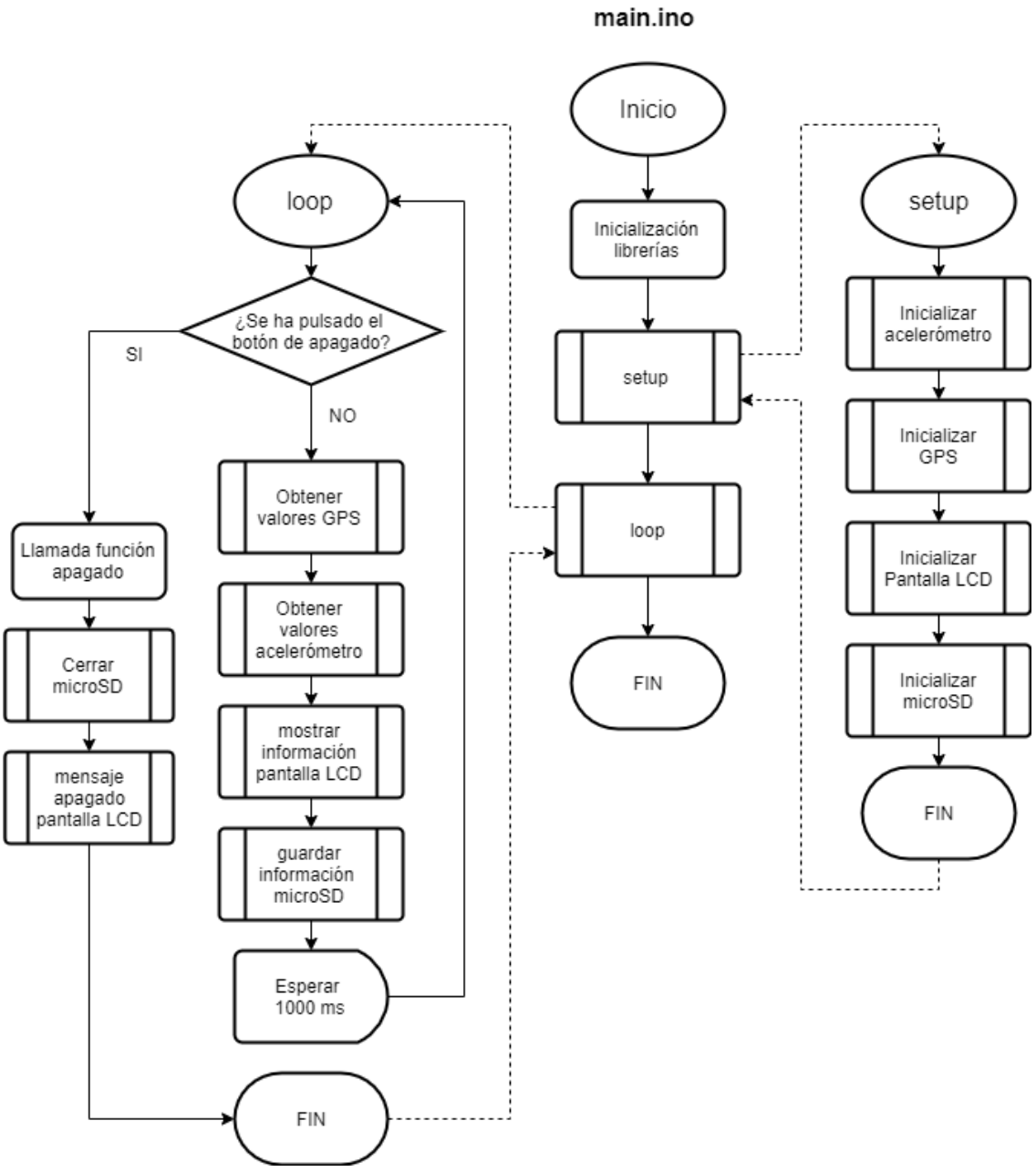


Ilustración 22 – Diagrama de flujo del archivo main.ino.

Como se puede observar, gran parte de las acciones que ocurren durante la ejecución de este archivo son llamadas a otros procesos. Por ejemplo, las acciones que cuelgan del proceso "`setup`" son llamadas a las funciones de inicialización de cada una de las librerías. A continuación, se muestra otro diagrama de flujo en el que se detalla el comportamiento de cada uno de estos métodos:

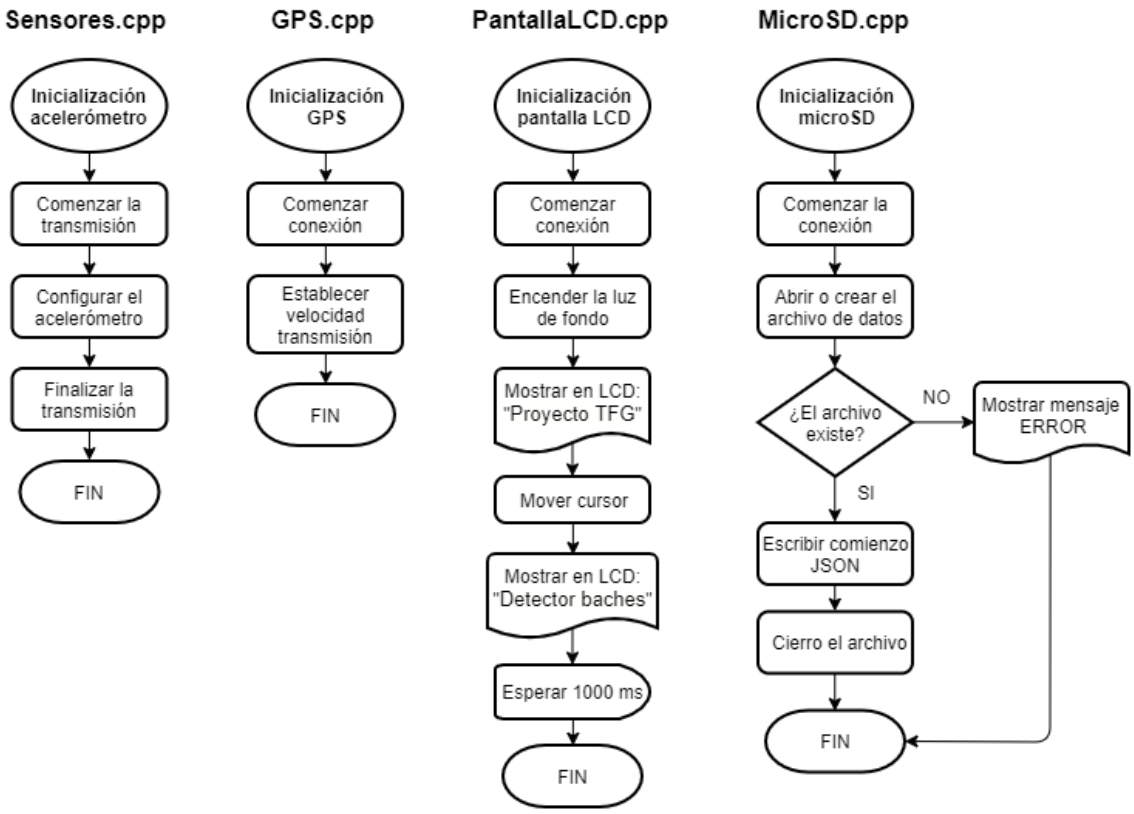


Ilustración 23 – Diagramas de flujo de las funciones de inicialización

Cuando todos estos procesos se ejecutan, los cuatro dispositivos conectados a la placa Arduino UNO Rev3 completan su inicialización y están listos para ser utilizados. Asimismo, la función "`setup()`" que se encuentra en el archivo "`main.ino`" termina su ejecución y da paso a la función "`loop()`". Esta función se puede observar a la izquierda del primer diagrama de este apartado.

Al comienzo del método "`loop()`" se realiza una comprobación del estado del botón de apagado. En el caso de que este no haya sido accionado, se procederá a la ejecución de las funciones que obtienen la información del sensor acelerómetro y del sensor GPS, de la función que muestra en tiempo real los datos de ambos sensores en la pantalla LCD y de la función que los almacena en el archivo dentro de la tarjeta microSD. A continuación, se puede observar cuatro diagramas de flujo que se corresponden con estas cuatro funciones mencionadas:

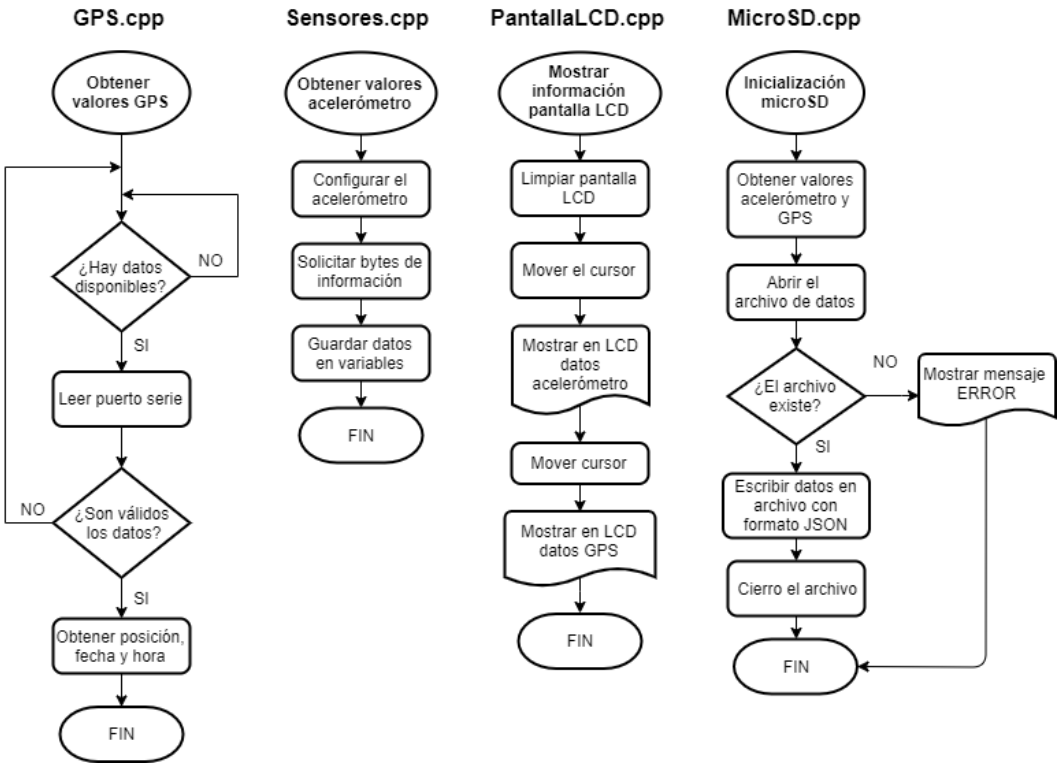


Ilustración 24 – Diagramas de flujo de la obtención de datos y su posterior tratamiento.

Al finalizar la ejecución de estas cuatro funciones, se produce una nueva iteración dentro del método "`loop()`". Esto conlleva que se vuelva a ejecutar la condición que comprueba si el botón de apagado ha sido pulsado o no. En caso negativo, volverían a ejecutarse las cuatro funciones que aparecen en el diagrama anterior. Sin embargo, si el resultado de la condición fuera afirmativo se ejecutarían dos métodos: uno para cerrar el archivo de datos y otro para mostrar un mensaje informando al usuario de que el dispositivo se ha apagado.

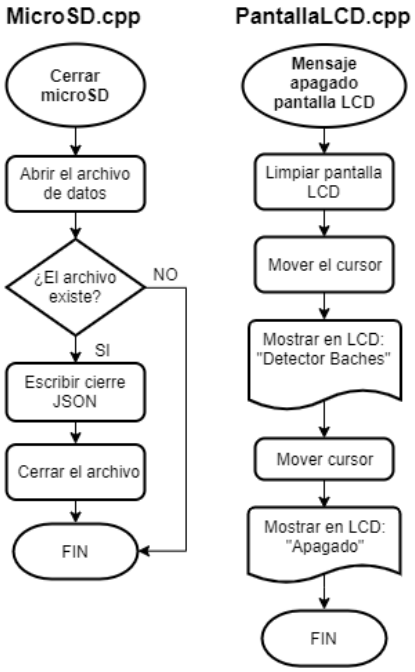


Ilustración 25 – Diagramas de flujos de las funciones de apagado.

8.3. IMPLEMENTACIÓN

En la fase de implementación se lleva a la práctica todos los aspectos tratados en los dos apartados anteriores. Parte del resultado de esta fase es el código que controla el dispositivo Arduino y el código de la aplicación para Android. Ambos se pueden encontrar en el repositorio donde se han subido.

8.3.1. DISPOSITIVO ARDUINO

A continuación, se describen todos los aspectos relacionados con la fase de implementación del dispositivo Arduino. Esta explicación se divide en la parte software y en la parte hardware.

8.3.1.1. SOFTWARE

El apartado software comienza con la descripción del script principal "`main.ino`". Este archivo controla el resto de los componentes software utilizados. A continuación, se detalla en profundidad.

8.3.1.1.1. Script principal: main.ino

Este script está escrito en el lenguaje propio de Arduino y en él se importan y se inicializan las librerías creadas para cada componente y se realiza, de manera iterativa, la toma de datos. En cada iteración se toman los datos del GPS y del acelerómetro, se muestran en la pantalla LCD y se guardan en la tarjeta *microSD*.

La inicialización de todos los componentes se produce dentro de una función propio del lenguaje de Arduino llamada "`setup()`". Este método es llamado una única vez al comienzo de la ejecución del programa y su objetivo principal es inicializar variables y librerías.

Por otra parte, las iteraciones que se mencionan anteriormente se producen gracias a otra función propia del lenguaje de Arduino llamada "`loop()`". Como su propio nombre indica, se trata de una función que se ejecuta en bucle. Esto es ideal para controlar la placa *Arduino UNO Rev3* y los dispositivos que están conectados a ella.

Por último, en este script posee una pequeña función que permite el apagado del dispositivo de manera controlada, evitando que los datos se corrompan.

8.3.1.1.2. Librerías desarrolladas

Para poder controlar todos los componentes hardware conectados a la placa Arduino UNO Rev3, se han desarrollado una serie de librerías en el lenguaje C++. Estas librerías están formadas por dos archivos: un archivo de cabecera (.h) y un archivo fuente (.cpp).

Por un lado, en la cabera se define la clase y todos los elementos que la componen, tanto funciones como variables. Además, en este archivo se declaran que elementos son públicos y cuales son privados. Por convenio, los elementos privados deben tener delante de su nombre el carácter '_'. Por otro lado, en el archivo fuente se crea un constructor para la clase y se implementan todas las funciones y variables que han sido declaradas en el archivo cabecera.

Como se mencionó anteriormente, se han desarrollado cinco librerías que se detallarán a continuación:

- *Librería GPS*

Esta librería tiene como objetivo controlar el módulo GPS, es decir, es la que se encarga de obtener la latitud y la longitud en la que se encuentra el dispositivo. Para llevar esto a cabo solo son necesarias dos funciones:

- El método "`initialize()`" inicializa la conexión con el módulo estableciendo la velocidad de comunicación
- El método "`getValues()`" obtiene los datos del sensor GPS cuando están disponibles y los almacena en dos variables públicas (*lat* y *lng*) que pueden ser accedidas desde otra librería o desde el script principal.

Para poder implementar correctamente estas funciones, primero es necesario incluir la librería "SoftwareSerial", que permite establecer comunicaciones seriales utilizando cualquiera de los pines digitales de la placa Arduino UNO Rev3. Esto es necesario puesto que el módulo GPS se comunica utilizando el dispositivo UART que permite comunicaciones en serie y asíncronas.

También es necesario importar la librería "TinyGPS" que permite decodificar los datos obtenidos del módulo GPS. Estos datos están estructurados siguiendo el formato de datos estándar NMEA y se requiere de un decodificador para poder extraer la información.

- *Librería Sensores*

Esta librería permite obtener los datos del acelerómetro que está integrado en el módulo GY-521. Está compuesta por dos funciones que, al igual que con la librería GPS, una inicializa el sensor y la otra obtiene los datos. Además, esta librería tiene la peculiaridad de tener un objeto "struct" público en el que se guardan los valores de los ejes X, Y y Z:

- El método "`initialize()`" inicializa la conexión con el módulo GY-521 mediante el protocolo de comunicación I2C.
- El método "`getValues()`" solicita al módulo GY-521 los bytes que contienen la información del acelerómetro. Posteriormente, estos datos se introducen en las variables X, Y y Z del objeto "struct".

El módulo GY-521 está diseñado para que se comunique con otros dispositivos utilizando el bus I2C, por lo que es necesario utilizar la librería "Wire" con la que podemos leer y enviar información a través de este bus.

- *Librería MicroSD*

Los datos del acelerómetro y del GPS se guardan en una tarjeta microSD. Para hacer uso de este dispositivo de almacenamiento se ha creado una librería llamada "MicroSD" formada por tres funciones:

- La primera de ellas, llamada "`initialize()`", inicializa la conexión con la tarjeta microSD y abre el archivo de datos indicando que se quiere realizar operaciones de escritura sobre él. En caso de que el archivo no exista, esta función crea uno nuevo.
- Una segunda función, con nombre "`writeLog(Sensores sensor, GPS gps)`", recibe por parámetros una instancia de la librería "Sensores" y otra de la librería "GPS". Con esto se logra acceder a las variables públicas de ambas librerías, obteniendo así los valores de los ejes X, Y y Z del acelerómetro y la latitud y la longitud del GPS. Esta información se escribe en el archivo que se encuentra en la tarjeta microSD.
- La última función, "`closeLog()`", se ejecuta cuando el usuario decide terminar con la toma de datos pulsando el botón que se encuentra en la placa de prototipado. Su función es la de añadir los caracteres de cierre "`}}`" al final de los datos.

Para poder implementar lo mencionado anteriormente, es necesario hacer uso de la librería "SD" que proporciona una serie de funciones para controlar una tarjeta de memoria como podría ser la utilizada en este proyecto. Principalmente se han utilizado las funciones "`open()`", "`print(data)`" y "`close()`".

- *Librería PantallaLCD*

Durante la toma de datos es importante tener un indicador de que los datos se están obteniendo correctamente. Por ello se ha implementado una pantalla LCD en la que se muestran los valores de los tres ejes del acelerómetro (X, Y y Z) y las coordenadas GPS (latitud y longitud). Para poder realizar esto se ha creado una librería llamada "PantallaLCD" en la que se han implementado tres funciones:

- La función "`initialize()`" se encarga de iniciar la comunicación y mostrar un mensaje de inicio en la pantalla.
- Una segunda función, llamada "`printData(Sensores sensor, GPS gps)`", recibe como parámetros una instancia de la librería "Sensores" y otra de la librería "GPS". El objetivo es poder acceder a las variables públicas de ambas librerías. Con esto se obtiene los valores del acelerómetro y del GPS, para posteriormente mostrarlos en la pantalla.
- La tercera y última función, "`shutdownMessage()`", muestra un mensaje para indicar al usuario que la toma de datos ha finalizado y que puede desconectar el dispositivo de la fuente de alimentación sin que se produzcan daños.

Al igual que con el módulo GY-521 (acelerómetro), la pantalla LCD se comunica utilizando un BUS I2C por lo que es necesario utilizar las librerías "Wire" y "LiquidCrystal_I2C" para lograr una comunicación entre la pantalla y la placa Arduino. Esta segunda librería provee de un conjunto de funciones que permiten controlar la pantalla LCD. Algunos de estos métodos son "`init()`", "`print(data)`", "`setCursor(x, y)`" y "`clear()`".

- *Librería BotonOnOff*

Por último, durante el desarrollo del dispositivo surgió la necesidad de colocar un botón que permitiera un apagado controlado del dispositivo para evitar que los datos se corrompieran y poder cerrar la toma de datos siguiendo el formato JSON. Para ello se creó la librería "BotonOnOff", con una única función llamada "`off()`" con la que se detecta el estado del botón y se modifica el estado de una variable booleana pública. Esta variable es accedida desde el script principal dentro de la función "`loop()`", con lo que en cada iteración se consulta si el usuario ha pulsado o no el botón de apagado.

8.3.1.1.3. Archivo de datos

Como se ha mencionado anteriormente, la información generada por el sensor acelerómetro y por el sensor GPS se almacena en un archivo que se encuentra alojado en una tarjeta microSD. Dentro de este archivo la información se estructura siguiendo un formato propio. En un principio, el objetivo era almacenar los valores siguiendo el formato JSON, pero debido a problemas de memoria se descartó este formato y se optó por crear uno personalizado que utilizara pocos caracteres. Este nuevo formato es una mezcla entre el formato JSON y el formato CSV (*Comma-Separated Values*). En la imagen que aparece a continuación se puede observar un fragmento de un archivo generado tras la toma de datos utilizando el dispositivo. En la línea se puede observar un número al comienzo que sirve de identificador de una captura de datos en concreto. Seguidamente, entre llaves, se muestran varios valores separados por comas:

```
101:{6.24,-1.67,13.72,28076941,-15454284,9135300},
```

Ilustración 26 – Contenido de un archivo de datos.

Los tres primeros valores se corresponden con los ejes del acelerómetro, es decir, el valor de X es “6.24”, el de Y es “-1.67” y el de Z es “13.72”. Los dos siguientes valores se corresponden con la latitud y la longitud expresados sin formato, es decir, el valor de la latitud “28076941” en grados decimales es “28,076941”. El último valor se corresponde con la hora a la que se realiza la toma de datos. Por ejemplo, el valor “9135300” indica que ese dato se capturó a las 9 horas, 13 minutos y 53 segundos.

Todas las tomas de datos realizadas en el dispositivo se almacenan en un mismo archivo de datos alojado dentro de la tarjeta microSD. Esta medida se adoptó para reducir la complejidad de los algoritmos utilizados para controlar el almacenamiento de los datos. Por ello es necesario indicar cuando comienza y termina una toma de datos dentro del archivo. La solución implementada consiste en utilizar la misma estructura del formato JSON:

```
{"root":{
1:{4.78,-1.69,14.41,0,0,0},
2:{4.72,-1.64,14.38,0,0,0},
          (. . .)
100:{6.28,-1.65,13.80,28076941,-15454284,9135300},
}}
```

El comienzo se indica con la ristra de caracteres “`{"root":{`”. La palabra *root* se utiliza para señalar que ese el punto raíz, es decir, desde esa línea hasta los caracteres de cierre “`}}`” cuelgan todos los datos de una toma de datos.

En esta muestra de datos también se puede observar que, en el comienzo, los valores del sensor GPS y de la hora están a cero. Esto es debido a que en los primeros segundos de la toma de datos el GPS no ha establecido la conexión con el sistema de posicionamiento global y por lo tanto no puede proporcionar los valores al dispositivo.

8.3.1.2. HARDWARE

Un aspecto importante del desarrollo del dispositivo es la implementación de todos componentes físicos. Esto se describe a continuación:

8.3.1.2.1. Colocación de los componentes

A partir de los esquemas del dispositivo Arduino desarrollados en la fase de diseño se procede a su implementación. A continuación, se puede observar una comparación entre el último esquema diseñado y su puesta en práctica:

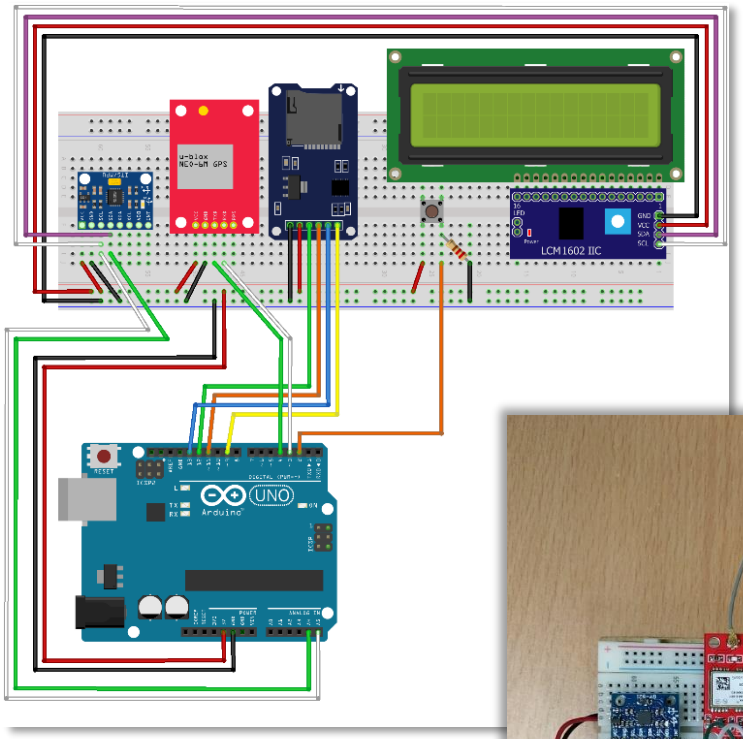


Ilustración 27 – Esquema del dispositivo Arduino.

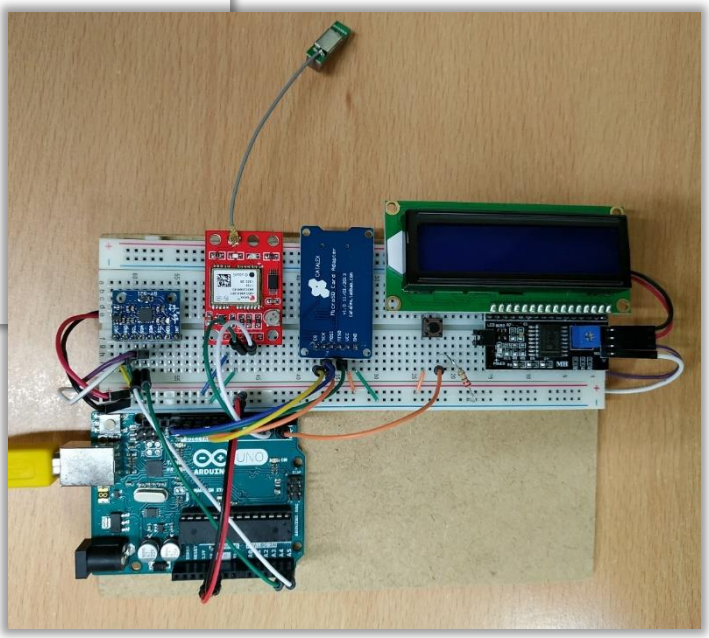


Ilustración 28 – Implementación del dispositivo Arduino.

Como se puede observar, en ambas imágenes la distribución de los componentes es la misma, solamente varía la posición de los cables. Además, a la hora de implementar el dispositivo se optó por colocar los componentes sobre una tabla para facilitar su movilidad y mejorar su estabilidad.

Antes de llegar a este resultado, se optaron por otras distribuciones de los componentes. A continuación, se muestra la primera de ellas, que se corresponde con esquema de la Ilustración 19, en la que se puede observar cómo solo están conectados la pantalla LCD (sin el adaptador a I2C) y el sensor acelerómetro. Además, a este sensor no se le han soldado los pines y está conectado precariamente con unos cables.

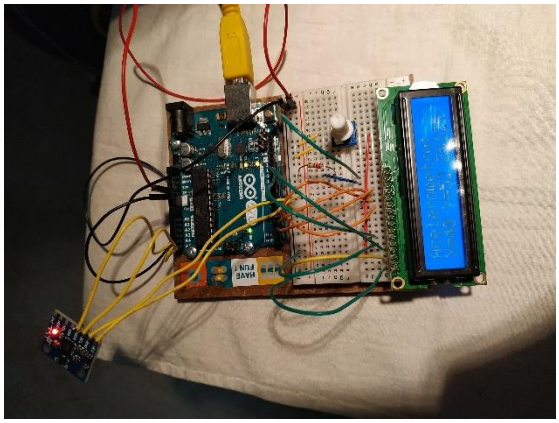


Ilustración 29 – Distribución de los componentes.

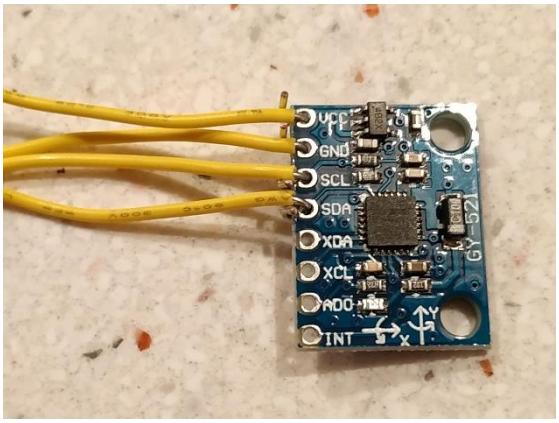


Ilustración 30 – Acelerómetro conectado por cables.

El desarrollo del dispositivo continúa evolucionando y se le añaden nuevos componentes como son el adaptador para la pantalla y el adaptador para la tarjeta microSD. Sin embargo, el sensor acelerómetro sigue conectado mediante cables:

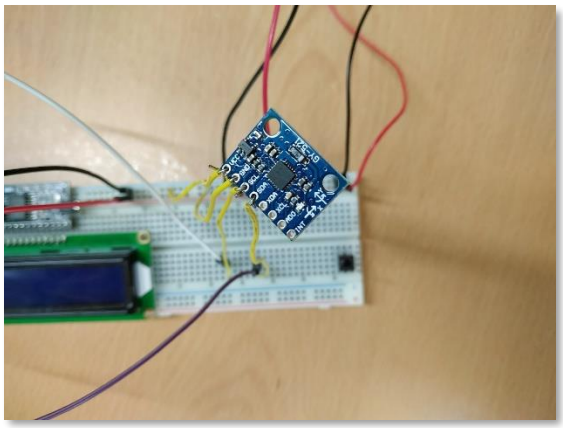


Ilustración 32 – Acelerómetro conectado por cables.

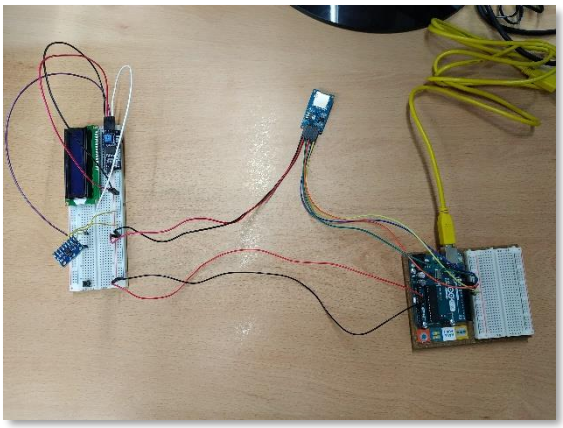


Ilustración 31 - Distribución de componentes.

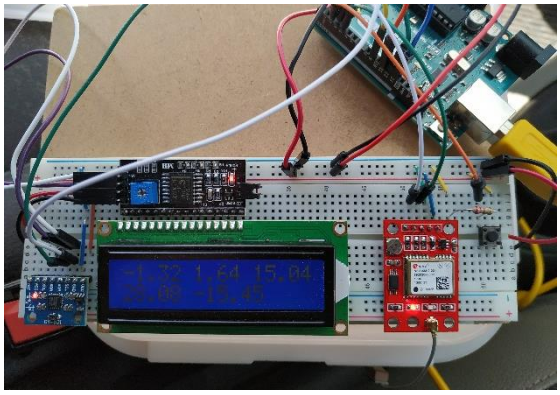


Ilustración 33 – Distribución de los componentes.

A finales del desarrollo se añade el sensor GPS y el sensor acelerómetro se suelda, permitiendo que pueda conectarse directamente a la placa de prototipado. En la imagen que aparece a la izquierda se puede observar lo mencionado:

8.3.1.2.2. Conexiones con la placa Arduino

Partiendo de la última distribución de los componentes (Ilustración 28) se va a explicar la conexión de cada uno de los dispositivos con la placa Arduino UNO Rev3:

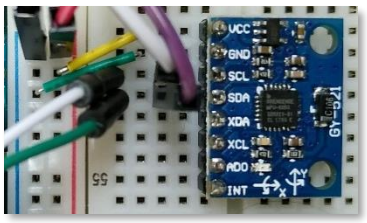


Ilustración 34 – Conexiones acelerómetro.

- *El sensor acelerómetro:* Se comunica utilizando el protocolo I2C. Para ello necesita los siguientes pines:
  - Vcc: Entrada alimentación.
  - Gnd: Salida alimentación.
  - SCL: Transmite la señal de reloj.
  - SDA: Transmite los datos.

Ilustración 35 – Conexiones GPS.

- *El sensor GPS:* Utiliza el dispositivo UART para comunicarse. Este posee los siguientes pines:
  - Vcc: Entrada alimentación.
  - Gnd: Salida alimentación.
  - Tx: Transmite información.
  - Rx: Recibe información.

Ilustración 36 – Conexiones adaptador microSD.

- *El adaptado microSD:* Se comunica usando SPI como esclavo (Slave). En total utiliza 6 pines para transmitir información:
  - CS: Selector de esclavo.
  - SCK: Señal de reloj.
  - MOSI: Master Output Slave Input.
  - MISO: Master Inpurt Slave Output.
  - Vcc: Entrada alimentación.
  - Gnd: Salida alimentación.

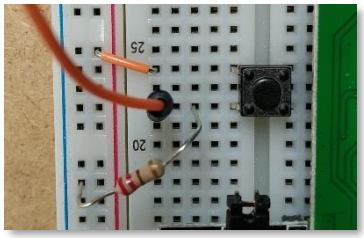


Ilustración 37 – Conexiones botón apagado.

- *El botón de apagado:* Su conexión es muy simple. Utiliza una resistencia *pull down* para mandar una señal:
  - Cable naranja corto: Entrada alimentación.
  - Cable naranja largo: Envío señal.
  - Resistencia: Salida alimentación

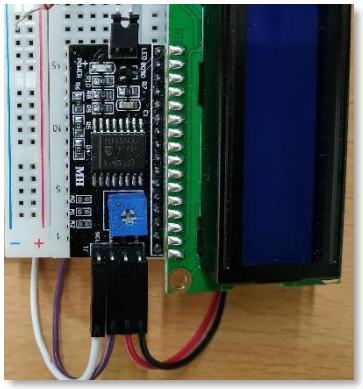
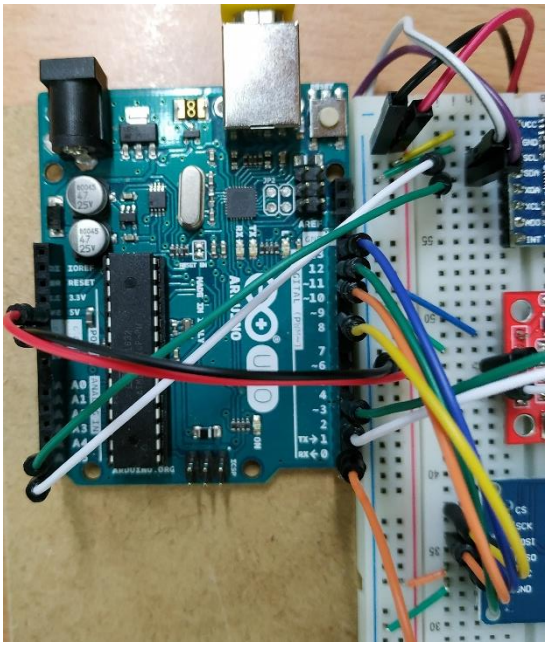


Ilustración 38 – Conexiones adaptador pantalla LCD.

- *Adaptador pantalla LCD:* La pantalla LCD se conecta a este adaptador y transforma sus dieciséis pines en cuatro pines que se comunican usando el protocolo I2C. Por lo tanto, sus conexiones son idénticas a las del sensor acelerómetro:
  - Vcc: Entrada alimentación.
  - Gnd: Salida alimentación.
  - SCL: Transmite la señal de reloj.
  - SDA: Transmite los datos.

Al otro lado de los componentes se encuentra la placa Arduino UNO Rev3. A continuación, se describe la función de cada uno de los pines utilizados:



- Tanto el *sensor acelerómetro* como el *adaptador de la pantalla* utilizan el protocolo I2C para comunicarse. Su conexión con la placa Arduino se lleva a cabo mediante los cables verde y blanco a los pines analógicos A4 y A5.
- El *sensor GPS* se conecta a la placa Arduino en los pines digitales 3 (Tx) y 4 (Rx).
- El *adaptador microSD* se conecta en los pines digitales 13, 12, 11, 9.
- El *botón de apagado* se conecta a la placa Arduino en el pin digital 2 con el cable naranja largo.

Ilustración 39 – Conexiones de la placa Arduino UNO Rev3.

8.3.2. APLICACIÓN ANDROID

Como se ha comentado anteriormente, durante el desarrollo de este proyecto surgió la necesidad de comparar los datos obtenidos para asegurar el correcto funcionamiento del dispositivo y poder determinar cuál es la mejor solución para el problema planteado en este trabajo. Por ello se desarrolló una aplicación móvil para el sistema operativo Android haciendo uso del framework IONIC. Este marco de trabajo permite utilizar módulos con los que acceder al hardware del móvil, logrando con ello obtener datos del acelerómetro y del GPS, y por guardar la información en el almacenamiento interno. A continuación, se describe el desarrollo y el comportamiento de la aplicación:

8.3.2.1. INTERFAZ DE USUARIO

En este apartado se muestran un total de ocho capturas de pantalla de la aplicación para Android. El objetivo es explicar el funcionamiento de la aplicación y demostrar cómo ha sido implementada. A continuación, se muestran dos de estas capturas:

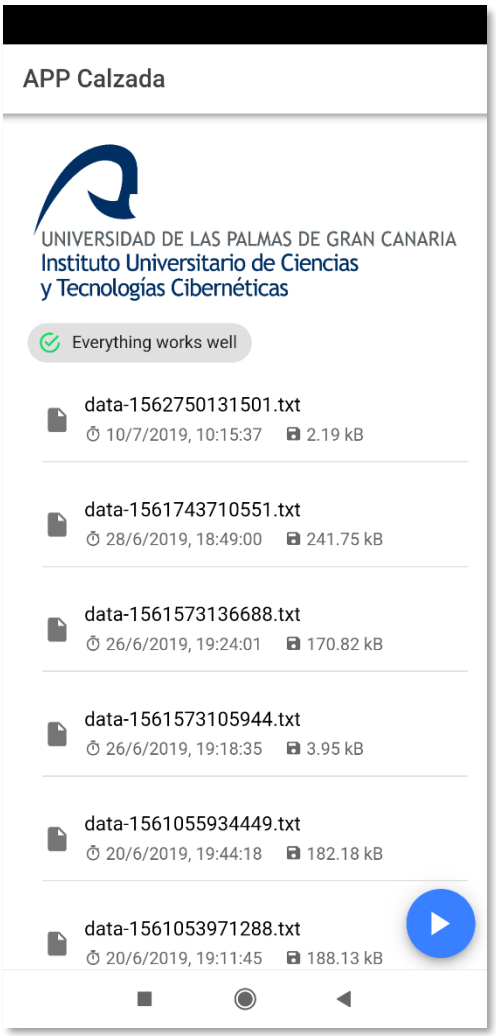


Ilustración 41 – App: Pantalla Principal.

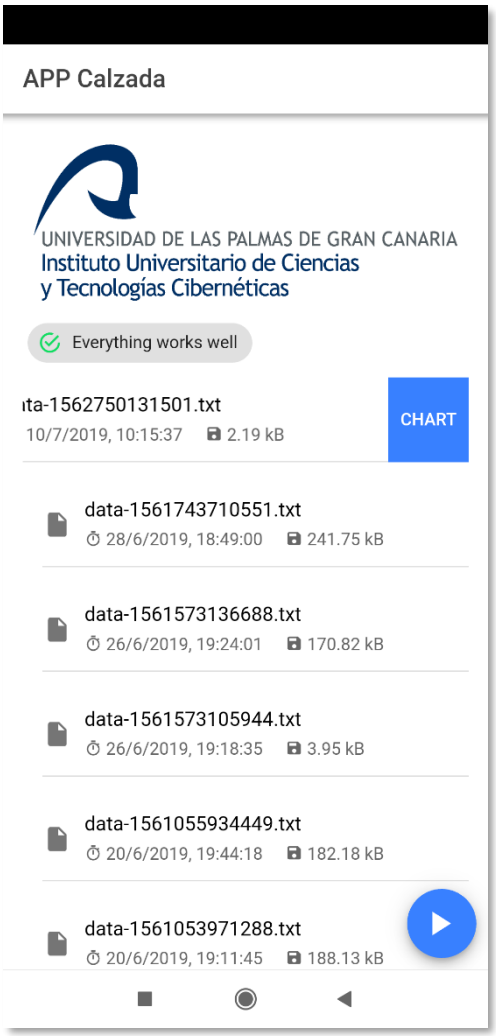


Ilustración 40 – App: Botón deslizante.

En la Ilustración 41 se puede observar la pantalla principal de la aplicación. Está formada por una lista de los archivos de datos tomados con la propia aplicación, un botón de inicio del proceso de toma de datos, un mensaje informativo del estado de la aplicación y el logo de la Universidad de Las Palmas de Gran Canaria y el Instituto Universitario de Ciencias y Tecnologías Cibernéticas.

Cada elemento de la lista se caracteriza por tener el nombre del archivo al que representa, la fecha y la hora en la que se creó y el tamaño en Kilobytes. Además, si se desplaza hacia la izquierda aparece a la derecha un nuevo botón con el texto “CHART” (Ilustración 40). Al pulsar este botón se cambia la vista de la aplicación y se muestra una gráfica con los datos del acelerómetro que se encuentran dentro del archivo de datos.

Otro elemento que cabe destacar de esta vista de la aplicación es el botón de inicio de la toma de datos. Al ser presionado la lista de archivos desaparece y surgen dos tarjetas que contienen los datos en tiempo real del sensor acelerómetro y del sensor GPS. Esto se puede observar en las imágenes que aparecen a continuación:

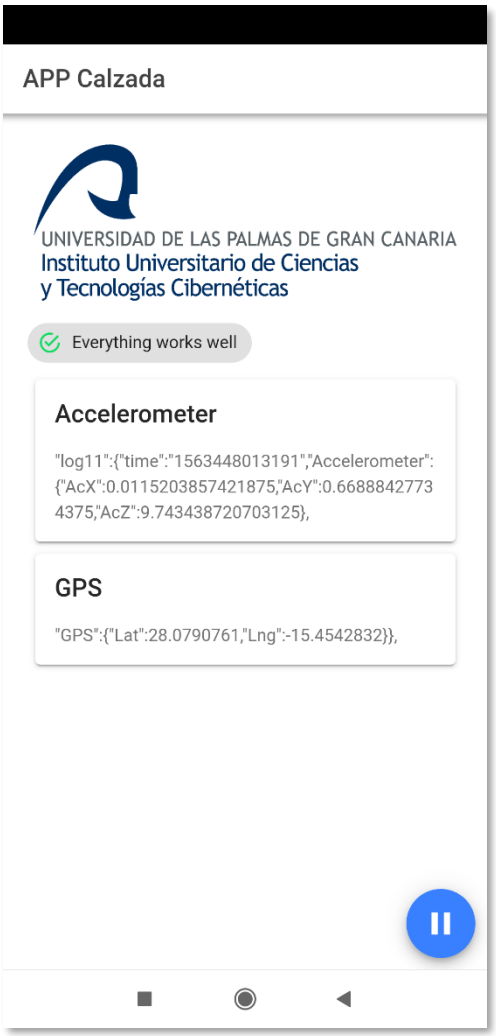


Ilustración 42 – App: Toma de datos.

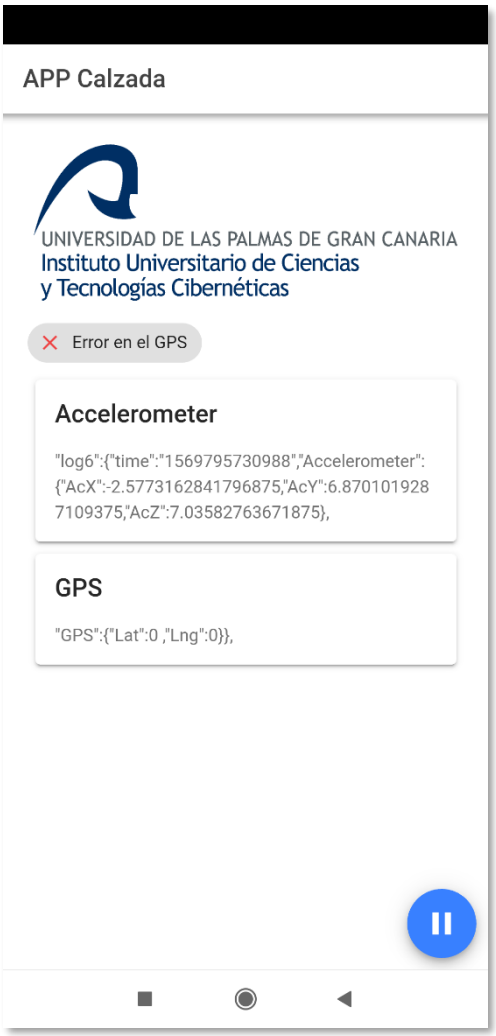


Ilustración 43 – App: Error en el GPS.

En la Ilustración 42 se puede observar que en el lugar donde se encontraba la lista de archivos ahora se muestran dos tarjetas con la información obtenida de los sensores:

- En la primera tarjeta se muestra principalmente la información del acelerómetro. Esta está compuesta por los ejes X, Y y Z (en la aplicación "AcX", "AcY" y "AcZ"). Además, se muestra también una marca temporal en formato Unix Epoch del momento en el que se realiza la consulta a los sensores. Este dato se puede observar después de la palabra "time". Por último, en esta tarjeta también se muestra un identificador único para cada una de las consultas a los sensores. Por ejemplo, en el momento en el que se realizó la captura de pantalla, la aplicación realizó la consulta número once ("log11").
- En la segunda tarjeta, se muestra la latitud y la longitud en la que se encuentra el teléfono móvil en ese instante. Esta información proviene del sensor GPS.

Por otra parte, en la Ilustración 43 se muestra el mismo comportamiento descrito anteriormente con la diferencia de que en el mensaje informativo se advierte al usuario de que el sensor GPS no se ha encendido. Esto conlleva a que en la segunda tarjeta los valores de latitud y longitud tengan como valor cero.

Otro elemento que se ha modificado es el botón de la esquina inferior derecha que aparece en ambas imágenes. Este botón permite ahora parar la toma de datos y volver a visualizar la lista de archivos.

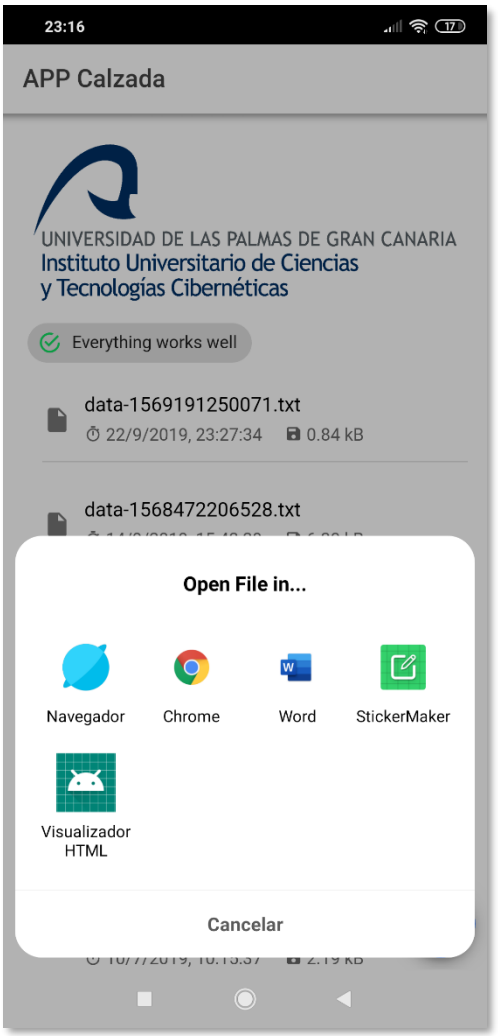


Ilustración 44 – App: Abrir archivo de datos.

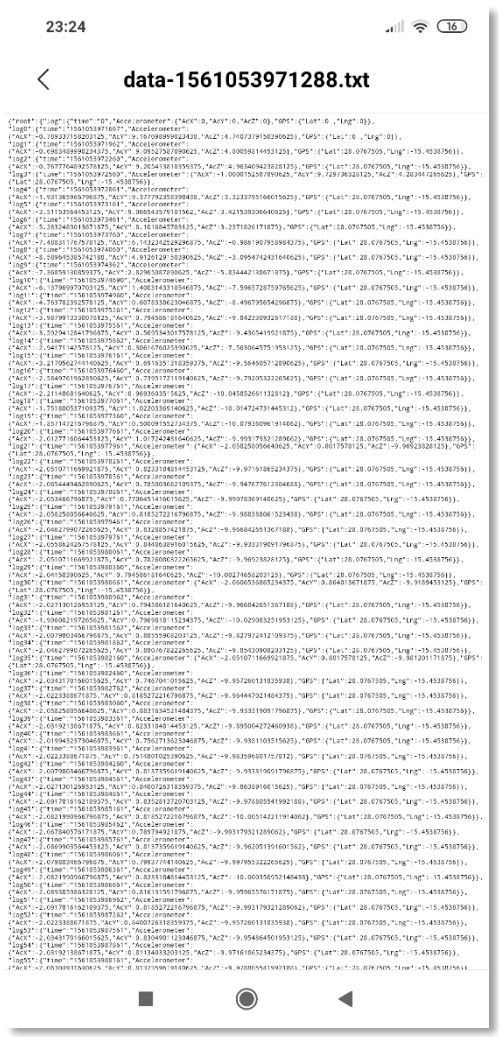


Ilustración 45 – App: Contenido archivo de datos.

Volviendo a la pantalla principal, si el usuario desea visualizar el contenido de uno de los archivos de datos solamente tiene que dar un toque sobre él en la lista de archivos. Esta acción abrirá el menú que aparece en la Ilustración 44, donde el usuario puede seleccionar una aplicación de lectura de archivos.

Seleccionando la aplicación “Visualizador HTML” se puede visualizar el contenido del archivo tal y como aparece en la Ilustración 45.

Como ya se mencionó anteriormente, si se quisiera visualizar estos datos de manera gráfica el usuario debe volver a la pantalla principal y, en la lista, desplazar el archivo a la izquierda revelando el botón “CHART” tal y como aparece en la Ilustración 40. Esto conlleva a que la aplicación muestre una gráfica con todos los valores tomados de los tres ejes del acelerómetro. En la imagen que aparece a continuación se muestra la gráfica de los datos que aparecen en la Ilustración 46.

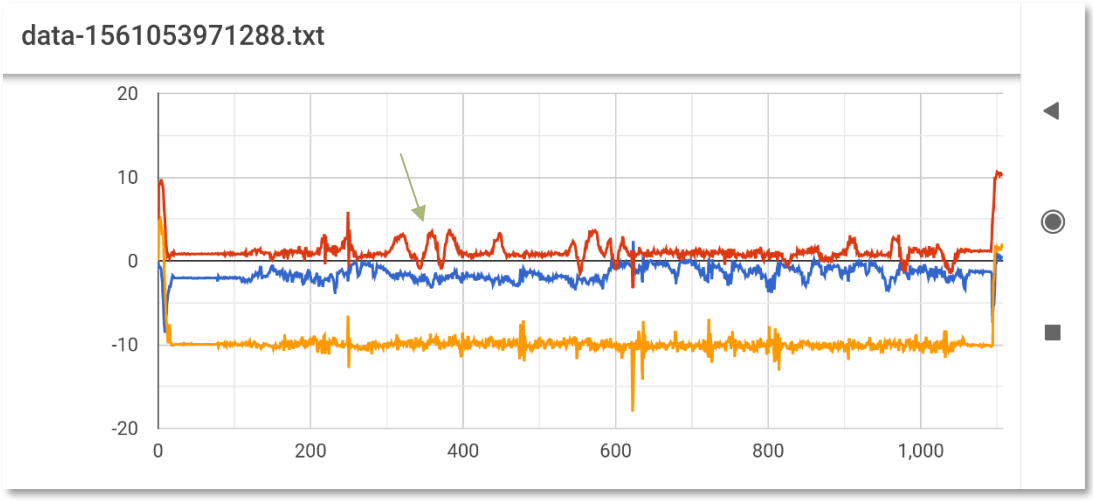


Ilustración 47 – App: gráfica de un archivo de datos.

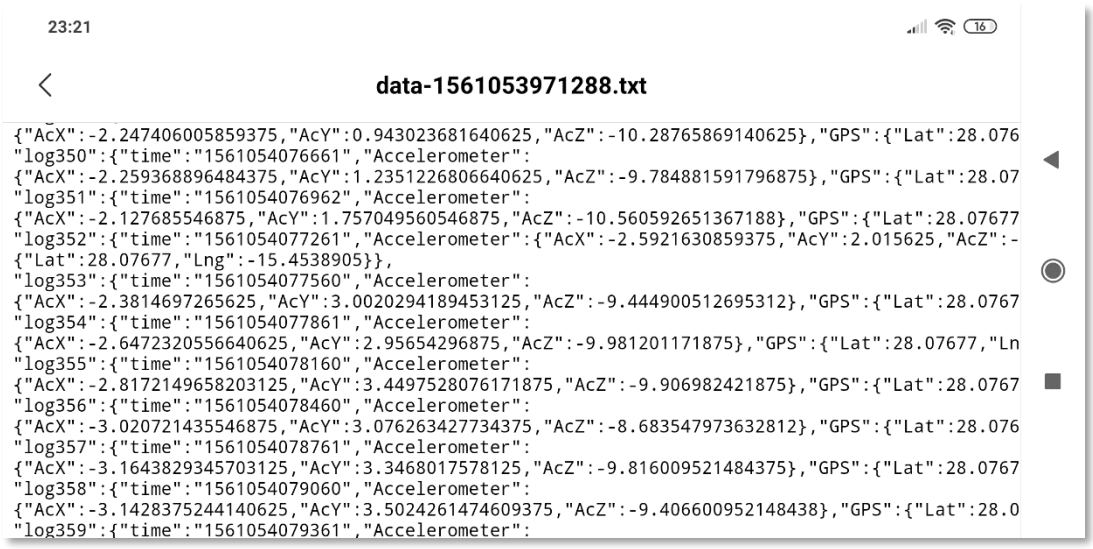


Ilustración 46 – App: fragmento de un archivo de datos.

Como se puede observar en la gráfica (Ilustración 47), existen tres líneas que se corresponden con los ejes X, Y y Z del acelerómetro. En color rojo se observa el valor del eje Y, en color azul el eje X y en amarillo el eje Z. También se puede observar una flecha verde (colocada posteriormente editando la imagen) que señala un pico en el eje Y. Estos valores se pueden observar en la Ilustración 46 junto a la etiqueta "AcY". En ellos se puede apreciar como el valor asciende de "0.94" a "3.50".

En función de la colocación del teléfono móvil en el vehículo se deberá prestar atención a un eje u otro. La toma de datos que aparece representada en la gráfica se tomó con el teléfono móvil colocado verticalmente, de manera que el eje que mejor representa la vibración vertical del vehículo es el eje Y.

Ilustración 49 – Vehículo circulando.

Ilustración 48 – Vehículo y móvil.

8.3.2.2. ESTRUCTURA DEL CÓDIGO

Las aplicaciones desarrolladas en el marco de trabajo IONIC deben cumplir con una estructura de código determinada. Como se ha mencionado anteriormente, la aplicación desarrollada en este trabajo tiene dos vistas. Una vista principal en la que se muestra la lista de archivos (Ilustración 41) y los datos obtenidos por los sensores cuando se comienza una toma de datos (Ilustración 42), y una segunda vista en la que se muestra la gráfica con los valores del acelerómetro que se encuentran almacenados en un archivo (Ilustración 47).

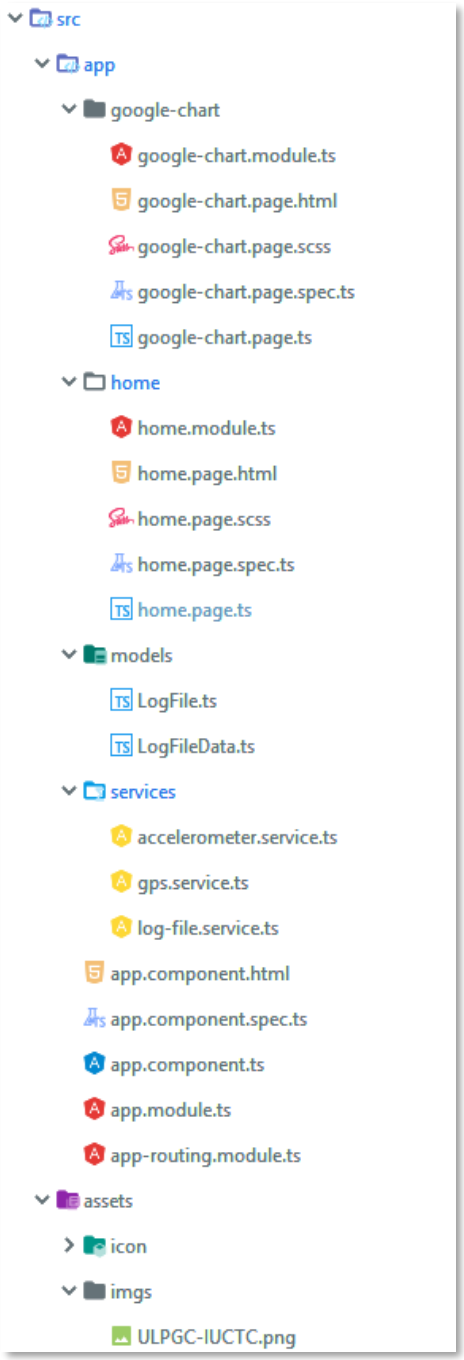


Ilustración 50 – App: Directorios.

Estas dos vistas se ven reflejadas en el directorio "`src/app/`". Concretamente, la vista principal se corresponde con el directorio "home" y la vista secundaria (la vista de la gráfica) con el directorio "google-chart". Aunque en la aplicación ambas vistas son distintas, en el código fuente comparten muchas similitudes. A primera vista, se puede observar que ambos directorios contienen cinco archivos y que la terminación de estos es igual en ambos casos. Los archivos terminados en "*.page.ts" contienen la lógica del programa, los terminados en "*.page.html" y "page.scss" estructuran la vista, los archivos "*.module.ts" son utilizados para labores de configuración y los "*.spec.ts" para realizar las pruebas de código.

El tercer directorio que se observa es el llamado "models". En él se encuentran los archivos "LogFile.ts" y "LogFileData.ts", cuyo objetivo es servir de modelo para los archivos y su contenido. Es decir, los archivos que se muestran en la lista que aparece en la vista principal son instancias del modelo "LogFile.ts". Los mismo ocurre cuando los datos que se almacenan en estos archivos son leídos por la aplicación. La información extraída se modela utilizando "LogFileData.ts".

Por último, el directorio "services" contiene tres archivos. Estos son los encargados de extraer la información de los sensores y de almacenarla en la memoria interna del teléfono móvil. Como es de suponer, el archivo "accelerometer.service.ts" se encarga del sensor acelerómetro, al igual que el archivo "gps.service.ts" se encarga del sensor GPS. El último archivo, el "log-file.service.ts" realiza toda la lógica relacionada con la escritura y lectura de los archivos de datos.

Cabe mencionar que en la carpeta "assets" se almacenan recursos adicionales como puede ser el logo de la Universidad ULPGC y el Instituto de Investigación IUCTC.

8.3.2.3. PLUGINS UTILIZADOS

Anteriormente se ha mencionado que el marco de trabajo utilizado para el desarrollo de la aplicación móvil ha sido IONIC. Al igual en el desarrollo del dispositivo Arduino se utilizaron librerías, en este entorno se pueden utilizar módulos (o *plugins*) para añadir funcionalidades a la aplicación. A continuación, se describen cuatro módulos utilizados en la aplicación:

- *File y FileOpener:*

La información generada por el sensor acelerómetro y por el sensor GPS se guarda dentro de un archivo en texto plano en el almacenamiento interno del dispositivo móvil. Para implementar esto se ha recurrido a un módulo llamado "File" [86], que permite leer y escribir archivos, y a "FileOpener" [87], un módulo con el que abrir archivos desde la aplicación. Algunas de las funciones utilizadas son: "`listDir()`" para listar el contenido de un directorio, "`writeFile()`" para escribir en un archivo, "`readAsText`" para leer un archivo y "`createDir()`" para crear un directorio.

Todos los datos se almacenan siguiendo el formato JSON que aparece a continuación:


```
"log0": {
  "time": "1561743710885",
  "Accelerometer": {
    "AcX": 0.000762939453125,
    "AcY": 10.124786376953125,
    "AcZ": -0.265533447265625
  },
  "GPS": {
    "Lat": 28.0764815,
    "Lng": -15.4538942
  }
},
```


Ilustración 51 – App: Formato JSON.

Como se puede observar, en cada toma de datos se almacena la fecha y hora en la variable "time" siguiendo el formato Unix Epoch (número de segundos transcurridos desde el uno de enero de 1970 a las 00:00). También se almacenan los ejes X, Y y Z del acelerómetro dentro de la variable "Accelerometer" y la posición GPS, es decir, la latitud y la longitud, en la variable "GPS". En la imagen que aparece a continuación se muestra cómo se guarda esta información dentro del archivo de datos, donde cada una de las líneas corresponde a una toma de datos.


```
{"root":{"log":{"time":"0","Accelerometer":{"AcX":0,"AcY":0,"AcZ":0},"GPS":{"Lat":0 ,"Lng":0}},
"log0":{"time":"1561743710885","Accelerometer":{"AcX":0.000762939453125,"AcY":10.124786376953125,"AcZ":-0.265533447265625},"GPS":{"Lat":28.0764815,"Lng":-15.4538942}},
"log1":{"time":"1561743711183","Accelerometer":{"AcX":-0.0279693603515625,"AcY":10.151123046875,"AcZ":-0.3589019775390625},"GPS":{"Lat":28.0764815,"Lng":-15.4538942}},
```


Ilustración 52 – App: Contenido del archivo de datos.

- *Google Maps:*

Este módulo llamado "Google Maps" [88] está basado en un módulo para Cordova [89] y permite acceder a las APIs (Applicaction Programming Interface) de Google Maps [90]. Gracias a ellas se puede obtener la posición geográfica en la que se encuentra el teléfono móvil. Además, estas APIs permiten mostrar mapas dentro de la aplicación móvil, trazar rutas sobre ellos y mostrar información sobre lugares.

Para este trabajo solo se ha utilizado la API "LocationService" [91], que permite obtener la latitud y la longitud sin necesidad de mostrar un mapa, utilizando una única función llamada "`getMyLocation()`".

- *Sensors:*

Con este módulo [92] podemos habilitar y utilizar todos los sensores de un dispositivo móvil. Gracias a estos sensores podemos saber cómo es el movimiento del teléfono, su orientación y las condiciones ambientales, como por ejemplo la cantidad de luz, la temperatura o la presión. Para este trabajo se ha utilizado el sensor acelerómetro, ya que con él podemos averiguar la aceleración de los ejes de coordenadas, y con ello determinar la vibración del dispositivo. De este módulo se ha utilizado la función "`enableSensor(TYPE_SENSOR)`" que activa el sensor que se le pasa por parámetro y la función "`getState()`" que obtiene los valores del sensor.

- *Google Chart:*

Para poder mostrar gráficamente los datos de los ejes del sensor acelerómetro se optó por implementar otro módulo de la empresa Google llamado "Google Chart" [93]. Este módulo permite la creación de múltiples tipos de gráficos dentro de la aplicación. Además, el módulo permite la customización total del gráfico permitiendo cambiar características de las líneas, de los ejes, de la leyenda o del texto entre otras muchas opciones. Para mostrar los datos se utilizó un gráfico lineal como el que se muestra en la Ilustración 47.

8.4. PRUEBAS EN UN ENTORNO REAL

Para demostrar que el dispositivo funciona correctamente se realizaron diversas pruebas en un entorno real. Para realizar estas pruebas se utilizó un vehículo de tipo furgón y en su interior se colocó el dispositivo Arduino en el salpicadero y un teléfono móvil sobre un soporte. Este móvil se utilizó para realizar las grabaciones de la carretera y para tomar datos con la aplicación desarrollada. La colocación de ambos dispositivos se puede observar en las dos imágenes que aparecen a continuación:

Ilustración 54 – Soporte para teléfono móvil.

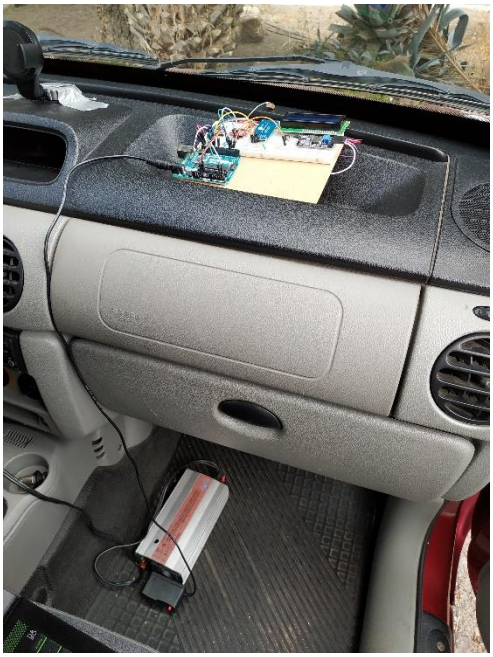


Ilustración 53 – Dispositivo Arduino dentro del vehículo.

El dispositivo Arduino se debe colocar cerca de un cristal puesto que la antena del sensor GPS no es capaz de captar la señal a través de la carrocería del vehículo. Esto no ocurre con el teléfono móvil, sin embargo, se coloca cerca del cristal delantero para poder realizar la grabación de la carretera. En la parte inferior de la Ilustración 53 también se puede observar la existencia de un transformador que permite alimentar el transformado AC/DC de la placa Arduino utilizando la batería del vehículo.

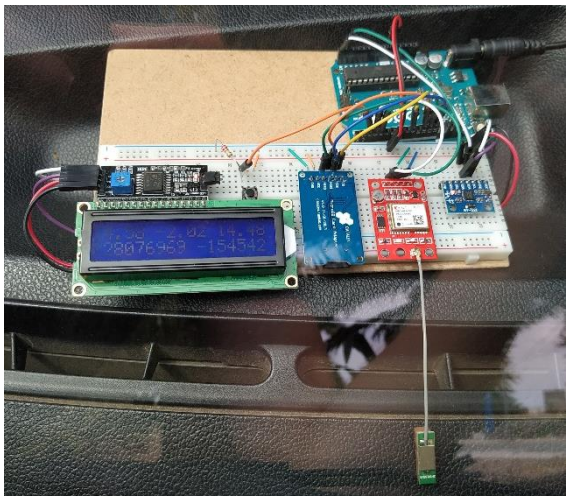


Ilustración 55 – Dispositivo Arduino funcionando.

Una vez el dispositivo Arduino es alimentado, este se enciende y comienza con la toma de datos. En la *Ilustración 55* se puede observar como en la primera línea de la pantalla LCD se muestran los valores de los ejes X, Y y Z del acelerómetro y en la segunda línea aparece la latitud y la longitud obtenidos del sensor GPS.

A continuación, se muestran dos imágenes. Por un lado, en la imagen de la izquierda se muestra una captura de un vídeo correspondiente a una de las pruebas realizadas. Esta grabación se realizó con una aplicación para Android que permite grabar vídeos y añadirle una marca temporal e información del GPS. La aplicación en cuestión se llama "Timestamp Camera Free" [46].

Por otro lado, a la derecha de esta imagen se muestra el recorrido circular [94] realizado para llevar a cabo las pruebas. Este recorrido se encuentra dentro del Campus Universitario de Tafira y, a una media de 40 Km/hora, tiene una duración de siete minutos. Este circuito es idóneo para testear el dispositivo puesto que el estado de la calzada es variado. Existen zonas donde el asfalto está en perfectas condiciones, otras zonas en las que se ha deteriorado, pero no presenta grandes baches y otras en las que hay socavones que obligan a reducir a la mitad la velocidad a la que se circula.

Ilustración 56 – Captura de un vídeo.

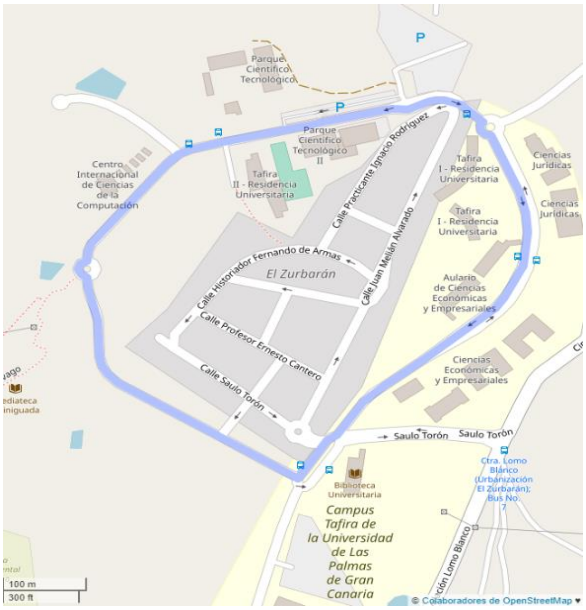


Ilustración 57 – Recorrido realizado para el testeo del dispositivo.

9. RESULTADOS

En este apartado se analizan y se comentan los resultados obtenidos de ambos dispositivos en las pruebas en entornos reales. Como se ha mencionado anteriormente, ambos sistemas proveen al usuario un archivo que contiene los datos obtenidos en la toma de datos. A continuación, se puede observar la comparación de la información obtenida por ambos dispositivos:

En primer lugar se muestra los datos obtenidos por el acelerómetro del dispositivo Arduino. Para poder realizar la gráfica, primero se tiene que extraer la información del archivo y separarlo por columnas en el programa Excel:

	X	Y	Z
1	-1,44	-0,98	14,44
2	-1,42	-0,66	14,21
3	-1,43	-0,65	14,48
4	-1,07	-1,28	15,06
5	-1,2	-0,96	14,47
6	-1,26	-1,27	15,18
7	-1,13	-1,03	14,91

Ilustración 58 – Muestra de datos del acelerómetro del dispositivo Arduino

En total, el dispositivo realizó 465 tomas de datos, de las cuales las 220 primeras han sido obviadas debido a que el vehículo se encontraba estacionado porque el sensor GPS no había establecido la conexión. A continuación, se muestran los datos representados en un gráfico:

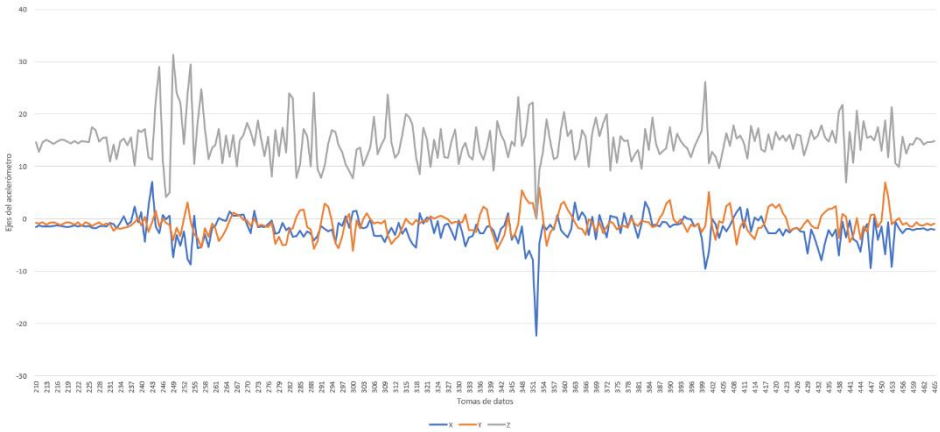


Ilustración 59 – Gráfica de los datos del acelerómetro del dispositivo Arduino.

En segundo lugar, los datos del acelerómetro de la aplicación Android se grafican directamente desde la propia aplicación. A continuación, se puede observar el gráfico generado:

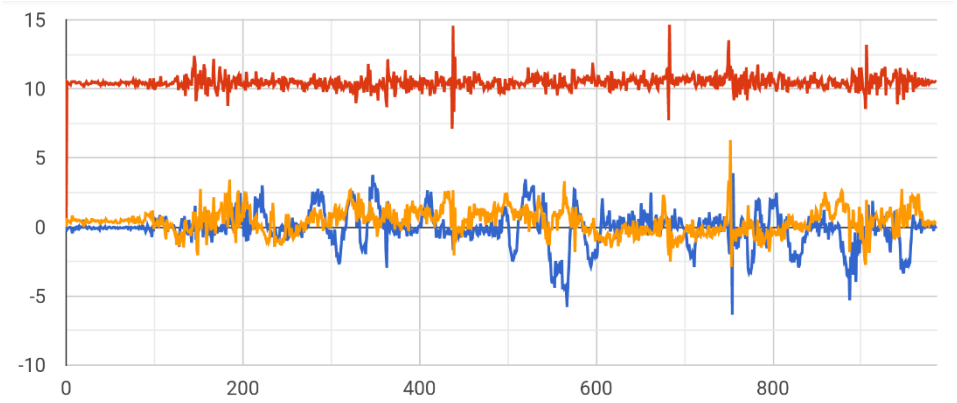


Ilustración 60 – Gráfica de los datos del acelerómetro de la aplicación para Android.

En este caso, no es necesario desperdiciar ningún dato puesto que el sensor GPS estableció la conexión desde el comienzo y se pudo realizar la prueba en un entorno real inmediatamente.

Si comparamos ambos gráficos se puede observar una diferencia en el número de tomas de datos. Esto es debido a que el dispositivo Arduino las realiza cada segundo y la aplicación Android cada 300 milisegundos. Esto ocurre debido a que los componentes hardware del dispositivo Arduino, en concreto el sensor GPS, no responde correctamente cuando el intervalo entre consultas es inferior a un segundo.

También se puede observar de manera gráfica como hay similitudes en los valores de los ejes. En la imagen que aparece a continuación se muestran dos grupos de líneas de color verde que resaltan algunas de estas semejanzas:

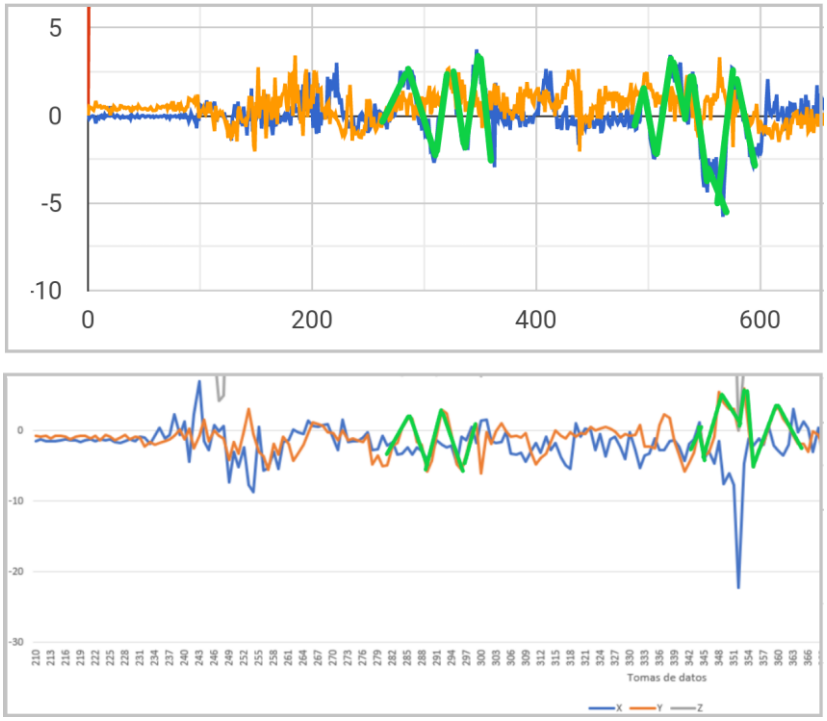


Ilustración 61 – Comparación de ambas gráficas.

10. PROBLEMAS ENCONTRADOS

A lo largo del desarrollo de este proyecto han surgido diversos problemas e imprevistos que han ocasionado que el tiempo invertido aumente. Sin embargo, estos inconvenientes han permitido mejorar las características implementadas y explorar nuevas opciones que en un principio no se habían tenido en cuenta. A continuación, se listan y se describen algunos de los problemas más importantes:

En relación con la placa Arduino UNO Rev3, esta se caracteriza por tener una memoria interna limitada, lo que supone que el código desarrollado debe estar optimizado. Por ejemplo, se puede optimizar cambiando el tipo de dato de las variables, almacenando cadenas de caracteres en la memoria Flash en lugar de en la memoria SRAM o utilizando constantes.

Si esta optimización no se produce pueden surgir problemas de corrupción de memoria, de comportamiento aleatorio o incluso se puede reiniciar la placa Arduino. Por ejemplo, en la imagen que se muestra a continuación se puede observar como las variables que almacenan la información del sensor GPS guardan datos corruptos:

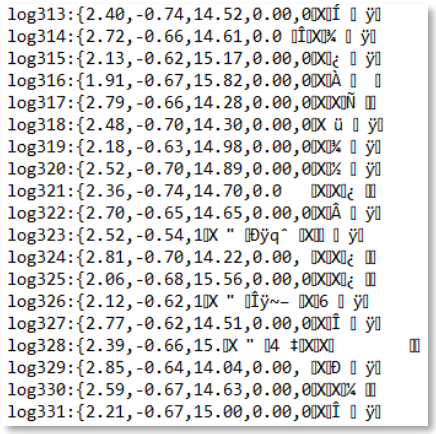


Ilustración 62 – Datos corruptos.

En el apartado físico del dispositivo también surgieron problemas. Como se optó por minimizar el costo del dispositivo, la calidad de los componentes no siempre es la idónea. Por ejemplo, en algunas situaciones la antena del sensor GPS no tiene la potencia suficiente para captar la señal. Esto puede ocurrir cuando el sensor está bajo un techo o cuando el día está muy nublado. También afecta a los tiempos de conexión que fluctúan entre los 30 segundos y los 10 minutos.

También se produjeron problemas con los cables utilizados para conectar el adaptador de la tarjeta microSD a la placa Arduino UNO. El adaptador se comunica mediante la interfaz SPI que requiere que la distancia entre dispositivos sea la menor posible, sin embargo, la longitud de los cables utilizados era demasiado grande (20 centímetros). Esto provocó que la información no se almacenara correctamente e incluso, es posible, que dañara físicamente el adaptador y la tarjeta microSD puesto que ambos se estropearon. Como es de suponer, al disminuir la longitud de los cables los problemas desaparecieron.

En el desarrollo de la aplicación para Android también surgieron problemas relacionados, principalmente, con la versión del marco de trabajo IONIC y la de los módulos utilizados, que en algunas ocasiones presentaban problemas de compatibilidad. La solución a este problema fue cambiar de la versión 3 de IONIC a la versión 4.

Otro aspecto que supuso un problema fue la integración dentro de la aplicación de una cámara. La intención era realizar las grabaciones desde ella, evitando tener que recurrir a aplicaciones de terceros. Finalmente, esta función tuvo que ser descartada debido a que el comienzo de la grabación no coincidía con el comienzo de la toma de datos, lo que provocaba que no estuvieran sincronizados. Esto complicaba la comparación de los datos obtenidos con el video de la prueba. Esto se solventó recurriendo a una aplicación externa llamada “Timestamp Camera Free” [46].

Estos y otros problemas menores fueron resueltos y se pudo dar fin al proyecto.

11. CONCLUSIONES

Tras completar todas las fases que conlleva el desarrollo del dispositivo Arduino y de la aplicación para Android se puede concluir que, aunque el hardware del dispositivo tiene un coste inferior que el del teléfono móvil y que el dispositivo tiene como única tarea la detección del estado de la calzada (dedica el 100% de su vida útil a ello), la aplicación para teléfonos móviles desempeña mejor la solución al problema planteado en este proyecto. A continuación, se puede observar una lista de las características que influyen en esta afirmación:

- En comparación con el dispositivo Arduino, el teléfono móvil dispone de componentes *hardware* de mejor calidad.
- El teléfono móvil presenta un mejor *rendimiento* a la hora de ejecutar el código creado puesto que posee mayor capacidad de cómputo y de memoria.
- Si fuera necesario interconectar el dispositivo o la aplicación con otro sistema sería más sencillo realizarlo con un teléfono móvil ya que presenta una mejor *conectividad* (posibilidad de conectarse a internet y a otros dispositivos mediante Wifi, comunicación por Bluetooth o por NFC (*Near Field Communication*) entre otras opciones).
- El hardware del teléfono móvil presenta una menor *probabilidad de error*, comparado con el hardware del dispositivo, a la hora de desarrollar software sobre él puesto que ha sido probado y cumple con varios estándares de calidad.
- Desde el punto de vista software, el desarrollo de la aplicación sobre un *framework*, con una gran comunidad de desarrolladores que colaboran en su desarrollo, permite que este sea relativamente sencillo y que el tiempo invertido en la solución de los problemas que puedan surgir sea menor.
- En el momento de realizar una *toma de datos* es más sencillo en la aplicación móvil que en el dispositivo Arduino. En el primero solamente se debe abrir la aplicación y pulsar un botón, mientras que en el dispositivo Arduino se requiere de una fuente de alimentación, del uso de una tarjeta microSD y se debe esperar por la conexión del sensor GPS para poder comenzar.

Sin embargo, aunque a priori parezca que el dispositivo Arduino está en desventaja, se debe tener en cuenta que se trata de un prototipo ensamblado sin ningún tipo de soldadura y con componentes diseñados para el prototipado. Para poder realizar una valoración en igualdad de condiciones habría que construir el sistema sobre una placa de circuito impreso.

A nivel personal ha sido altamente enriquecedor el haber desarrollado este proyecto porque me ha permitido comprender en profundidad todas las fases por las que pasa un proyecto informático y todos los aspectos que se deben tener en cuenta a la hora de desarrollar un producto. Este trabajo me ha ayudado también a mejorar la redacción de documentos técnicos y científicos.

Para finalizar, me gustaría mencionar que ha sido gratificante comprobar como lo que en un principio era poco más que una idea finalmente se convirtió en un producto funcional. Además, me quedo con que independientemente de los imprevistos que puedan surgir durante un desarrollo siempre existe una solución al problema.

12. TRABAJO FUTURO

Aunque el proyecto que se presenta junto con esta memoria cumple con lo establecido en un principio, puede seguir trabajándose, añadiendo mejoras y nuevas funcionalidades. A continuación, se hace una descripción de algunas ideas que pueden ser implementadas en un futuro:

Uno de los puntos de este proyecto que podría ser evolucionado en un futuro es el tratamiento de los datos obtenidos por los sensores, principalmente los del dispositivo Arduino. En la actualidad esta información solamente se almacena en un archivo en texto plano y no es utilizada de ninguna manera. Un posible tratamiento podría ser un sistema que, de manera automática, plasmara la información sobre un mapa siguiendo un código de colores. Con esto se lograría tener una imagen en la que se juntan los datos del acelerómetro y los datos de posición del sensor GPS. El usuario, al visualizar el mapa, sabría rápidamente cuales son las zonas que requieren más atención. A continuación, se muestra una simulación de un mapa creado a partir de una captura de pantalla de la web “OpenStreetMap” [94] y que ha sido editado posteriormente:

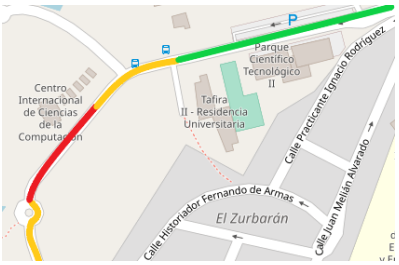


Ilustración 63 – Ejemplo de mapa personalizado.

Para lograr esto se puede recurrir a librerías de JavaScript como “Leaflet” [95], que provee de una serie de funciones que permiten crear mapas totalmente personalizables.

Otra posible mejora es la sustitución de la placa *Arduino UNO Rev3* por otra de mayor capacidad como puede ser la placa *Arduino Mega 2560*, consiguiendo con ello un mayor número de conexiones y un aumento de la memoria Flash y SRAM. Con esto último se podría implementar nuevas funcionalidades como el control de varios sensores o el envío de datos a través de una red wifi.

También es positivo realizar un cambio de la antena del sensor GPS para mejor la conexión con el sistema GPS y lograr así un mejor rendimiento del dispositivo en general.

13. BIBLIOGRAFÍA

[1] I. Por, E. L. Mal, and E. D. E. La, “Estudio percepción del estado de las carreteras - Nota de prensa,” 2019. [Online]. Available: https://www.aecarretera.com/sala-de-prensa/comunicados/comunicados-2019/2985-mas-de-la-mitad-de-los-usuarios-reconoce-haber-sufrido-algun-incidente-por-el-mal-estado-de-la-via.

[2] Abc, “El mal estado del asfalto provoca costosas averías en los coches,” 2019. [Online]. Available: https://www.abc.es/motor/reportajes/abci-estado-asfalto-provoca-costosas-averias-coches-201903260221_noticia.html. [Accessed: 10-Jun-2019].

[3] J. J. Castillo Aguilar, J. A. Cabrera Carrillo, A. J. Guerra Fernández, and E. Carabias Acosta, “Robust Road Condition Detection System Using In-Vehicle Standard Sensors.,” *Sensors (Basel, Switzerland)*, 19-Dec-2015. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/26703605. [Accessed: 09-Oct-2019].

[4] Sensovant, “Sensor de Calzada TCS.” [Online]. Available: http://sensovant.com/productos/meteorologia/pluviometria-hielo/estado-calzada/sensor-inteligente-calzada-TCS.html. [Accessed: 20-Sep-2019].

[5] Sensovat, “Sensor de Asfalto WSS.” [Online]. Available: http://sensovant.com/productos/meteorologia/pluviometria-hielo/estado-calzada/sensor-inteligente-asfalto-WSS.html. [Accessed: 20-Sep-2019].

[6] Darrera, “Medidor de Estado de la Calzada RCM411 | Darrera.” [Online]. Available: https://www.darrera.com/wp/es/producto/rcm411-medidor-estado-calzada/. [Accessed: 20-Sep-2019].

[7] Darrera, “Sensor de Estado de la Calzada M201 | Darrera.” [Online]. Available: https://www.darrera.com/wp/es/producto/m201-sensor-estado-calzada/. [Accessed: 20-Sep-2019].

[8] Darrera, “Sensor Óptico de Estado de la Calzada M251 | Darrera.” [Online]. Available: https://www.darrera.com/wp/es/producto/m251-sensor-optico-estado-calzada/. [Accessed: 20-Sep-2019].

[9] Dilus, “Sensores de asfalto y calzada | Dilus.” [Online]. Available: http://dilus.es/es/blog/los-sensores-de-asfalto-y-calzada/. [Accessed: 20-Sep-2019].

[10] Boschung, “Active Pavement Sensor | IT-Arctis | Boschung America.” [Online]. Available: https://boschungamerica.com/products/pavement-sensors/it-arctis. [Accessed: 20-Sep-2019].

[11] Boschung, “Passive Pavement Sensor | IT-Sens | Boschung America.” [Online]. Available: https://boschungamerica.com/products/pavement-sensors/it-sens. [Accessed: 20-Sep-2019].

[12] R. Bajwa, R. Rajagopal, E. Coleri, P. Varaiya, and C. Flores, “In-pavement wireless weigh-in-motion,” *Proceedings of the 12th international conference on Information processing in sensor networks - IPSN ’13*, 2013. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2461381.2461397. [Accessed: 20-Sep-2019].

[13] X. Yu and E. Salari, “Pavement pothole detection and severity measurement using laser imaging,” *2011 IEEE INTERNATIONAL CONFERENCE ON ELECTRO/INFORMATION TECHNOLOGY*, May-2011. [Online]. Available: http://ieeexplore.ieee.org/document/5978573/. [Accessed: 20-Sep-2019].

[14] I. Schiopu, J. P. Saarinen, L. Kettunen, and I. Tabus, “Pothole detection and tracking in car video sequence,” *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, Jun-2016. [Online]. Available: http://ieeexplore.ieee.org/document/7760975/. [Accessed: 20-Sep-2019].

[15] A. Dhiman and R. Klette, “Pothole Detection Using Computer Vision and Learning,” *IEEE Transactions on Intelligent Transportation Systems*, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8788687/. [Accessed: 20-Sep-2019].

[16] P. M. Harikrishnan and V. P. Gopi, “Vehicle Vibration Signal Processing for Road Surface Monitoring,” *IEEE Sensors Journal*, 15-Aug-2017. [Online]. Available: https://ieeexplore.ieee.org/document/7959046/. [Accessed: 20-Sep-2019].

[17] keuwlsoft, “Accelerometer Meter - Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.keuwl.accelerometer. [Accessed: 20-Sep-2019].

[18] ItGeeks, “Full system info - Aplicaciones en Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=itgeeks.fullsysteminfo. [Accessed: 20-Sep-2019].

[19] Yasiru Nayanajith, “Información del Dispositivo - Aplicaciones en Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.ytheekshana.deviceinfo. [Accessed: 20-Sep-2019].

[20] Wered Software, “Sensores Multiherramienta - Aplicaciones en Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.wered.sensorsmultitool. [Accessed: 20-Sep-2019].

[21] EXA Tools, “Sensores Multiherramienta - Aplicaciones en Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.exatools.sensors. [Accessed: 20-Sep-2019].

[22] AyudaLeyProtecciónDatos, “Normativa Protección de Datos en Aplicaciones - geolocalización,” 2016. [Online]. Available: https://ayudaleyprotecciondatos.es/2016/06/06/normativa-lopd-aplicaciones-moviles/#Aplicaciones_de_geolocalizacion. [Accessed: 20-Sep-2019].

[23] Agencia Española de protección de datos, “Guía sobre el uso de videocámaras para seguridad y otras finalidades.” [Online]. Available: https://www.aepd.es/media/guias/guia-videovigilancia.pdf. [Accessed: 20-Sep-2019].

[24] N. Vollmer, “Artículo 89 UE Reglamento general de protección de datos,” 05-Sep-2018. [Online]. Available: http://www.privacy-regulation.eu/es/89.htm. [Accessed: 21-Sep-2019].

[25] P. Ágiles, “Desarrollo iterativo e incremental – Proyectos Ágiles.” [Online]. Available: https://proyectosagiles.org/desarrollo-iterativo-incremental/. [Accessed: 12-Sep-2019].

[26] Git, “Git - git Documentation.” [Online]. Available: https://git-scm.com/docs/git. [Accessed: 24-May-2019].

[27] Wikipedia, “Git,” 2019. [Online]. Available: https://es.wikipedia.org/wiki/Git. [Accessed: 24-May-2019].

[28] Git Kraken, “Free Git GUI Client - Windows, Mac & Linux | GitKraken.” [Online]. Available: https://www.gitkraken.com/. [Accessed: 09-Oct-2019].

[29] Atlassian, “Bitbucket: What is Bitbucket? - Atlassian Documentation.” [Online]. Available: https://confluence.atlassian.com/confeval/development-tools-evaluator-resources/bitbucket/bitbucket-what-is-bitbucket. [Accessed: 24-May-2019].

[30] Wikipedia, “Bitbucket,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Bitbucket. [Accessed: 24-May-2019].

[31] Wikipedia, “Arduino IDE,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Arduino_IDE. [Accessed: 28-May-2019].

[32] Arduino, “Arduino - Libraries.” [Online]. Available:

https://www.arduino.cc/en/Reference/Libraries. [Accessed: 19-Sep-2019].

[33] M. Schwartz, “LiquidCrystal I2C - Arduino Libraries.” [Online]. Available: https://www.arduinolibraries.info/libraries/liquid-crystal-i2-c. [Accessed: 19-Sep-2019].

[34] JetBrains, “WebStorm: The Smartest JavaScript IDE by JetBrains,” 2019. [Online]. Available: https://www.jetbrains.com/webstorm/. [Accessed: 28-May-2019].

[35] Npm, “About npm | npm Documentation.” [Online]. Available: https://docs.npmjs.com/about-npm/. [Accessed: 28-May-2019].

[36] W3schools, “What is npm.” [Online]. Available: https://www.w3schools.com/whatis/whatis_npm.asp. [Accessed: 28-May-2019].

[37] Wikipedia, “npm (software),” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Npm_(software). [Accessed: 28-May-2019].

[38] GitHub, “nvm,” 2010. [Online]. Available: https://github.com/nvm-sh/nvm. [Accessed: 28-May-2019].

[39] Coreybutler, “nvm for Windows,” 2014. [Online]. Available: https://github.com/coreybutler/nvm-windows. [Accessed: 28-May-2019].

[40] Fritzing, “Fritzing.” [Online]. Available: http://fritzing.org/home/. [Accessed: 28-May-2019].

[41] “Fritzing | Aprendiendo Arduino,” 2015. [Online]. Available: https://aprendiendoarduino.wordpress.com/category/fritzing/. [Accessed: 28-May-2019].

[42] Winworldpc, “WinWorld: Microsoft Word 1.x (Windows).” [Online]. Available: https://winworldpc.com/product/microsoft-word/1x. [Accessed: 19-Sep-2019].

[43] winworldpc, “WinWorld: Microsoft Word 1.x (DOS).” [Online]. Available: https://winworldpc.com/product/microsoft-word/1x-dos. [Accessed: 19-Sep-2019].

[44] Wikipedia, “Office Open XML (docx) - Microsoft Office Word.” [Online]. Available: https://en.wikipedia.org/wiki/Office_Open_XML. [Accessed: 19-Sep-2019].

[45] Wikipedia, “Formato DOC - Microsoft Office Word.” [Online]. Available: https://en.wikipedia.org/wiki/Doc_(computing). [Accessed: 19-Sep-2019].

[46] Yubin Chen, “Timestamp Camera Free - Aplicaciones en Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.jeyluta.timestampcamerafree. [Accessed: 20-Sep-2019].

[47] Arduino, “Arduino Uno Rev3.” [Online]. Available: https://store.arduino.cc/arduino-uno-rev3. [Accessed: 29-May-2019].

[48] oomlout, “Protoboard Image,” 2009. [Online]. Available: https://flic.kr/p/6wXjoB. [Accessed: 29-May-2019].

[49] I. Inc., “MPU-6000 and MPU-6050 Product Specification,” *InvenSense Inc. Product Specification*, 2013. [Online]. Available: https://www.cdiweb.com/datasheets/invensense/MPU-6050_DataSheet_V3 4.pdf.

[50] E. V Board and U. Guide, “MPU-6000/MPU-6050 9-Axis Evaluation Board User Guide,” 2011. [Online]. Available: www.invensense.com.

[51] B. Ave, “MPU-6000 and MPU-6050 Register,” 2013. [Online]. Available: www.inversense.com.

[52] Luis Llamas, “Determinar la orientación con Arduino y el IMU MPU-6050,” 2016. [Online].

Available: https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/. [Accessed: 28-Feb-2019].

[53] PNT, “GPS.gov: El Sistema de Posicionamiento Global.” [Online]. Available: https://www.gps.gov/systems/gps/spanish.php. [Accessed: 07-Sep-2019].

[54] Techmake, “Introducción al módulo GPS NEO-6M.” [Online]. Available: http://www.techmake.com/intro-gps-neo. [Accessed: 07-Sep-2019].

[55] Luis Llamas, “Localización GPS con Arduino y los módulos GPS NEO-6.” [Online]. Available: https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/. [Accessed: 10-Oct-2019].

[56] JHD, “SPECIFICATION OF LCD MODULE.” [Online]. Available: https://www.arduino.cc/documents/datasheets/LCDscreen.PDF. [Accessed: 28-Feb-2019].

[57] Naylampmechatronics, “Tutorial LCD con I2C, controla un LCD con solo dos pines.” [Online]. Available: https://www.naylampmechatronics.com/blog/35_Tutorial--LCD-con-I2C-controla-un-LCD-con-so.html. [Accessed: 28-Feb-2019].

[58] Kingston Technology, “microSD Canvas Select - kingston technology.” [Online]. Available: https://www.kingston.com/es/memory-cards/canvas-select-microsd-card. [Accessed: 06-Sep-2019].

[59] Luis Llamas, “Leer y escribir en una tarjeta SD o micro SD con Arduino.” [Online]. Available: https://www.luisllamas.es/tarjeta-micro-sd-arduino/. [Accessed: 10-Oct-2019].

[60] Mountain Switch, “Tactile Switches,” 2005. [Online]. Available: https://www.arduino.cc/documents/datasheets/Button.pdf. [Accessed: 29-Jun-2019].

[61] L. Del Valle Hernández, “Resistencia pull up y pull down con Arduino, para qué sirven.” [Online]. Available: https://programarfacil.com/blog/arduino-blog/resistencia-pull-up-y-pull-down/. [Accessed: 09-Oct-2019].

[62] Wikipedia, “I2C,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/I^2C#Reference_design. [Accessed: 29-Jun-2019].

[63] Circuits Basics, “Basics of the I2C communication protocol,” 2016. [Online]. Available: http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/. [Accessed: 29-Jun-2019].

[64] Circuit Basics, “Basics of UART Communication.” [Online]. Available: http://www.circuitbasics.com/basics-uart-communication/. [Accessed: 29-Jun-2019].

[65] Sparkfun, “UART - Serial Communication.” [Online]. Available: https://learn.sparkfun.com/tutorials/serial-communication/uarts. [Accessed: 29-Jun-2019].

[66] Sparkfun, “Serial Peripheral Interface (SPI) - learn.sparkfun.com.” [Online]. Available: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all. [Accessed: 29-Jun-2019].

[67] Wikipedia, “SPI,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface.

[68] Wikipedia, “Node.js,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Node.js#Overview.

[69] Android, “Conoce Android Studio | Android Developers.” [Online]. Available: https://developer.android.com/studio/intro. [Accessed: 30-Jun-2019].

[70] Wikipedia, “Android Studio,” 2019. [Online]. Available:

https://es.wikipedia.org/wiki/Android_Studio. [Accessed: 30-Jun-2019].

[71] Wikipedia, “Java Development Kit,” 2019. [Online]. Available: https://en.wikipedia.org/wiki/Java_Development_Kit. [Accessed: 30-Jun-2019].

[72] Wikibooks, “Arduino Lenguaje de Programación,” 2018. [Online]. Available: https://es.wikibooks.org/wiki/Lenguaje_de_programación_Arduino. [Accessed: 03-Jul-2019].

[73] Ionic Framework, “Ionic Framework - Ionic Documentation.” [Online]. Available: https://ionicframework.com/docs. [Accessed: 09-Oct-2019].

[74] I. Framework, “Ionic Native Community Edition - Ionic Documentation.” [Online]. Available: https://ionicframework.com/docs/native/overview. [Accessed: 03-Jul-2019].

[75] Apache, “Apache Project Information.” [Online]. Available: https://projects.apache.org/project.html?cordova. [Accessed: 19-Sep-2019].

[76] Apache, “Architectural overview of Cordova platform - Apache Cordova.” [Online]. Available: https://cordova.apache.org/docs/en/latest/guide/overview/index.html. [Accessed: 03-Jul-2019].

[77] Wikipedia, “Angular framework.” [Online]. Available: https://es.wikipedia.org/wiki/Angular_(framework). [Accessed: 19-Sep-2019].

[78] Angular, “Angular.” [Online]. Available: https://angular.io/. [Accessed: 19-Sep-2019].

[79] GitHub, “GitHub - angular/angular: One framework. Mobile & desktop.” [Online]. Available: https://github.com/angular/angular. [Accessed: 19-Sep-2019].

[80] M. Torres Búa, “El método de proyectos en tecnología.,” 2014. [Online]. Available: https://www.edu.xunta.es/espazoAbalar/sites/espazoAbalar/files/datos/1464945204/contido/1_fase_definicin_y_anlisis_del_problema.html. [Accessed: 10-Jun-2019].

[81] “Sensor tilt SW-520D.” [Online]. Available: https://www.luisllamas.es/medir-inclinacion-con-arduino-y-sensor-tilt-sw-520d/. [Accessed: 25-Sep-2019].

[82] A. Peña, “Ingeniería de Requisitos. 2 Temario Definiciones Requisitos Funcionales y No Funcionales.” [Online]. Available: https://slideplayer.es/slide/13974699/. [Accessed: 12-Sep-2019].

[83] F. Ruiz, P. López, and F. de ciencias Universidad de Cantabria, “Ingeniería del Software - Contexto y requisitos del sistema.” [Online]. Available: https://www.ctr.unican.es/asignaturas/is1/is1-t08-trans.pdf. [Accessed: 12-Sep-2019].

[84] Universidad de Salamanca, Gríal, F. J. García Peñalvo, and A. García Holgado, “FUNDAMENTOS DE LA VISTA DE CASOS DE USO.” [Online]. Available: https://repositorio.grial.eu/bitstream/grial/1155/1/UML - Casos de uso.pdf. [Accessed: 22-Sep-2019].

[85] E. Mediavilla, “II Modelos y herramientas UML.” [Online]. Available: https://www.ctr.unican.es/asignaturas/MC_OO/Doc/Casos_de_uso.pdf. [Accessed: 22-Sep-2019].

[86] Ionic Framework, “File - Ionic Documentation.” [Online]. Available: https://ionicframework.com/docs/native/file. [Accessed: 19-Sep-2019].

[87] Ionic Framework, “File Opener - Ionic Documentation.” [Online]. Available: https://ionicframework.com/docs/native/file-opener. [Accessed: 19-Sep-2019].

[88] Ionic-team and GitHub, “GitHub - ionic-team/ionic-native-google-maps: Google maps plugin for Ionic Native.” [Online]. Available: https://github.com/ionic-team/ionic-native-google-maps.

[Accessed: 19-Sep-2019].

[89] Mapsplugin and GitHub, “GitHub - mapsplugin/cordova-plugin-googlemaps: Google Maps plugin for Cordova.” [Online]. Available: https://github.com/mapsplugin/cordova-plugin-googlemaps. [Accessed: 19-Sep-2019].

[90] Google, “Overview | Maps SDK for Android | Google Developers.” [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/intro. [Accessed: 19-Sep-2019].

[91] Ionic-team and GitHub, “LocationService - ionic-team/ionic-native-google-maps · GitHub.” [Online]. Available: https://github.com/ionic-team/ionic-native-google-maps/blob/master/documents/locationservice/README.md. [Accessed: 30-Sep-2019].

[92] Ionic Framework, “Sensors - Ionic Documentation.” [Online]. Available: https://ionicframework.com/docs/native/sensors. [Accessed: 30-Sep-2019].

[93] Google, “Google Charts | Google Developers.” [Online]. Available: https://developers.google.com/chart. [Accessed: 30-Sep-2019].

[94] Colaboradores de OpenStreetMap, “OpenStreetMap.” [Online]. Available: https://www.openstreetmap.org/#map=17/28.07764/-15.45103. [Accessed: 01-Oct-2019].

[95] Leaflet, “Leaflet - a JavaScript library for interactive maps.” [Online]. Available: https://leafletjs.com/. [Accessed: 01-Oct-2019].

14. ANEXO: MANUAL DE USUARIO

En este apartado se describe brevemente como utilizar las diferentes funciones del dispositivo Arduino y de la aplicación para Android desde el punto de vista del usuario:

14.1. DISPOSITIVO ARDUINO

Encender el dispositivo: Para encender el dispositivo es necesario disponer de un transformador AC/DC de 9 Voltios y una toma de corriente o de un cable USB (de tipo A en un extremo y de tipo B en el otro) y un ordenador. El usuario simplemente debe conectar la placa Arduino UNO Rev3 utilizando el transformador o el cable USB. Una vez hecho, el dispositivo se encenderá y mostrará un mensaje de bienvenida.

No se debe alimentar la placa Arduino UNO Rev3 con un voltaje superior a 12 voltios. En caso de que esto ocurra, el dispositivo se puede ver dañado.

Apagar el dispositivo: Para apagar el dispositivo de manera controlada y segura se debe hacer uso del botón que se encuentra conectado en la placa de prototipado. Al pulsar este botón se ejecutarán las operaciones de apagado evitando que se produzcan daños en los componentes hardware y en la información almacenada en la tarjeta microSD. Tras la pulsación, se mostrará en la pantalla LCD un mensaje de despedida. En ese momento se podrá desconectar el dispositivo de la corriente eléctrica sin que se produzcan daños.

Consultar la información en la pantalla LCD: La pantalla LCD permite al usuario ver los datos que se están tomando en tiempo real del sensor acelerómetro y del sensor GPS. Además, sirve como indicador del correcto funcionamiento del dispositivo. Por ejemplo, si no se muestran datos, se observan caracteres aleatorios o hay cambios bruscos en la intensidad lumínica de la pantalla, puede ser indicador de un posible problema.

Insertar la tarjeta microSD: Para almacenar la información del sensor acelerómetro y del sensor GPS es necesario disponer de una tarjeta microSD. Esta debe estar formateada en el sistema de archivos FAT16 o FAT32. El primer paso para insertar la tarjeta microSD es comprobar que el dispositivo está apagado. En caso afirmativo, se debe conectar la tarjeta en el adaptador que se encuentra en la placa de prototipado. Una vez hecho, el usuario puede hacer uso normal del dispositivo.

Extraer la tarjeta microSD: Para extraer la tarjeta microSD del dispositivo primero se debe comprobar que este se encuentra apagado. Seguidamente, se puede extraer la tarjeta del adaptador de microSD conectado a la placa de prototipado.

14.2. APLICACIÓN ANDROID

Ver la lista de archivos de datos: En la pantalla principal de la aplicación se puede observar la lista de archivos de datos correspondientes a las tomas de datos.

Abrir un archivo de datos: En la lista de archivos de datos se debe buscar el archivo que se desea abrir. Al pulsar sobre él se mostrará un menú con varias aplicaciones para visualizar el archivo. Seguidamente, al seleccionar una de estas opciones, se mostrará el contenido del archivo.

Ver el gráfico de un archivo de datos: Para poder visualizar una gráfica con los datos del acelerómetro de una toma de datos en concreto, se debe seleccionar un elemento de la lista de archivos y desplazarlo hacía

la izquierda. Tras esta acción, aparecerá un nuevo botón que permite acceder a la vista de la gráfica. Para poder visualizar esta información correctamente se debe girar 90º el teléfono móvil.

Comenzar una toma de datos: En la esquina inferior derecha de la pantalla principal se puede observar un botón con el icono de "empezar" (▶). Al pulsar sobre él se inicia una toma de datos. Previamente se debe encender el sensor GPS desde las opciones del teléfono móvil.

Finalizar una toma de datos: Para finalizar una toma de datos se debe pulsar el botón "pausar" (⏸) que se encuentra en la esquina inferior derecha.