ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

ORDENADOR DE A BORDO PARA BICICLETAS USANDO ARDUINO

ON-BOARD COMPUTER FOR BICYCLES USING ARDUINO

Realizado por
Joaquín Trillo Escribano
Tutorizado por
Daniel Garrido Márquez
Departamento
Lenguaje y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, septiembre 2017

Fecha defensa:
El Secretario del Tribunal

Resumen

Este Trabajo de Fin de Grado describe el desarrollo de un ordenador de a bordo para bicicletas.

El sistema está formado por varios componentes. En primer lugar, por un subsistema hardware basado en microcontroladores Arduino que, haciendo uso de varios sensores (el fundamental, un sensor de efecto hall), es capaz de mostrar en una pequeña pantalla LCD información relativa al estado actual de un trayecto en bicicleta. Dicho dispositivo se fija en el manillar del vehículo para que el ciclista pueda ver con facilidad y, de un simple vistazo, la distancia recorrida, el tiempo que lleva empleado en la ruta, la velocidad en ese preciso instante y la temperatura ambiente actual.

Por otro lado, existe una aplicación móvil que recibe estos datos del subsistema hardware cuando termina la ruta, a través de Bluetooth.

Además de ver los detalles de cada ruta, la aplicación permite eliminar cada una de estas rutas de forma permanente y almacenar la longitud de la rueda de la bicicleta usada, medida necesaria para que el ordenador de a bordo realice sus cálculos.

Palabras clave

Bicicleta, rueda, ruta, ordenador de a bordo, dispositivo, Arduino, efecto Hall, Android, aplicación móvil, Bluetooth.

Abstract

This Final Degree Project describes the development of an on-board computer for bicycles.

The system is composed of several components. First of all, a hardware subsystem based on Arduino microcontrollers that, making use of some sensors (the most important, a Hall effect sensor), can display on a small LCD screen information relating to a bicycle route's current state. That device is fixed on the handlebar of the vehicle so that the rider can see at a single glance the distance travelled, the time spent in that ride, the speed at that moment and the ambient temperature.

On the other hand, we also have a mobile application which receives that data from the hardware subsystem when the route ends, through Bluetooth.

In addition to see the details of each route, the app allows to remove each one of these routes permanently and store the length of the wheel of the bicycle used. This measure is necessary for the on-board computer to perform its calculations.

Keywords

Bicycle, wheel, route, on-board computer, device, Arduino, Hall effect, Android, mobile app, Bluetooth.

ÍNDICE

CAPÍTULO 1 | INTRODUCCIÓN

Es bien sabido por todos que cada vez más personas usan una bicicleta como medio de transporte. Algunos la utilizan para hacer ejercicio, otros para dar un paseo por la naturaleza. Asimismo, están los que montan en bici para evitar la contaminación implícita de coches o motos, y los hay también que prefieren usar este vehículo para evitar atascos gracias a los cada vez más frecuentes carriles bicis [1] [2] [3].

Lo que está claro es que el número de ciclistas, por una razón u otra, está en claro aumento. Y, al igual que los conductores de vehículos de motor de combustión interna, desean saber a qué velocidad van, cuanto tiempo llevan circulando o la distancia que llevan recorrida.

Por ello surge la idea de crear un ordenador de a bordo para bicicletas, gracias a la cual este gran número de usuarios verán cumplidos los deseos mencionados en el párrafo anterior.

Si bien es cierto que ya existen dispositivos comerciales similares, el que se documenta en esta memoria se diferencia de estos en el almacenamiento de los datos de la ruta llevada a cabo.

Siendo más concretos, el sistema está formado por dos elementos principales. Por un lado, tenemos un dispositivo hardware, situado en el manillar de la bicicleta, encargado de realizar mediciones y de mostrarlas por pantalla. Por el otro, una aplicación para *smartphones* en la que, al finalizar cada ruta, se guardarán los datos de cada uno de los trayectos realizados para su posterior consulta.

El subsistema hardware (la pieza clave del sistema) basa su funcionamiento en el efecto Hall [4]. En otras palabras, en un radio de la bici se sitúa un imán y en la horquilla un sensor de efecto Hall a la misma altura para que, cuando el sensor detecte el imán, se cuente un giro de la rueda.

1.1 | Motivación

Con este sistema aquí descrito he pretendido poner en práctica los conocimientos que adquirí el pasado verano en desarrollo de aplicaciones Android y en el manejo de Arduino y sus componentes y sensores, todo ello junto con todo lo aprendido durante mis años de estancia en la Universidad de Málaga. Y que mejor modo que aunar las posibilidades de estas tecnologías a una de mis grandes aficiones, el deporte.

Además, la memoria está pensada para que cualquiera que disponga de los componentes, que más adelante se describen, y posea conocimientos en Android y Arduino pueda hacer también este dispositivo o el sistema completo, siguiendo los capítulos 3, 4 y 5.

Por último, comentar que con este ordenador de a bordo se ha pretendido hacer algo con más posibilidades y flexible que el resto de dispositivos comerciales existentes, ya que los posibles cambios y ampliaciones en el subsistema hardware abren un amplio abanico de configuraciones del mismo. En las conclusiones de la memoria se profundiza en estas posibles modificaciones y mejoras del dispositivo.

1.2 | Objetivos

El principal objetivo de este TFG ha sido desarrollar un sistema que no solo permita al ciclista conocer la información básica de su ruta en bici, sino, además, almacenar cada uno de estos paseos para su posterior consulta.

Concretamente se han marcado los siguientes tres objetivos:

- **Desarrollo de un dispositivo mediante Arduino**, el cual se coloca en el manillar de la bicicleta e indicará a través de una pequeña pantalla LCD los datos del trayecto actual.

- **Desarrollo de una aplicación Android** para presentar el listado de rutas realizadas. En ella se permite ver en detalle la información de cada uno de los viajes llevados a cabo, se pueden eliminar estos trayectos y se almacena la longitud de la rueda de la bicicleta en la que se va a usar este ordenador de a bordo. Esta última utilidad de la aplicación es fundamental para el correcto funcionamiento del sistema.

- **Implementación de la comunicación entre el dispositivo y la aplicación.** Se ha optado por usar una conexión Bluetooth para comunicar las dos partes del sistema.

1.3 | Estructura de la memoria

En el Capítulo 2 se estudian algunos proyectos o trabajos similares, equiparando sus ventajas e inconvenientes respecto al que se ha desarrollado en esta memoria, además de realizar una comparativa con algunos productos comerciales. También se da una breve descripción de las distintas tecnologías que se han usado tanto en el desarrollo de la aplicación Android como en el del subsistema hardware.

El desarrollo del dispositivo, la explicación de sus componentes y pruebas del dispositivo aislado del resto del sistema se estudian en el Capítulo 3 de la memoria.

En el Capítulo 4, se describe el desarrollo de la aplicación Android, así como su compatibilidad con móviles, sus casos de uso y arquitectura de la misma.

La implementación de la comunicación entre el subsistema hardware y la aplicación para smartphones, además de pruebas del correcto funcionamiento de todo el sistema, se estudia en el Capítulo 5.

En el sexto y último capítulo, se explica, en forma de conclusiones, cómo surgió la idea para este proyecto y qué ha supuesto la realización del mismo en lo que a nuevos conocimientos se refiere. También se exponen una serie de líneas y trabajos futuros que se podrían realizar para mejorar y completar aún más el sistema.

Por último, en el Anexo se encuentran los manuales de instalación y de uso del sistema, tanto del dispositivo hardware como de la aplicación Android.

CAPÍTULO 2 | ESTADO DEL ARTE Y TECNOLOGÍAS UTILIZADAS

A lo largo de este capítulo se analizan una serie de trabajos similares al que se ha desarrollado, comentando ventajas e inconvenientes de cada uno de ellos con respecto a nuestro ordenador de a bordo. En la segunda parte de este capítulo se presentan brevemente las tecnologías que se han utilizado para construir el sistema.

2.1 | Estado del arte

En primer lugar, debemos diferenciar entre trabajos realizados con fines académicos o bien lúdicos, y productos pensados para su posterior comercialización.

Dentro del primer grupo de sistemas nos encontramos con un proyecto en la web de IEEE Spectrum, *Build a Readable Bicycle Computer* [5]. Consiste en crear un ordenador de a bordo para una bicicleta usando una Raspberry Pi y un libro Kindle. Al igual que en mi dispositivo, se coloca un imán en un radio para controlar la cadencia de pedaleo, pero en lugar de tener un pequeño dispositivo con una pantalla LCD, se usa un Kindle de Amazon, de un tamaño mucho mayor.

Figura 1: Ordenador de a bordo con Raspberry Pi y un libro Kindle.

Esto se debe a que su autor, David Schneider, necesitaba un dispositivo en el cual ver perfectamente los datos de sus escapadas en bicis, porque según él a su edad ya no le valían los pequeños equipos que se venden normalmente.

También existe un proyecto realizado por alumnos de la Universidad Rey Juan Carlos de Madrid para la asignatura Sistemas Empotrados y de Tiempo Real, *Ordenador de a bordo (visualización de velocidad, distancia, temperatura…)* [6].

Figura 2: Ordenador de a bordo para coches.

En este caso, la diferencia es clara, el medio de transporte en el que se usa, aunque el diseño del dispositivo es similar al que se ha desarrollado en este TFG. Otro aspecto en el que no coinciden es en el uso de una pantalla táctil en lugar de una LCD, inviable la primera en este proyecto por motivos de seguridad (el tener que pulsar la pantalla para ver los diferentes datos en lugar de que estos vayan apareciendo periódicamente puede provocar distracciones en el ciclista, con su correspondiente riesgo de accidente).

Y por otro lado tenemos una amplia gama de productos comerciales. En el escalón más bajo encontramos productos más sencillos, como el *Cuentakilómetros B'TWIN 100* [7] que, de forma similar a nuestro dispositivo, sitúa un sensor en un radio de la rueda para calcular el giro de la misma. Su precio de 9,99 euros es muy atractivo, pero su funcionalidad limitada.

En un peldaño intermedio se sitúan los cuentakilómetros como los de la *gama TOPLINE 2012 de la marca Sigma* [8], que poseen más funciones que los anteriores (temperatura, altitud o porcentajes de pendiente) y su precio se

encuentra entre los 25 euros y más de 70, dependiendo de la cantidad de funciones de medición que posean. Tanto este grupo como el anterior no cuentan con un registro de trayectos, cosa que nuestro sistema si.

Por último, nos encontramos con los llamados ciclo-computadores, potentes máquinas con una amplia gama de funcionalidades (GPS, cuenta-calorías, medición de altitud mediante barómetro o alertas de llamadas y mensajes al móvil) además de que disponen de registros de rutas y trayectos realizados. La conocida firma *Garmin* [9] es un buen ejemplo de estos ciclo-computadores, cuyos precios se encuentran en un rango desde los 130 hasta los 550.

2.2 | Tecnologías usadas

Un sistema como el que se ha desarrollado requiere de la utilización de múltiples tecnologías tanto hardware como software que lleven a cabo todas y cada una de las funcionalidades.

El dispositivo físico está basado íntegramente en Arduino, la popular plataforma de desarrollo de proyectos electrónicos, utilizando además diferentes componentes tales como diversos tipos de sensores, resistencias o una matriz LED.

En cuanto a la aplicación móvil, en ella convergen varias tecnologías. Para la construcción de las diferentes vistas, la generación dinámica de las mismas y la lógica y el back-end de la aplicación se ha utilizado Android. Además, se ha utilizado una base de datos SQLite para proporcionar soporte de almacenamiento al sistema. La elección de SQLite se debe a que es el sistema de gestión de bases de datos relacional incluido en Android, por lo que está elección no ha sido voluntaria.

Por último, se ha optado por comunicar ambos dispositivos con Bluetooth, ya que es una tecnología gratuita, sencilla y que al ser inalámbrica evita el uso de cables.

2.2.1 | Arduino

Arduino [10] [11] [12] es una plataforma de desarrollo electrónico basada en un hardware y software fáciles de usar. Las placas Arduino (el hardware) son capaces de leer entradas – como la luz de un sensor, la pulsación de un botón, o un mensaje de Twitter – y convertirlo en una salida – activar un motor, encender/apagar un LED, publicar algo en Internet. Se le puede decir a la placa qué hacer enviándole una serie de instrucciones al microcontrolador del dispositivo. Para hacer esto se usa el lenguaje de programación de Arduino y su entorno de desarrollo (el software).

Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios. Toda la plataforma, incluyendo sus componentes hardware (esquemáticos) y software, son liberados con licencia de código abierto que permite libertad de acceso a ellos.

El hardware consiste en una placa de circuito impreso con un microcontrolador, puertos digitales y analógicos de entrada/salida, los cuales pueden conectarse a placas de expansión (shields), que amplían características de funcionamiento de la placa. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación con el computador.

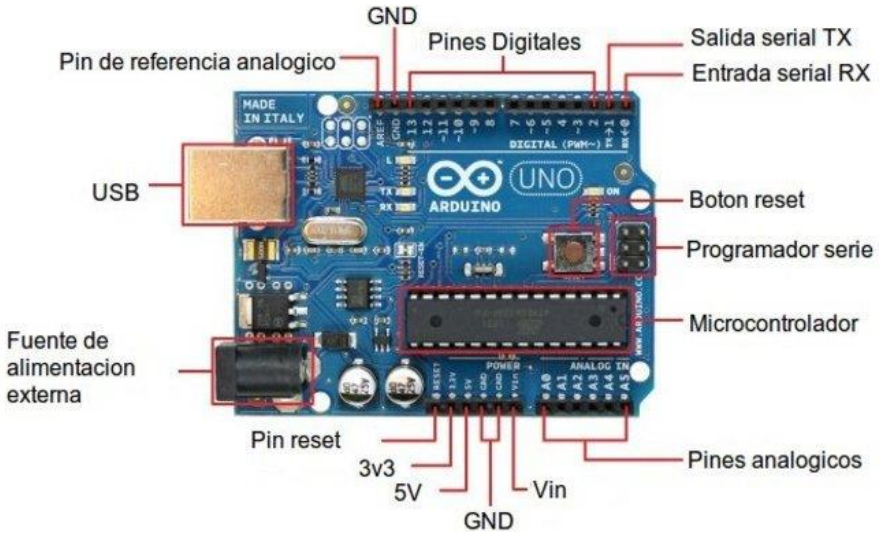


Figura 3: Conexiones de la placa Arduino UNO.

Conexiones de la placa

Como ya se ha mencionado, Arduino cuenta con una serie de entradas y salidas para aumentar su funcionalidad. En la Figura 3 se observan todas las conexiones de la placa Arduino UNO (la placa usada en el proyecto), basada en el microcontrolador ATmega328. A continuación, se describen brevemente las más relevantes:

- **Alimentación:** puede alimentarse mediante la conexión USB que proporciona 5 V, o a través del puerto Jack de alimentación, mediante una fuente de alimentación de entre 6 y 20 V, aunque se recomienda que sean de entre 7 y 12 V. En nuestro caso, se ha optado por utilizar una pila de 9 voltios.

- **Pines de alimentación:**
  - **Vin:** proporciona la tensión máxima con la que está alimentada la placa.
  - **GND:** toma de tierra.
  - **5 V:** proporciona una tensión de salida de 5 V y una intensidad máxima de 300 mA.
  - **3.3 V:** proporciona una tensión de salida de 3.3 V y una intensidad máxima de 50 mA.

- **Pines de entrada y de salida:** la placa cuenta con 14 pines digitales y 6 analógicos, y la intensidad máxima de todos estos es de 40 mA. En estos pines se conectan los diferentes componentes que amplían la funcionalidad de la placa.
  - **Analógicos**: los valores tanto de entrada como de salida van desde 0 a 5 voltios, en un rango de 0 a 1023 en la entrada (precisión de 10 bits) y de 0 a 255 en la salida (8 bits).
  - **Digitales:** pueden suministrar o aceptar (según el modo de trabajo en el que el pin este configurado) entre 0 y 5 V. Además, 6 de estos pines proporcionan PWM.

Por otro lado, el software consiste en un entorno de desarrollo y lenguaje de programación (este último derivado de C), así como el cargador de arranque (bootloader) que es ejecutado en la placa. El microcontrolador de la placa se programa mediante un computador, usando una comunicación serial.

2.2.2 | Android

Android [13] es un sistema operativo de código abierto basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como smartphones y tablets, aunque actualmente también se usa en relojes inteligentes, televisores y automóviles.

Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde compró.

Entre sus características más importantes destacan las siguientes:

- Adaptable a muchas pantallas y resoluciones.
- Utiliza SQLite para el almacenamiento de datos.
- Ofrece diferentes formas de mensajería.
- Navegador web basado en WebKit incluido.
- Soporte de Java y muchos formatos multimedia.
- Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software.
- Bluetooth.
- Multitarea.

Los componentes básicos de toda aplicación Android son las Actividades. Una *Activity* [14] es un componente que contiene una pantalla o vista con la que los usuarios pueden interactuar para realizar una acción, como marcar un número de teléfono, tomar una foto, enviar un correo electrónico o ver un mapa. A cada actividad se le asigna una ventana en la que diseña su interfaz de usuario.

Una aplicación consiste generalmente en múltiples actividades vinculadas entre sí. Normalmente, una de ella se especifica como la actividad principal que se presenta al usuario cuando este inicia la aplicación. Cada actividad a su vez

puede iniciar otra actividad para poder realizar diferentes acciones. Cuando se inicia una nueva actividad, se detiene la anterior, pero el sistema conserva la actividad en la **pila de actividades**. En el momento en el que el usuario termina de interactuar con la actividad actual y presiona el botón *Atrás*, se elimina de la pila (se destruye) y se reanuda la actividad anterior. Todo este ciclo de vida de las actividades Android queda mejor explicado en la Figura 4.

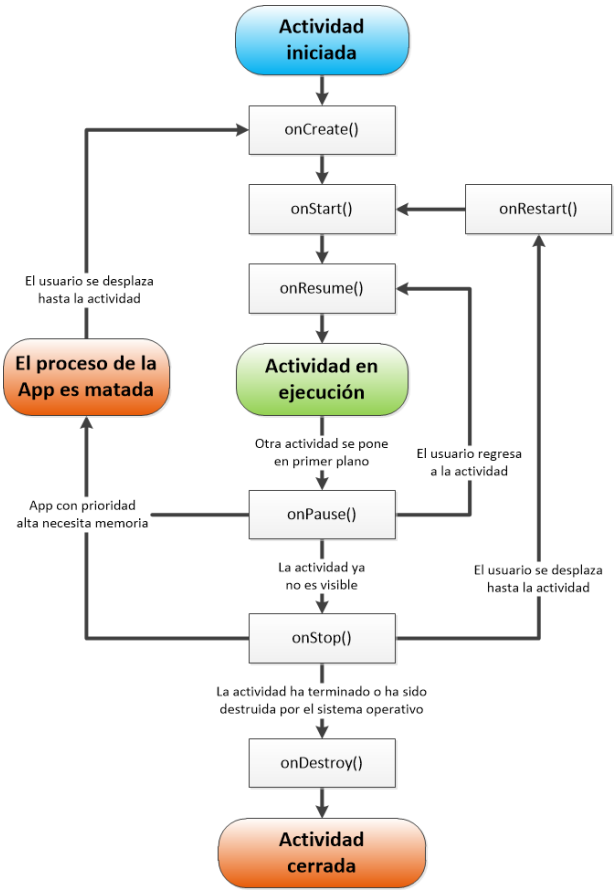


Figura 4: Ciclo de vida de las actividades de Android.

Además, cabe destacar que Android es el sistema operativo que acapara el mayor número de ventas en el mundo. Según datos de la consultora Kantar Worldpanel [15], en los meses de febrero, marzo y abril de este año la cuota de mercado de Android en los principales mercados europeos es del 78,3 por ciento, mientras que en Estados Unidos y China las ventas de teléfonos inteligentes con este sistema operativo son del 61,7 y del 83,4 por ciento, respectivamente. En cuanto a España, la cuota de mercado de Android asciende hasta el 92 por ciento, prácticamente el monopolio en nuestro país.

2.2.3 | SQLite

SQLite [16] [17] es una pequeña biblioteca escrita en C que implementa un motor de base de datos SQL autocontenido, sin servidor, sin configuración y transaccional.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones, lo cual reduce notablemente la latencia en el acceso a la base datos.

El conjunto de la base de datos (definiciones, tablas, índices y los propios datos), son almacenados como un solo fichero en el dispositivo – en este caso en el móvil, soportando bases de datos de hasta 2 Terabytes.

Debido a su pequeño tamaño, SQLite es muy adecuado para los sistemas integrados, y, aparte de en Android, también está incluido en BlackBerry, Windows Phone, Google Chrome o iOS.

2.2.4 | Bluetooth

Bluetooth [18] [19] [20] es la especificación de un protocolo de comunicación inalámbrica inventada por la compañía Ericsson en 1994 y que posteriormente se estandarizó como 802.15.1

Permite la conexión de dispositivos variados (teléfonos, portátiles, cámaras, impresoras, …) en distancias cortas mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz, reemplazando cables de conexión.

Cada dispositivo tiene una dirección única de 48 bits (6 bytes). Adicionalmente es posible darle un nombre que nos permita reconocerlo con mayor facilidad del resto.

El alcance depende de muchos factores: antena, obstáculos, potencia de transmisión o la línea de visión. En condiciones normales el rango esperado va

de decenas de metros a un kilómetro, aunque realmente va desde pocos centímetros a, como mucho, 100 metros.

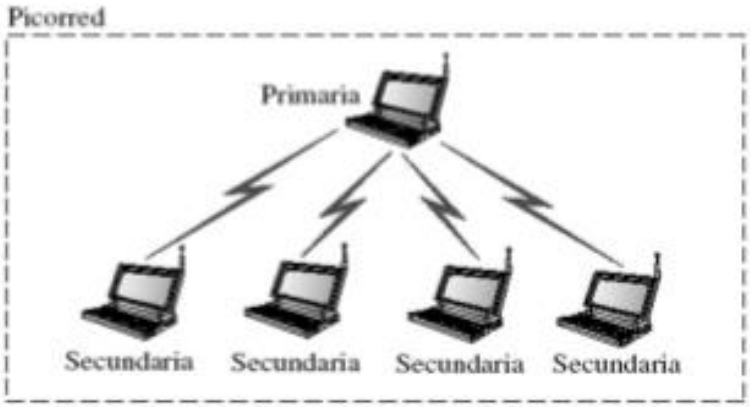


Figura 5: Arquitectura picorred.

Estos dispositivos se encuentran unos a otros constituyendo una red *ad hoc*, o red inalámbrica descentralizada. Esta red que forman se denomina *piconet* o picorred, en las que existe hasta un máximo de 8 estaciones, donde una de ellas actúa de primaria y el resto son secundarias. En la Figura 5 se puede observar un pequeño ejemplo de una picorred.

CAPÍTULO 3 | EL DISPOSITIVO FÍSICO

En este tercer capítulo se describe el subsistema hardware por completo, sus componentes, conexiones entre ellos, el software y los distintos estados por los que puede pasar.

Este dispositivo es la pieza clave del sistema, ya que es el calcula la velocidad, distancia recorrida y tiempo transcurrido del trayecto, además de la temperatura actual.

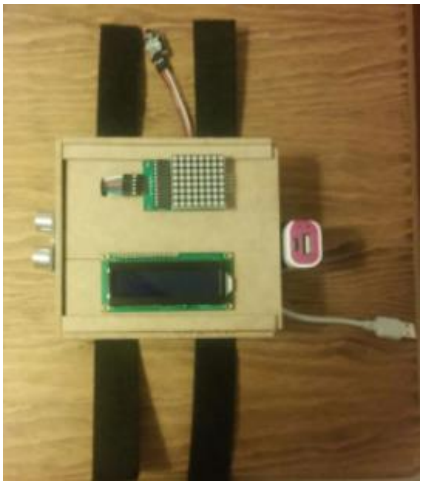


Figura 6: Ordenador de a bordo con todos sus componentes en la carcasa.

3.1 | Componentes del dispositivo

Para el desarrollo del dispositivo se han utilizado diferentes módulos y componentes para aumentar la funcionalidad de la placa Arduino. La elección de los mismos ha facilitado desde un primer momento el desarrollo del subsistema hardware dada la sencillez de sus conexiones e integración con la placa principal. A continuación, se describen todos los elementos usados y su función.

3.1.1 | Placa Arduino UNO

En el capítulo 2 ya se ha explicado con detalle las conexiones de la placa. En resumidas cuentas, es el componente principal del dispositivo al que se conectan los diferentes módulos y sensores. Además, contiene el programa que desempeña la funcionalidad principal.

3.1.2 | Pantalla LCD

Este componente permite al usuario visualizar en todo momento los ya mencionados datos relacionados con el estado actual del trayecto. Consta de 2 filas de 16 columnas, para un total de 32 caracteres.

Figura 7: Pantalla LCD 16x2.

Como se puede observar en la Figura 8, se incluye un potenciómetro de hasta 10.000 ohmios para regular el brillo de la pantalla.

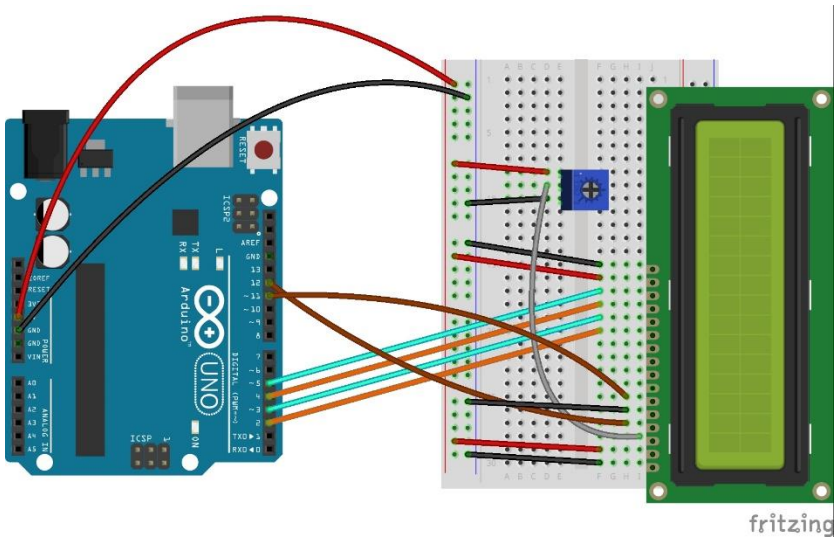


Figura 8: Esquema visual de las conexiones (sólo pantalla).

Conexiones

La pantalla tiene 16 pines, que se usan para dar corriente y para que la placa sepa qué escribir en la pantalla.

La clavija de selección de registro (**SR**) controla dónde aparecerán los caracteres en la pantalla. El pin de lectura/escritura (**L/E**) pone la pantalla en modo de lectura o escritura (en este proyecto usaremos el modo de escritura). La clavija de habilitación (**H**) le dice al LCD que va a recibir una instrucción. Los pines de datos (**D0-D7**) se usan para enviar datos de caracteres a la pantalla, aunque solo se utilizarán 4 de ellos (**D4-D7**) [21].

Las conexiones (Tabla 1) se sitúan en una placa en la parte trasera de la pantalla, aunque sólo se usarán 12 de estos 16 pines (Figura 9).

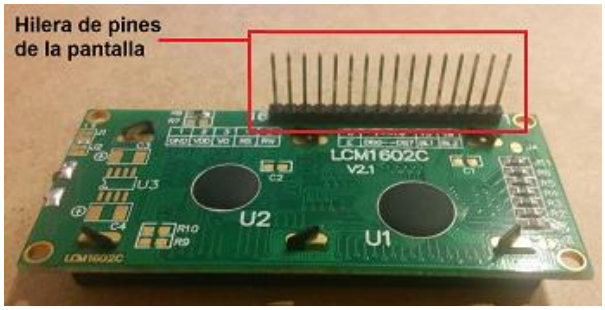


Figura 9: Conexiones de la pantalla LCD.

Pantalla	Arduino
Vss	GND
LED-	
L/E	
Vcc	5 V
LED+	
D7	Pin 5
D6	Pin 4
D5	Pin 3
D4	Pin 2
V0	Potenciómetro
RS	6
H	7

Tabla 1: Resumen de conexiones pantalla-Arduino.

Prueba del circuito

Se hizo una pequeña prueba para comprobar el correcto funcionamiento de la pantalla y ver cuál era el ajuste de brillo más adecuado.

Como se puede apreciar en las Figuras 10 y 11, la primera opción es la más óptima para una mejor visualización de la pantalla, por lo que el brillo queda fijado al máximo. De hecho, si no el brillo no está al máximo es difícil leer la pantalla a plena luz del sol

Figura 10: Pantalla con el brillo al máximo.

Figura 11: Pantalla con el brillo a medias.

3.1.3 | Sensor de Temperatura

Gracias a este componente podemos conocer la temperatura actual del lugar en el que este sensor se encuentre.

Figura 12: Sensor de temperatura

Para ello antes hay que realizar un par de conversiones previas. En primer lugar, hay que convertir la lectura del sensor a voltaje (**voltaje** = (**lectura**/1024.0) * 5.0) y este a grados centígrados (**grados** = (**voltaje** – 0.5) * 100) [22].

Conexiones

El sensor de temperatura consta de 3 patillas (Figura 13), la número 1 para alimentar el sensor, la central para la salida, y la número 3 para la toma de tierra. La Tabla 2 muestra el resumen de conexiones entre el sensor y los pines de la placa Arduino.

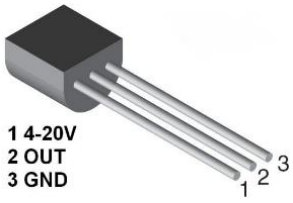


Figura 13: Conexiones del sensor de temperatura

Sensor	**Arduino**
4-20 V	5 V
OUT	Pin A0
GND	GND

Tabla 2: Resumen de conexiones sensor-Arduino.

3.1.4 | Sensor de Efecto Hall

Es el componente más importante del dispositivo, ya que nos permite conocer el número de revoluciones que da la bicicleta gracias a que detecta campos magnéticos (cada vez que detecta el imán, se contabiliza una vuelta de la rueda).

Figura 14: Sensor de efecto Hall

Conexiones

El sensor de efecto Hall consta de 3 patillas (Figura 15), una para alimentar el sensor, la central para la toma de tierra, y la otra patilla lateral para la salida. La Tabla 3 muestra el resumen de conexiones entre el sensor y los pines de la placa Arduino.

Figura 15: Conexiones del sensor de efecto Hall

Sensor	Arduino
Vcc	5 V
OUT	Pin 8
GND	GND

Tabla 3: Resumen de conexiones sensor-Arduino.

Prueba del circuito

A continuación, se muestra el funcionamiento del sensor a través de las Figuras 16, 17, 18 y 19.

Figuras 16 y 17: Si el sensor no detecta un imán, el contador no se aumenta.

Figuras 18 y 19: Si le acercamos un imán al sensor, el contador se aumenta.

3.1.5 | Sensor de Ultrasonidos

Este componente permite conocer la distancia a la que un objeto está de este sensor. Su funcionamiento se basa en emitir un pitido y medir el tiempo que la señal tarda en regresar [23]. En el dispositivo actúa de interruptor, cuando el usuario acerca la mano a él a una distancia determinada. Para el ciclista es más fácil y seguro acercar la mano al señor que pulsar un botón en mitad de la marcha, por eso se escogió este componente en lugar de un pulsador.

Figura 20: Sensor de ultrasonidos.

Conexiones

El sensor de ultrasonidos cuenta con 4 patillas (Figura 21), la roja para alimentar el sensor, la celeste para la señal de entrada de 10 microsegundos, la naranja para la señal de salida (eco), y la patilla negra para la toma de tierra. En la Tabla 4 se puede observar el resumen de conexiones entre el ultrasonido y la placa.

Ultrasonido	**Arduino**
Vcc	5 V
Trig	Pin 12
Echo	Pin 13
GND	GND

Figura 21: Conexiones del sensor de ultrasonidos

Tabla 4: Resumen de conexiones ultrasonido-Arduino.

3.1.6 | Matriz LED

Este componente es únicamente visual. En él se presentarán diferentes figuras que servirán de señales al usuario antes y al terminar la ruta (no durante la misma). Consta de 8 filas de 8 columnas cada una, por lo que tenemos un total de 64 leds.

Figura 22: Matriz LED 8x8.

Conexiones

La matriz cuenta con 10 pines (Figura 23), 5 para alimentar la matriz y para comunicar con la placa, y otros 5 pines de ampliación que sirven para concatenar otra u otras matrices a esta primera. La Tabla 5 muestra el resumen de conexiones entre el sensor y los pines de la placa Arduino.

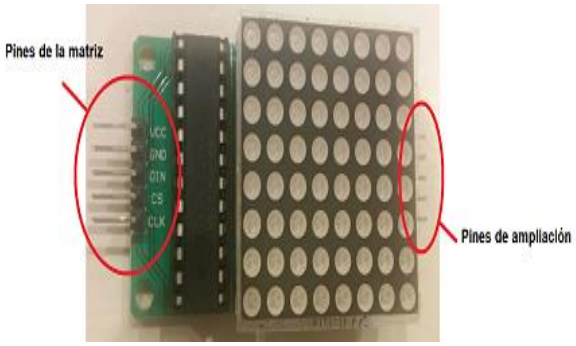


Figura 23: Conexiones de la matriz LED

Matriz	**Arduino**
VCC	5 V
GND	GND
DIN	Pin 11
CS	Pin 10
CLK	Pin 9

Tabla 5: Resumen de conexiones matriz-Arduino.

Prueba del circuito

Las figuras 24 y 25 muestras algunas de las señales que se le aparecerán al usuario antes y después del trayecto en bici.

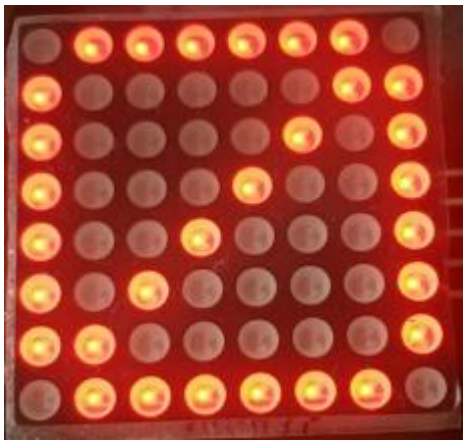


Figura 24: Señal para parar.

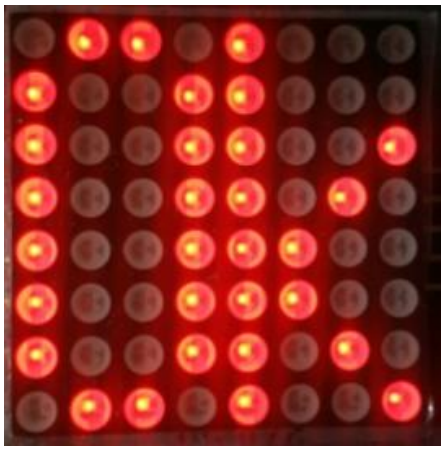


Figura 25: Señal de OK.

3.1.7 | Fuente de alimentación

Lo habitual a la hora de alimentar la placa Arduino es, mediante un cable, conectar su puerto USB a un ordenador. Pero en este proyecto no nos sirve cualquier tipo de fuente de alimentación. Esta tiene que ser de un tamaño reducido, portátil y debe tener suficiente carga eléctrica para poder alimentar la placa y todos sus componentes [24].

En un primer momento se pensó en utilizar una pila de 9V conectándola al puerto Jack de la placa, pero este tipo de fuente de alimentación tienen una densidad energética muy baja, entre 500-600 mAh. Por ello se ha optado por usar un pequeño banco de energía portátil de 2600 mAh, que sí que satisface las necesidades energéticas del dispositivo. Se conecta por el puerto USB del Arduino.

Figura 26: Fuente de alimentación.

3.1.8 | Otros componentes

Los componentes restantes (Figuras 27-29) se encargan de interconectar los diferentes componentes a la placa principal.

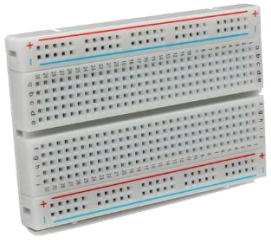


Figura 27: Protoboard.

Figura 28: Imanes de neodimio.

Figura 29: Potenciómetro.

3.2 | Dispositivo completo

Una vez han sido explicados los componentes utilizados para formar el dispositivo, queda exponer cómo todos ellos interaccionan entre sí para que el ordenador de a bordo funcione correctamente. En la Figura 30 se puede observar cómo han quedado las conexiones en el circuito.

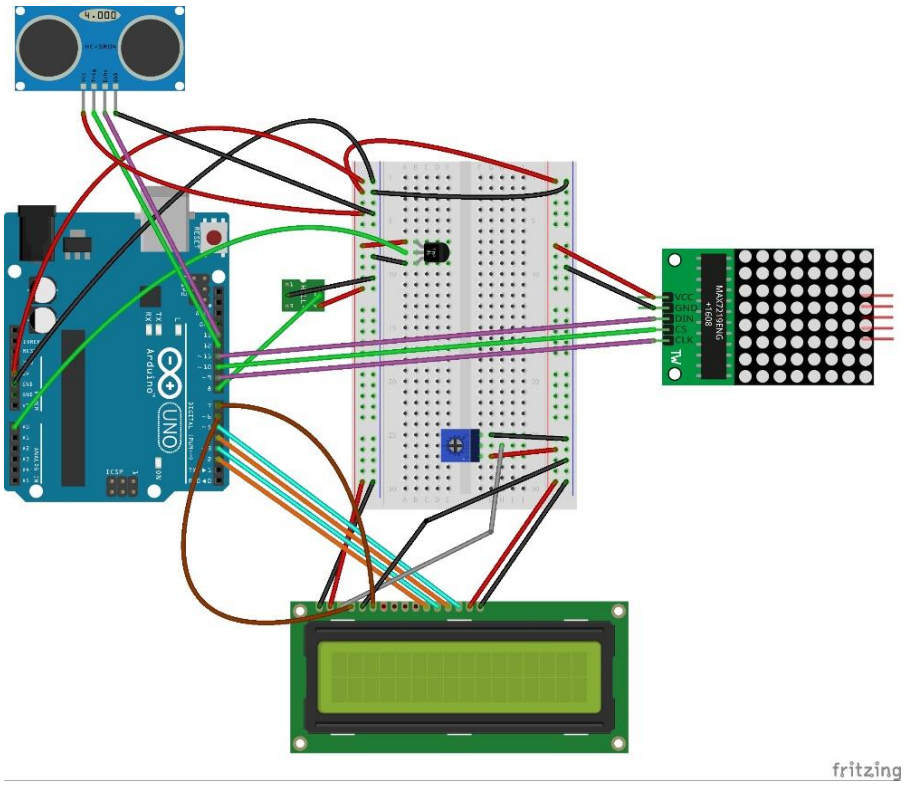


Figura 30: Esquema visual de las conexiones (dispositivo completo).

3.2.1 | Casos de uso

En el dispositivo solo existen tres posibles acciones realizables por el usuario: empezar la ruta, terminarla, y volver al estado inicial del ordenador de a bordo. La Figura 31 muestra con mayor claridad los casos de uso del dispositivo.

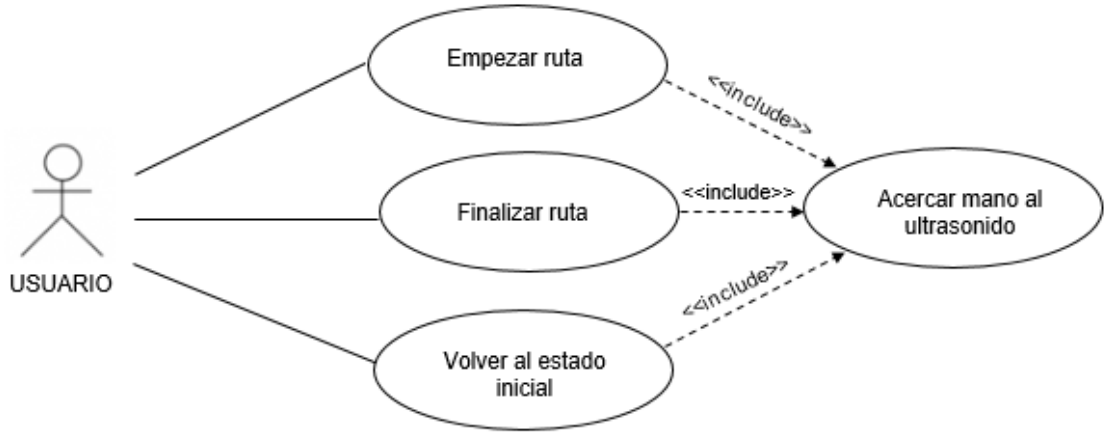


Figura 31: Casos de uso del dispositivo.

3.2.2 | Estados del dispositivo

La máquina de estados del sistema hardware está formada por varios estados y por las transiciones entre estos. Como se mencionó en el apartado anterior, el sensor de ultrasonidos actúa de interruptor en el dispositivo, ya que para el ciclista es más cómodo acercar su mano al ultrasonido que pulsar un pequeño botón durante la marcha. Además, de esta forma se evitan posibles distracciones que puedan desencadenar en un accidente.

Según el estado en el que el sistema hardware se encuentre, la matriz LED y la pantalla LCD mostraran al usuario una señal y un mensaje diferentes, respectivamente.

En la Figura 32 se puede observar el diagrama de estados del dispositivo.

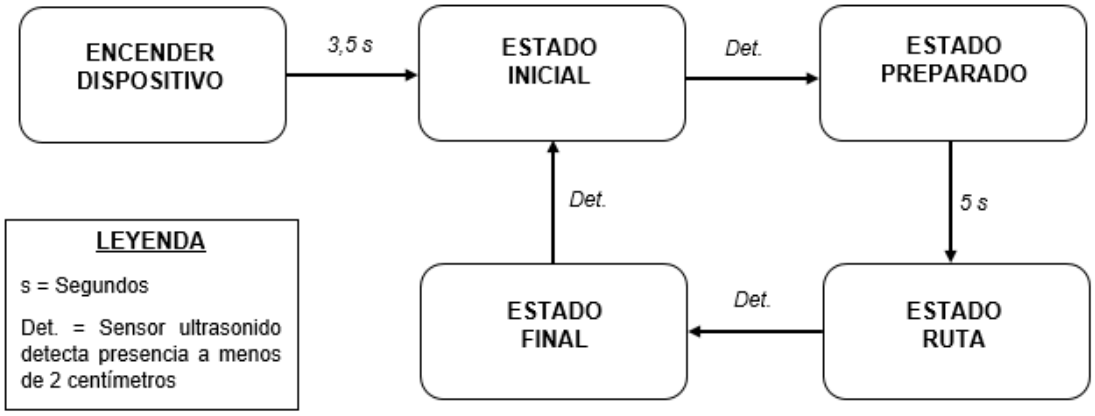


Figura 32: Diagrama de estados del dispositivo.

A continuación, se van a explicar brevemente cada uno de los estados y sus transiciones:

- **Encender dispositivo:** nada más encender el dispositivo nos aparecerá un mensaje de bienvenida en el LCD. A los 3 segundos y medio, pasará al *Estado Inicial*.

- **Estado Inicial:** la matriz LED muestra una señal de OK y la pantalla le sugiere al usuario que acerque la mano al ultrasonido a menos de dos centímetros para comenzar la ruta. Si este lo hace, el dispositivo transitará al *Estado Preparado*.
- **Estado Preparado:** el LCD advierte al ciclista que se prepare para comenzar el trayecto, mientras que la matriz hace una cuenta atrás empezando en 5. Al terminar esta cuenta, se pasa al estado básico del sistema, el *Estado Ruta*.

- **Estado Ruta:** Durante los primeros 3 segundos y medio, la matriz LED mostrará una flecha hacia delante parpadeando. Por su parte la pantalla irá alternando cada 5 segundos los datos a mostrar. Primero veremos la velocidad y temperatura actuales, y luego la distancia recorrida y el tiempo que llevamos en ruta. Para finalizar la ruta y transitar al *Estado Final*, se deberá acercar la mano al sensor de ultrasonidos a menos de 2 centímetros.

- **Estado Final:** al igual que en el *Estado Ruta*, en el LCD se verán de forma alterna los datos finales del trayecto, mientras que la matriz LED mostrará una señal de parar. Si el usuario acerca la mano a la distancia ya mencionada, volverá al *Estado Inicial*.

Si se quiere apagar el dispositivo, hay que desconectar la fuente de alimentación de la placa.

3.2.3 | Coste total del dispositivo

En este apartado se aborda una parte fundamental en el desarrollo de cualquier proyecto, el tema económico.

Para la realización del proyecto se ha necesitado hacer un pequeño desembolso para la adquisición de los componentes del sistema hardware.

Aunque si disponía de algunos elementos (placa Arduino Uno, pantalla LCD, potenciómetro, sensor de temperatura y algunos cables) en la Tabla 6, aparece el coste de todos los componentes utilizados para la realización del ordenador de a bordo.

Placa Arduino Uno	19,00 €
Pantalla LCD 16x2	9,90 €
Sensor de temperatura	1,77 €
Sensor de efecto Hall	1,95 €
Sensor de ultrasonidos HC-SR04	1,95 €
Matriz LED MAX7219	3,91 €
Imanes de neodimio	3,91 €
Cables	5,86 €
Potenciómetro 10k Ω	0,48 €
Batería Externa 2600 mAh	6,53 €
TOTAL	**55,26 €**

Tabla 6: Coste de los componentes del dispositivo.

El dispositivo en su totalidad tendría un coste total de unos 55 euros, de los cuales habría que descontar unos 5 de los cables e imanes sobrantes, por lo que el precio del ordenador de a bordo sería aproximadamente de unos 50 euros, en

el que se incluirían este dispositivo y la aplicación que se describirá en el Capítulo 4.

Como se comentó en el Capítulo 2, existen otros dispositivos más baratos, pero este permite más funcionalidades que estos. Además, cuando el usuario no esté usando el ordenador de a bordo, puede utilizar la batería externa para cargar su móvil o alimentar cualquier otro tipo de aparato electrónico.

3.2.4 | Software utilizado

Una vez definida toda la parte hardware del dispositivo, pasamos a describir el software que le permite desempeñar su función.

El entorno de desarrollo Arduino IDE 1.8.1 ha sido el utilizado para el desarrollo del código, el cual se divide en 4 partes fundamentales. Dichos módulos son los que siguen:

- **Control de la pantalla LCD:** para el control de la pantalla se ha requerido de la utilización de la librería *LiquidCrystal*, que viene con el software de Arduino. Maneja todo el trabajo de escritura, y simplifica el proceso de redactar el código para visualizar los caracteres.
  Para inicializar la librería hay que indicarle al constructor qué pines va a usar para comunicarse (*LiquidCrystal lcd (Selección de registro, Habilitación, pines de datos)*, inicialización de la librería).

- **Control del sensor de ultrasonidos:** para obtener la distancia en centímetros del objeto detectado por el sensor hay que realizar unas conversiones. La fórmula es la que sigue:

$$distancia\ (cm) = \frac{tiempo\ entre\ pulsos\ (\mu s) * 0.034\ \left(\frac{\mathrm{cm}}{\mu s}\right)}{2}$$

  0.034 cm/µs es la velocidad del sonido en el aire y se divide entre dos porque solo nos interesa la distancia de ida.

- **Actualización de la velocidad, tiempo y distancia recorrida:** Una vez el comienza el estado ruta, el contador de revoluciones de la rueda y el temporizador se reinician. La distancia se calcula multiplicando el número de giros por la longitud de la rueda. Para el tiempo se resta el instante actual con el instante en que se comenzó el trayecto. Por último, para actualizar la velocidad se divide la distancia que se avanza en 10 revoluciones entre el tiempo que se ha tardado en dar esos 10 giros.

- **Control de la matriz LED:** al igual que para la pantalla LCD, se requiere del uso de una librería, concretamente la librería *LedControl* [25]. A diferencia de la librería de control de pantallas LCD, esta no está incluida en Arduino, por lo que tenemos que instalarla. En la Figura 33 se puede apreciar la búsqueda a realizar en el Gestor de Librerías de Arduino IDE.

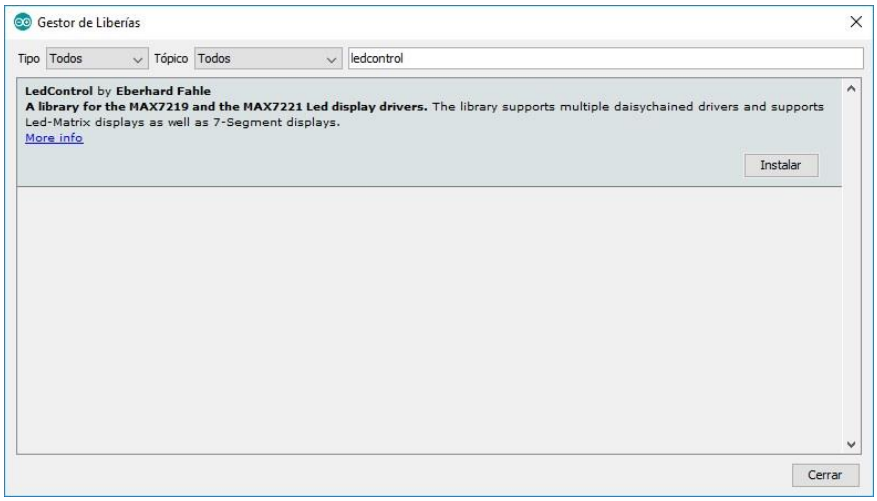


Figura 33: Instalación de la librería LedControl, en el Gestor de Librerías de Arduino IDE.

Los parámetros que el constructor necesita para controlar la matriz son los 3 pines de esta y el número de matrices a usar, (se pueden concatenar matrices para aumentar el tamaño de la figura a mostrar) en este proyecto una.

CAPÍTULO 4 | LA APLICACIÓN ANDROID

En este capítulo se abordará el desarrollo de la aplicación Android, sus casos de uso y su arquitectura. El nombre de la app es **OnBoardStats** (del inglés *onboard* = a bordo y *stats* = abreviación de estadísticas) y es compatible con Android 4.1, también conocido como Jelly Bean, y versiones posteriores a esta. En la Figura 34 se puede observar el logo diseñado.

Figura 34: Logo de la app Android desarrollada.

4.1 | Casos de uso

La descripción de la funcionalidad está descrita por los diferentes casos de uso que el usuario pueda darle al sistema.

El primero de todos es la visualización de los datos de cualquiera de las rutas realizadas. Como ya se ha comentado anteriormente, estos son la fecha en la que se llevó a cabo el trayecto, la hora de comienzo, la duración de la ruta, la velocidad media y la distancia recorrida en la misma.

Por otro lado, el usuario tendrá acceso al listado de rutas, en donde podrá visualizar la información de cada una de ellas o bien eliminarlas una a una si lo desea.

Por último, podrá guardar y modificar la medida de la longitud de la rueda de su bicicleta. Esta medición deberá indicarse en milímetros.

En la Figura 35 se puede observar un esquema con los casos de uso de la aplicación.

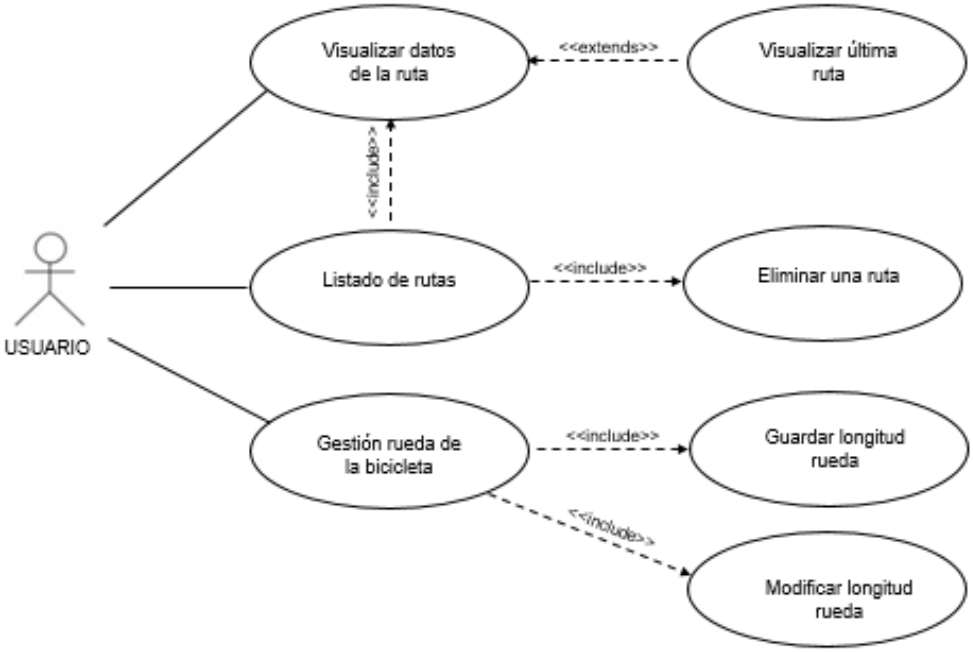


Figura 35: Casos de uso de la aplicación.

4.2 | Arquitectura de la aplicación

Lo primero a la hora de diseñar la aplicación es elegir su arquitectura. Esta se escogerá según el tipo de requisitos y funcionalidad de dicha app.

En este caso se ha optado por un modelo en tres capas, en la que cada una de ellas lleva a cabo una función y ofrece servicios a la capa inmediatamente superior. Se puede ver con más claridad la comunicación entre las capas en la Figura 36.

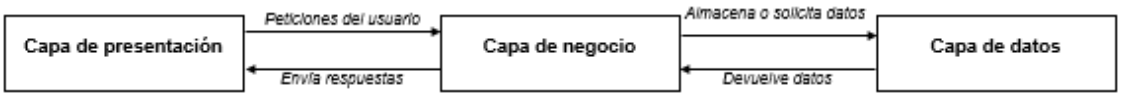


Figura 36: Arquitectura de la aplicación móvil.

4.2.1 | Capa de presentación

Esta primera capa es la que ve el usuario, le presenta la aplicación, le comunica la información y captura las acciones del usuario. Contiene las vistas de la aplicación, que solicitan acciones a la capa de negocio mediante el envío de un formulario o la pulsación de un botón.

A continuación, se describen brevemente las vistas desarrolladas, todas ellas en formato XML [26]:

- **main.xml (Figura 38):** es la vista principal de la aplicación, la primera que le aparece al usuario cuando abre la app. Desde esta vista se puede navegar al resto de ellas.

- **route_detail.xml (Figura 39):** en esta vista se muestran todos los detalles (fecha, hora de inicio, duración, velocidad media y distancia recorrida) de la última ruta realizada o bien de la ruta seleccionada por el usuario del listado de trayectos almacenados.

- **routes.xml (Figura 40):** aquí se le presentan al usuario todas las rutas almacenadas, cada una de ellas identificada por su fecha y hora de inicio. Se pueden consultar más datos sobre estos trayectos o bien se pueden eliminar uno a uno.

- **edit_wheel.xml (Figura 41):** esta vista sirve para guardar o modificar la longitud de la rueda de la bici de la que se quieren obtener datos durante una ruta. Además, si se conoce el marcado internacional de la rueda, existe la opción de convertir esta medida a la longitud del neumático. En el manual de usuario se amplía la información sobre el marcado de las ruedas de bicicletas y sus distintas variantes.

También se ha editado el fichero **styles.xml** para cambiar los colores del encabezado de las vistas, el color de los mensajes de las ventanas emergentes y el estilo de los botones de la aplicación. Para estos últimos además se han creado los ficheros **boton_normal.xml**, **botón_pulsado.xml** y

botón_deshabilitado.xml¸ todos ellos englobados en **botones.xml**, donde se gestiona cuál de estos tres ficheros usar según el estado del elemento (habilitado, pulsado o deshabilitado).

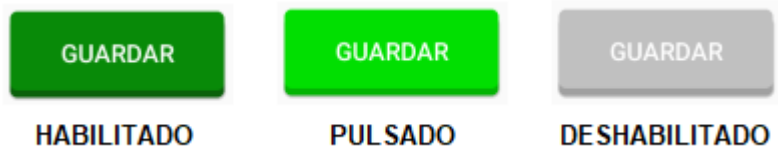


Figura 37: Estilo de los botones según su estado.

En las Figuras 38, 39, 40 y 41 se puede ver el diagrama de interacción entre las vistas de la aplicación.

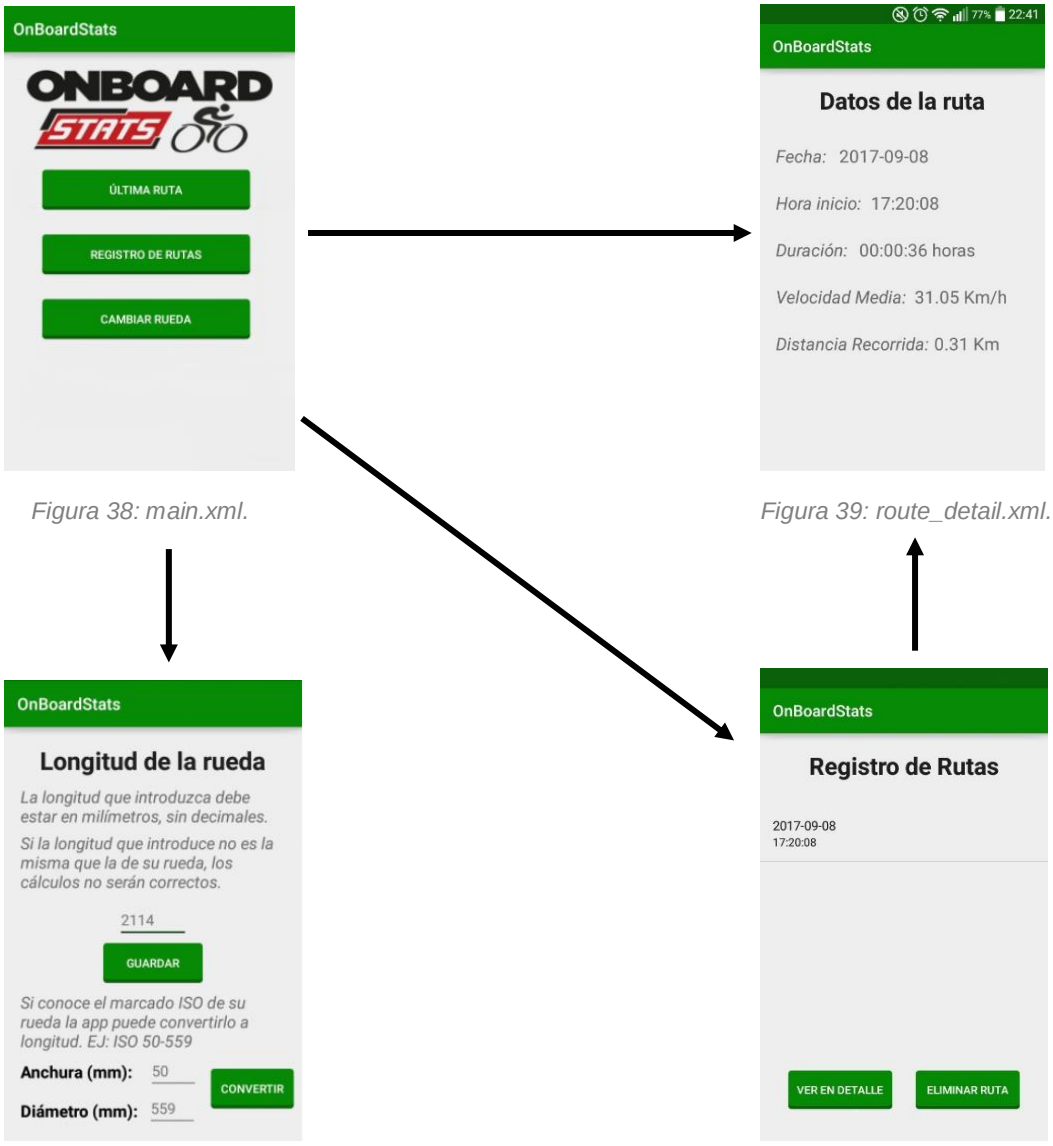


Figura 38: main.xml.

Figura 39: route_detail.xml.

Figura 41: edit_wheel.xml.

Figura 40: routes.xml.

4.2.2 | Capa de negocio

Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los datos, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

En Android, los programas que controlan las vistas son clases Java denominadas **Actividades**. En este caso se ha optado por hacer que todas estas clases hereden de *AppCompatActivity*. Dicho esto, las actividades implementadas para el correcto manejo de las vistas de la app son las que se describen:

- **MainActivity.java:** controla la vista **main.xml**. Según el botón que se pulse en la vista, se navegará hacia una vista u otra y se creará un intent para poder comenzar la nueva actividad. Además, al crearse y al volver a esta actividad, si no se ha almacenado aún la longitud de la rueda aparecerá un diálogo de alerta (Figura 42) pidiéndole al usuario que ingrese esta medida. Si este acepta, la actividad **EditWheel** comenzará.

Figura 42: Diálogo de alerta inicial.

- **RouteDetail.java:** controla la vista **route_detail.xml**. Se encarga de obtener los datos de la ruta mediante una llamada a la base de datos. Dependiendo de qué actividad llame a RouteDetail, esta conseguirá la información del último trayecto realizado o del escogido por el usuario. Si no hay ninguna ruta almacenada, se avisa de esto mediante una notificación.

- **Routes.java:** controla la vista **routes.xml**. Al igual que hace la actividad **RouteDetail**, si el registro de rutas está vacío, se indica esto mediante una notificación y aquí además se deshabilitan los botones. Si no lo está, obtiene este listado y se lo pasa a la vista para la presentación de esta información.

  Cuando el usuario selecciona una ruta de la lista y pulsa el botón de 'Ver Detalles', se navega hasta **RouteDetail**, en donde se visualizará toda la información del trayecto.

  Por el contrario, si se pulsa el botón de 'Eliminar Ruta', se genera un diálogo de alerta (Figura 43) para confirmar la operación por parte del usuario.

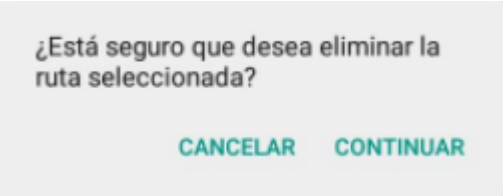


Figura 43: Diálogo de confirmación.

- **EditWheel.java:** controla la vista **edit_wheel.xml**. Esta actividad sirve para guardar la longitud de la rueda. El usuario deberá escribir (en milímetros) esta medida, pero si el formato dado es incorrecto no se guardará en la base de datos y se le notificará al usuario. Si el formato es correcto sí que se almacenará la longitud y también se avisará de la compleción de la operación.

  También existe la opción de convertir la medida del marcado internacional (más información sobre el marcado en el manual) a longitud del neumático. Si el formato de la medida dada por el usuario es correcto, se harán los cálculos para obtener dicha longitud, y este valor se colocará en el cuadro de texto de esta medida. De este modo, el usuario solo tiene que pulsar el botón 'Guardar' para almacenar la longitud de la rueda de su bicicleta.

4.2.3 | Capa de datos

Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por el gestor de bases de datos SQLite que realiza todo el tratamiento de los mismos. Recibe solicitudes de almacenamiento o recuperación de la información desde la capa de negocio.

La base de datos que se ha desarrollado es bastante simple, ya que sólo cuenta con dos tablas que, además, no se relacionan entre ellas. A continuación, se explican brevemente cada una:

- **Rutas:** tabla principal de la base de datos. Aquí se almacenarán los datos de cada una de los trayectos guardados por el usuario. Campos:
    - **Id:** Identificador numérico de la ruta que además es la clave primaria de la tabla.
    - **Fecha:** fecha de la ruta en formato AAAA-MM-DD.
    - **Hora:** hora de comienzo en formato HH:MM:SS.
    - **Duración:** tiempo empleado en la ruta en segundos.
    - **Vmedia:** velocidad media durante el trayecto en kilómetros por hora.
    - **Distancia:** distancia recorrida en kilómetros.

- **Rueda:** tabla para almacenar la longitud de la rueda. Tendrá una entrada o ninguna, ya que las posteriores modificaciones de la longitud se harán sobre la única fila de la tabla. Campos:
    - **Id:** clave primaria de la tabla.
    - **Longitud:** longitud de la rueda de la bici.

Para poder gestionar la interacción con la base de datos se ha implementado la clase **DBAdapter.java**. Las actividades de la capa de negocio que necesitan comunicarse con la base datos, crean una instancia de esta clase. Los métodos que se han definido permiten las siguientes funcionalidades:

- Abrir la base de datos (en modo lectura o modo escritura, según las operaciones que se vayan a realizar) y cerrarla.

- Insertar una ruta, eliminarla, obtener el listado de estas o solo una de ellas.
- Insertar la longitud de la rueda de la bicicleta, modificarla y recuperar este dato.

Internamente también se define la clase **DataBaseHelper** que a su vez extiende de *SQLiteOpenHelper* que tiene métodos para crear y actualizar la base de datos.

CAPÍTULO 5 | COMUNICACIÓN ENTRE DISPOSITIVO Y APLICACIÓN

Este quinto capítulo se centra en explicar de qué manera se ha implantado la conexión entre las dos partes del sistema, el dispositivo físico y la aplicación Android. Cómo se comentó en el Capítulo 2, se ha optado por utilizar Bluetooth como medio de comunicación, ya que es una tecnología que permite conectar dispositivos que se encuentran a poca distancia el uno del otro y su implementación no es complicada.

Para poder llevar a cabo dicha implementación se tuvieron que hacer unos retoques tanto en el subsistema físico como en la app con respecto a lo explicado en los Capítulos 3 y 4. Estos cambios se detallan en los siguientes apartados.

5.1 | Cambios en el dispositivo

Se incorpora un módulo Bluetooth y se realizan unos pequeños cambios en la máquina de estados del subsistema hardware. En las siguientes líneas se explica el funcionamiento de este módulo y se muestran los cambios en los estados del subsistema.

5.1.1 | Módulo Bluetooth

Lo primero que se hizo para adaptar el dispositivo es añadirle un módulo Bluetooth. De este modo el subsistema hardware ya puede comunicarse con el móvil en el que la aplicación esté instalada, y enviarle a este los datos medidos durante la ruta.

El módulo Bluetooth elegido ha sido el HC-06 (Figura 44). Su hermano, el modelo HC-05 cuenta con un número mayor de órdenes de configuración, y puede actuar

tanto de maestro como de esclavo, mientras que el HC-06 solo puede ejercer de esclavo y dispone de un juego reducido de instrucciones. A pesar de ello, para nuestro sistema el HC-06 es el ideal, ya que el *smartphone* será el que haga las veces de maestro, ese limitado número de instrucciones nos sirve para configurar correctamente el módulo, y, evidentemente, es más económico que su hermano.

Figura 44: Módulo Bluetooth HC-06.

Conexiones

El módulo cuenta con 4 pines (Figura 45), de los cuales el RX se conecta al TX (pin 1) del Arduino, y el TX al RX (pin 0) de la placa, usando así el puerto serie de esta para realizar las comunicaciones. Se cruzan los pines RX y TX porque cuando el módulo transmite, la placa recibe, y cuando Arduino transmite, el módulo recibe datos. Los otros pines del módulo, +5V y GND, se conectan a la alimentación y a la toma de tierra del Arduino, respectivamente. En la Tabla 7 se resumen estas conexiones entre el módulo Bluetooth y la placa Arduino.

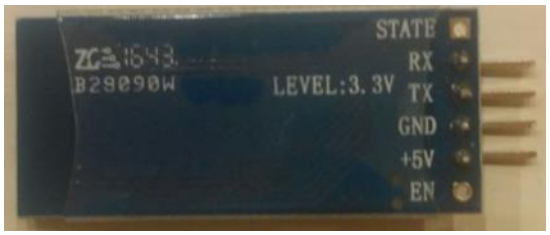


Figura 45: Conexiones del módulo Bluetooth

Módulo	Arduino
RX	TX (Pin 1)
TX	RX (Pin 0)
GND	GND
+5V	5V

Tabla 7: Resumen de conexiones módulo-Arduino.

Configuración del módulo

Antes de incorporar este nuevo componente al dispositivo, hay que configurarlo. Para ello, se cambian algunas características del mismo, como el nombre del módulo, su código de emparejamiento o su velocidad de comunicación, mediante comandos AT [27]:

- **Nombre del módulo:** por defecto se llama HC-06 pero se puede cambiar con el comando *AT+NAME<Nombre>*. El nombre puede ser de hasta 20 caracteres y el elegido fue **OnBoardStats**.
- **Código de emparejamiento:** por defecto viene con el código 1234, para cambiarlo hay que enviar el comando *AT+PIN<código>*. Ha de ser de 4 dígitos y el pin fijado es **9600**.
- **Velocidad de comunicación:** la velocidad por defecto es de 9600 baudios y se puede modificar con el comando *AT+BAUD<número>*, donde *número* equivale a una velocidad en baudios. Estos valores pueden ser:
    - **1 –** 1200 baudios
    - **2 –** 2400
    - **3 –** 4800
    - **4** – 9600
    - **5** – 19200
    - **6** – 38400
    - **7** – 57600
    - **8** – 115200

    Este valor no se modifica, por lo que la velocidad de comunicación se queda con su valor por defecto (9600 baudios).

Una vez que hemos configurado el módulo ya podemos incorporarlo al subsistema hardware. En la Figura 46 se aprecia cómo han quedado las conexiones del dispositivo, donde todos los pines digitales de la placa han sido utilizados.

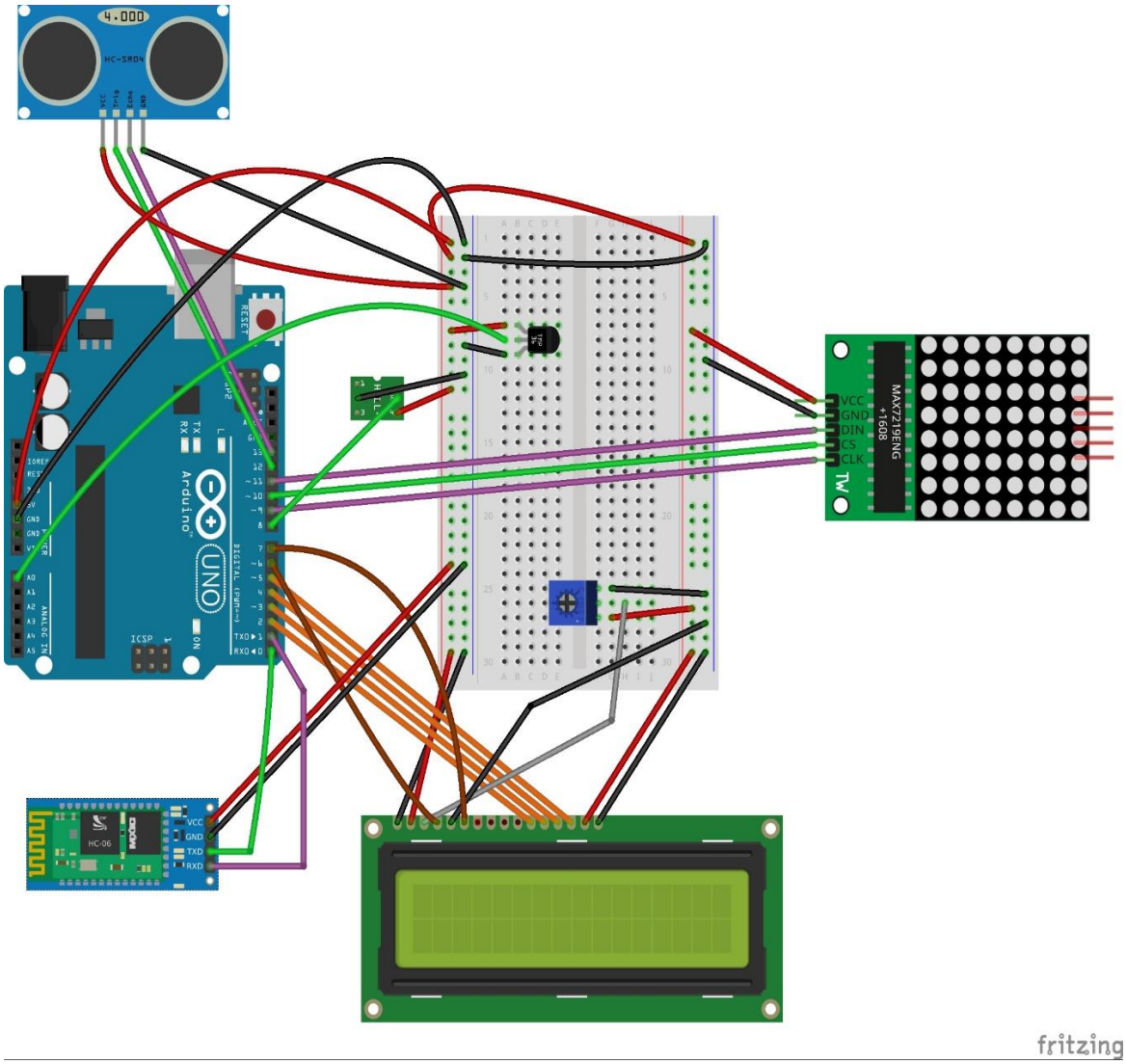


Figura 46: Esquema visual de las conexiones (dispositivo completo + módulo Bluetooth).

Coste del dispositivo

Como ya se explicó en el Capítulo 3, el coste del dispositivo era de unos 50 euros aproximadamente, ya que habría que descontar unos 5 euros de cables e imanes sobrantes. Si aquí también descontamos esa cantidad, el coste del dispositivo con módulo Bluetooth rondaría los 56 euros (el módulo costó 5,87 euros). En la Tabla 8 se desglosa el precio de cada uno de los componentes.

Placa Arduino Uno	19,00 €
Pantalla LCD 16x2	9,90 €
Sensor de temperatura	1,77 €
Sensor de efecto Hall	1,95 €
Sensor de ultrasonidos HC-SR04	1,95 €
Matriz LED MAX7219	3,91 €
Imanes de neodimio	3,91 €
Cables	5,86 €
Potenciómetro 10k Ω	0,48 €
Batería Externa 2600 mAh	6,53 €
Módulo Bluetooth HC-06	5,87 €
TOTAL	**61,13 €**

Tabla 8: Desglose del coste del dispositivo con el módulo Bluetooth.

5.1.2 | Nuevos casos de uso

Al incluir el módulo Bluetooth, ya se pueden enviar los datos calculados durante el trayecto y por lo tanto el usuario tiene disponible una nueva acción, enviar los datos de ruta al móvil. En la Figura 47 se pueden ver los casos de uso, estando los cambios coloreados en rojo.

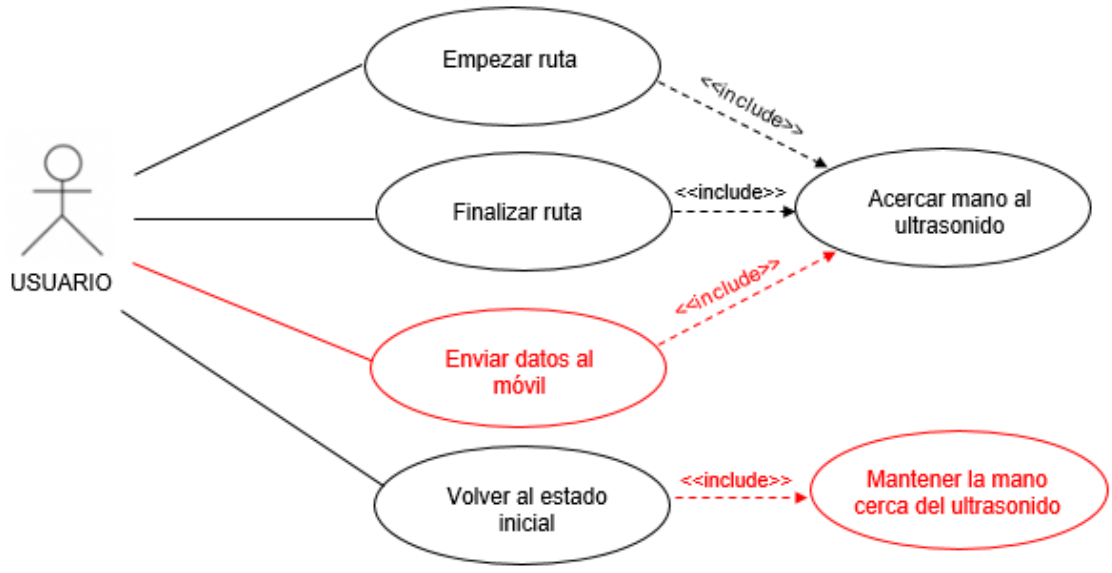


Figura 47: Nuevos casos de uso del dispositivo.

5.1.3 | *Cambios en la máquina de estados*

Para adaptar el dispositivo a la comunicación Bluetooth, los estados del subsistema hardware sufren unas pequeñas modificaciones.

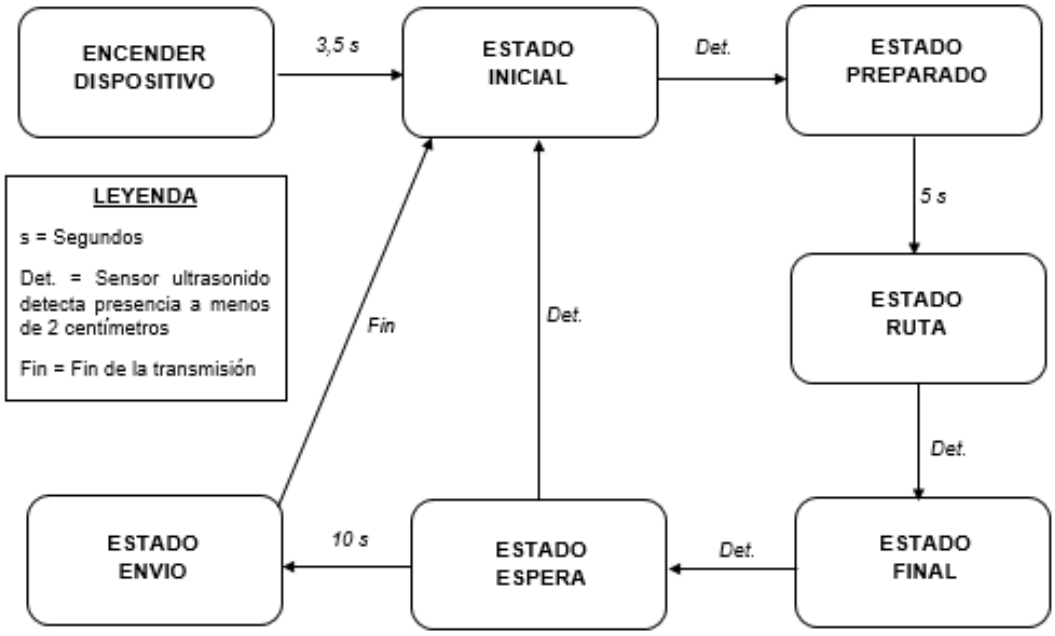


Figura 48: Diagrama de estados del dispositivo.

Seguidamente se explicarán los cambios en los estados y la función de los estados nuevos. Tanto el *Estado Preparado* como el *Estado Ruta* no han sufrido ninguna modificación.

- **Estado Inicial:** si en el puerto serie hay datos por leer, significa que el móvil está enviando la longitud de la rueda, y de este modo se actualiza esta variable en el programa cargado en la placa. Por lo demás, no cambia nada más con respecto al *Estado Inicial* del tercer capítulo.

- **Estado Final:** ahora en lugar de transitar al *Estado Inicial* cuando se detecta presencia a menos de 2 centímetros, lo hace al *Estado Espera*.

- **Estado Espera:** es el único estado de la máquina que puede transitar a dos estados diferentes. El usuario tiene 10 segundos para volver a acercar su mano al sensor de ultrasonido para pasar al *Estado Inicial* y así evitar el envío de los datos de la ruta al móvil. Si por el contrario deja pasar esos 10 segundos, se transita hasta el *Estado Envío*. Al igual que en el *Estado*

Final, en el LCD se verán de forma alterna los datos finales del trayecto, mientras que la matriz LED mostrará una señal de parar, de modo que el usuario no nota el paso del *Estado Final* al *Estado Espera*.

- **Estado Envío:** aquí se produce la transmisión de los datos de la ruta al móvil. En la pantalla LCD se notifica de esta operación y en la matriz LED aparecerá una señal de envío. Una vez se termina la comunicación, se pasa de nuevo al *Estado Inicial*, para el comienzo de un nuevo trayecto.

Al igual que se explicaba en el Capítulo 3, si se quiere apagar el dispositivo, hay que desconectar la fuente de alimentación de la placa.

5.2 | Cambios en la aplicación

Del mismo modo que el subsistema hardware sufre algunas modificaciones, ocurre igual con la aplicación Android. En este caso, se amplían los casos de uso y la arquitectura se mantiene, aunque hay algunos cambios internos en cada una de las tres capas de la misma. En los próximos apartados se explican las adaptaciones llevadas a cabo.

5.2.1 | Nuevos casos de uso

Como ya se comentó en el Capítulo 4, la funcionalidad está descrita por los diferentes casos de uso que el usuario pueda darle al sistema. Estos eran la visualización de los datos de cualquiera de las rutas realizadas, el acceso al listado rutas, donde el usuario puede ver y eliminar cualquiera de estas, y gestionar la longitud de la rueda.

Con las nuevas modificaciones, se añaden a los casos ya descritos tres nuevos que son el envío de la longitud de la rueda al dispositivo, la recepción de los datos de la ruta, y la visualización de la ayuda al usuario.

- **Envío de la longitud:** el usuario selecciona el ordenador de a bordo de entre los dispositivos que estén emparejados con su *smartphone*, se

conecta con el ordenador y le envía la longitud de la rueda almacenada en el base de datos.

- **Recepción de los datos de la ruta:** el *smartphone* espera el envío de la información del trayecto y cuando la recibe, la guarda en la base de datos, en la tabla 'Rutas'.

- **Visualización de ayuda al usuario:** en la propia aplicación existe una pequeña ayuda al usuario sobre estos dos nuevos casos de uso que se acaban de explicar.

En la Figura 49 se puede observar un esquema con los casos de uso de la aplicación, estando en negro los que ya estaban y en rojo los nuevos.

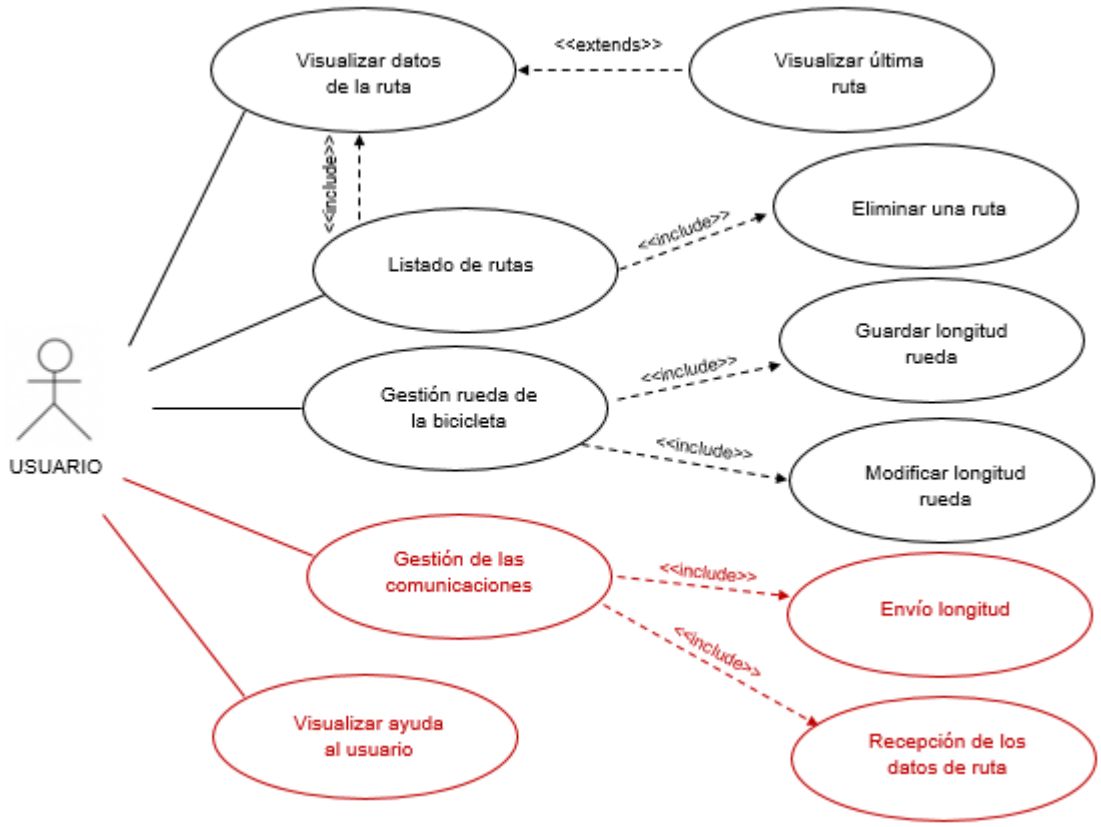


Figura 49: Nuevos casos de uso de la aplicación.

5.2.2 | Cambios en la arquitectura

La arquitectura en tres capas se mantiene, pero se han realizado modificaciones en cada una de estas capas. A continuación, se comentan estos cambios llevados a cabo.

- **Capa de presentación:** las vistas **route_detail.xml (Figura 39)**, **routes.xml (Figura 40)** y **edit_wheel.xml (Figura 41)** no sufren ninguna actualización, mientras que a la vista **main.xml (Figura 38)** sí que se la han realizado pequeñas modificaciones. Además, se crean dos vistas nuevas.

    - **main.xml:** se le añaden dos nuevos botones que permiten comenzar el trayecto y visualizar la ayuda al usuario. Por otro lado, la apariencia de esta vista cambia dependiendo de si se está en ruta o no. Si no lo está, se puede navegar a todas las otras vistas. En cambio, si está en ruta, el texto del botón 'Comenzar' es ahora 'Finalizar' y los otros botones están deshabilitados. Estas diferencias se aprecian mejor en la Figura 50.

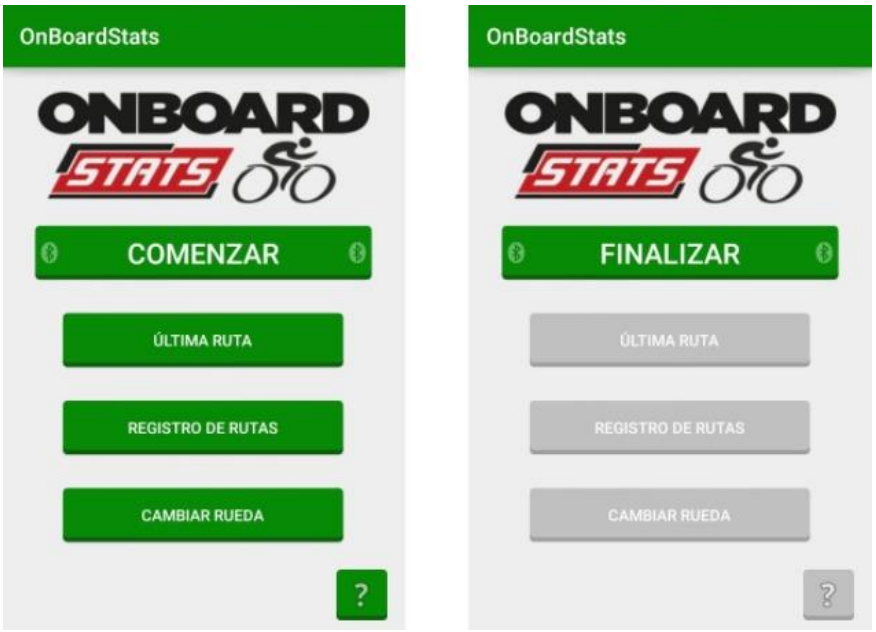


Figura 50: Diferencias en la vista main.xml.

- **bluetooth.xml (Figura 51):** es una de las nuevas vistas. Muestra el listado de dispositivos con los que nuestro *smartphone* está emparejado. De cada uno de ellos se visualiza su nombre y su dirección, con la cual se establece la conexión. Para que el ordenador de a bordo aparezca en el listado, previamente se ha de emparejar con el móvil que se está usando.

- **help.xml (Figura 52):** en esta vista se muestra el funcionamiento de parte de la aplicación. Concretamente se intenta explicar con imágenes cómo funciona la comunicación entre el móvil y el dispositivo. Dado que en esta pantalla se incluyen varias capturas de las otras vistas, esta se define como *ScrollView* [28], lo cual permite deslizar verticalmente la pantalla para ver todo el contenido incluido en la vista.

En la siguiente página se muestra el diagrama de interacción de las vistas de la aplicación, en el que se incluyen las dos nuevas creadas para la comunicación Bluetooth y la ayuda al usuario.

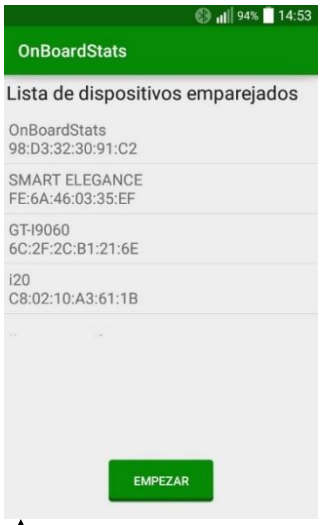


Figura 51: bluetooth.xml.

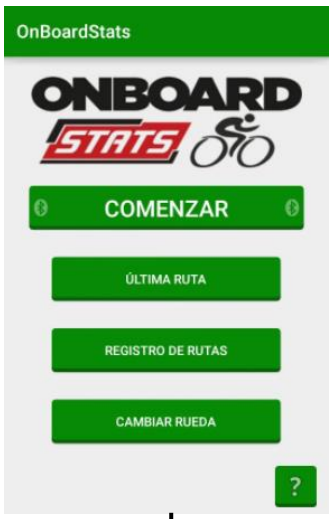


Figura 52: help.xml.

- **Capa de negocio:** al igual que con ocurría con las otras vistas, para las nuevas es necesario crear una actividad que las controle. Además, **MainActivity.java** se modifica para adaptarse a la nueva funcionalidad.
  La descripción de estas actividades, todas ellas herederas de *AppCompatActivity*, se expone a continuación:

  - **BluetoothAdmin.java:** controla la vista **bluetooth.xml**. En primer lugar, comprueba si el *smartphone* tiene Bluetooth, y si lo tiene se asegura de que esté activado. Si no lo está, pide permiso al usuario para activarlo (Figura 53). Una vez está activo, recupera la lista de dispositivos a los que el teléfono móvil está emparejado. Si el usuario selecciona uno de estos dispositivos listados, la dirección de este se guarda en una variable. Si el botón 'Empezar' es pulsado, se guardan en base de datos la fecha y hora de comienzo y esta dirección Bluetooth.

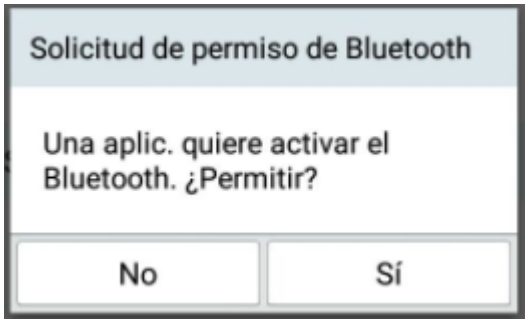


Figura 53: Permiso de activación del Bluetooth.

  - **Help.java:** simplemente permite la visualización de la vista **help.xml**.

  - **MainActivity.java:** sigue controlando la vista **main.xml**, y mantiene toda la funcionalidad anterior. Aquí además comprueba si en la base de datos están almacenados la dirección y la fecha y hora de comienzo, y si lo están, intenta establecer conexión con el ordenador de a bordo para enviarle la longitud de la rueda almacenada. Si consigue conectar con el dispositivo, se cambia el

texto del botón ‘Comenzar’ por ‘Finalizar y se deshabilita el resto de botones.

Para cambiar esta apariencia y que la vista **main.xml** vuelva a su estado original existen dos opciones: esperar a que se termine la ruta y el ordenador de a bordo envíe los datos del trayecto, o bien que el usuario pulse el botón ‘Finalizar’, descartando así el almacenamiento en base de datos de la información recibida. Sea de la manera que sea, los datos auxiliares (fecha y hora de inicio y dirección Bluetooth del dispositivo) se eliminan.

- **Capa de datos:** aquí los cambios han sido mínimos. Se mantienen las dos tablas ya creadas y se añade una nueva, llamada **Auxiliar**. Los campos de esta tabla son los que siguen:
  - **Id:** clave primaria de la tabla.
  - **Aux1:** fecha de comienzo de la ruta.
  - **Aux2:** hora de comienzo.
  - **Dirección:** dirección Bluetooth del dispositivo seleccionado en la vista **bluetooth.xml**.

Además, la clase **DBAdapter.java**, incluye nuevos métodos para insertar, recuperar y eliminar la fila de la tabla **Auxiliar**, además de las otras funciones ya explicadas en el cuarto capítulo.

CAPÍTULO 6 | CONCLUSIONES Y LÍNEAS FUTURAS

En este último capítulo se explica cómo surge la idea para este proyecto, lo que ha supuesto su realización y su posible comercialización. También se comentan posibles mejoras futuras del sistema, principalmente en el sistema hardware.

6.1 | Conclusiones

La idea de crear un ordenador de a bordo para bicicletas surge de mi deseo de poner en prácticas los conocimientos que adquirí el verano pasado en desarrollo de aplicaciones Android y desarrollo con Arduino, junto con todo lo que he aprendido a lo largo de mis años de estancia en la Universidad de Málaga.

Después de darle muchas vueltas y descartar otros posibles proyectos, me decanté por este porque el deporte es una de mis grandes aficiones y porque pienso que este sistema – ordenador de a bordo más la aplicación móvil –puede ser realmente útil para todo tipo de ciclistas, sobre todo para aquellos que la bici es una manera de hacer ejercicio o simplemente un pasatiempo. Como se explicó en el Capítulo 2, para ciclistas profesionales existen otros dispositivos más completos (a la par que caros) que les pueden ayudar más en su trabajo que el que aquí se ha documentado.

Además, gracias al desarrollo de este proyecto he podido adquirir nuevos conocimientos. He aprendido algunos conceptos de electrónica y he comenzado a consultar los datasheets – documento en el que vienen las características del componente en cuestión – para saber que uso tiene cada pin en los sensores, en la matriz, en la pantalla y en el módulo Bluetooth, para informarme sobre el consumo de corriente de estos elementos o bien para conocer otros detalles de los componentes utilizados para formar el sistema hardware.

Asimismo, he ampliado mis conocimientos en Arduino – uso de nuevas librerías para mí – y en Android – manejo de bases de datos y comunicación mediante Bluetooth.

También he mejorado en otros aspectos que no están directamente ligados con la informática, como la manera de organizarme y planificar un proyecto. El tener que seguir una planificación, marcándome fechas de entrega y metas, me ha llevado a adquirir unos buenos hábitos de trabajo.

En cuanto al tema de una posible comercialización del dispositivo, creo que con unos pocos retoques y reduciendo costes comprando al por mayor los componentes, no sería una utopía lograrlo. Una buena formar de financiar esta idea es mediante *crowdfounding*, de modo que según la cantidad de dinero que el inversor aporte, recibirá un ordenador de a bordo con más o menos prestaciones y una aplicación con más o menos características.

6.2 | Líneas futuras

Estos pequeños retoques en el dispositivo que se acaban de mencionar serían principalmente usar en lugar de Arduino Uno, Arduino Mini o Nano (según las necesidades), de menor tamaño y de coste inferior (Nano de coste similar), y usar un circuito impreso en lugar de una protoboard. Estos cambios reducirían considerablemente el precio y el tamaño del dispositivo. También se podría utilizar otra fuente de alimentación, aunque considero interesante que esta batería recargable le sirva al usuario para otras funciones, como cargar su teléfono móvil.

Otra modificación interesante es utilizar un material diferente a la madera para la carcasa e idear un método de montaje y configuración en serie para suministrar grandes pedidos si finalmente se optara por comercializar el sistema.

En cuanto a la funcionalidad del sistema hardware, existen unas cuantas ampliaciones bastante interesantes:

- **Medición de la humedad:** Este cambio sería muy fácil de llevar a cabo, ya que únicamente habría que sustituir el sensor de temperatura, por un DHT11 o DHT22. Son dos modelos de una misma familia de sensores que permiten realizar la medición simultánea de temperatura y humedad. DHT11 es el hermano pequeño de la familia, y cuenta con peores características técnicas, mientras que el DHT22 es el modelo superior, pero, por contra, tiene un precio superior [29]. En la tabla 9 se pueden observar las diferencias entre los dos sensores de humedad y temperatura.

	DHT11	**DHT22**
Medición de temperatura	Entre 0 a 50 (precisión 2 ºC)	Entre -40 a 125 (precisión 0.5 ºC)
Medición de humedad	Entre 20 a 80 % (precisión 5%)	Entre 0 a 100 % (precisión 2.5%)
Frecuencia de muestreo	1 muestra por segundo (1 Hz)	2 muestras por segundo (2 Hz)

Tabla 9: Diferencias entre el sensor DHT11 y el DHT22

- **Medición de la altitud:** Otra modificación sencilla de implementar es añadir al subsistema hardware un barómetro digital, que mide la presión del aire y puede usarse como altímetro, estimando la altitud del sensor respecto al nivel del mar.

  El sensor barométrico a usar sería el BMP180 [30], ampliamente empleado en aplicaciones meteorológicas, como estaciones registradoras o relojes que muestran el clima, entre otras. También puede ser empleado en aplicaciones de climatización o de control de ventilación.

  Su rango va desde 300hPa a los 1110hPa, equivalente a una altitud de -500 a 9000m sobre el nivel del mar. La precisión también es configurable, desde 0.06 hPa (0.5 metros) en el modo de bajo consumo, a 0.02 hPa (0.17 metros) en el modo de alta precisión.

- **Localización (GPS):** En este caso la mejora es bastante significativa, ya que se podría registrar el camino que el usuario ha tomado durante su ruta en bici.

Para ello se podría utilizar un GPS de la familia NEO-6 [31], que son una familia de receptores fáciles de conectar a un autómata o a un procesador como Arduino.

Los datos del NEO-6 están en formato $GPRMC, una de las secuencias disponibles en el protocolo NMEA 0183 [32] (National Marine Electronics Asociation). Para interpretar la secuencia $GPRMC de forma sencilla disponemos de la librería *TinyGPS*, que nos permite transformar los datos obtenidos con el módulo GPS a las clásicas coordenadas (latitud y longitud)

- **Uso de una pantalla más grande:** Con todos estos nuevos datos que se pueden obtener, se necesitará una pantalla de mayor capacidad de caracteres para mostrarlos. Existen pantallas LCD de 20x4 (muy similar a la usada en el dispositivo, sólo que un poco más grande) o de 128x64. Esta última además sólo requiere de 3 pines (de ellos uno para la alimentación y otro para la toma de tierra), lo cual libera considerablemente los pines digitales de la placa, y permite hacer figuras en ella.

- **Uso de un teclado:** Consistiría en incorporar al dispositivo un teclado como el que aparece en la Figura 52. Gracias a este nuevo componente, el usuario podría comunicarse con la placa Arduino sin la obligatoriedad de usar la aplicación diseñada.

Figura 52: Teclado para la placa Arduino

En cuanto a la aplicación móvil, lo más inmediato sería adaptarla a otros sistemas operativos, principalmente a iOS. Aunque en el segundo capítulo se dijo que Android es, con diferencia, el sistema operativo más usado en *smartphones,* también es cierto que la cuota de mercado de iOS ha crecido considerablemente en el último año tanto en España como en el resto del mundo [15]. Por ello, tener la aplicación disponible en los dos sistemas operativos líderes sería una gran mejora. Si en los próximos años se alza algún otro S.O., se estudiaría la adaptación de la app a este nuevo contendiente en el mercado.

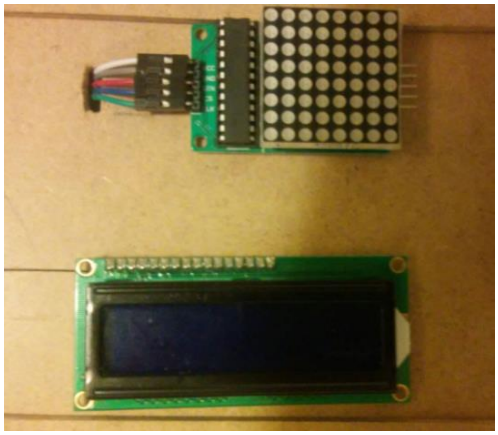
ANEXO A | MANUAL DE INSTALACIÓN Y USO

En las próximas líneas se explicará cómo preparar el ordenador de a bordo y la aplicación para su uso, así como la correcta utilización de ambos componentes del sistema.

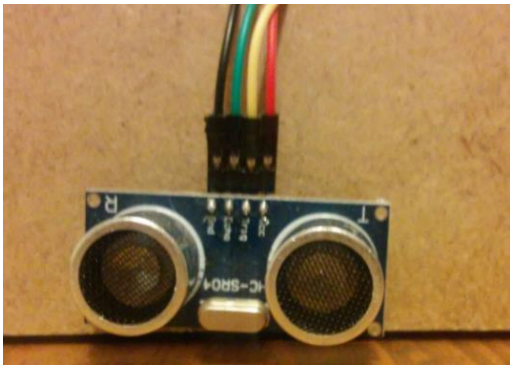
Manual de Instalación

Antes de nada, se deben conocer las partes del ordenador de a bordo, que se muestran con imágenes:

- Parte superior, en la que se encuentran la pantalla LCD y la matriz LED.

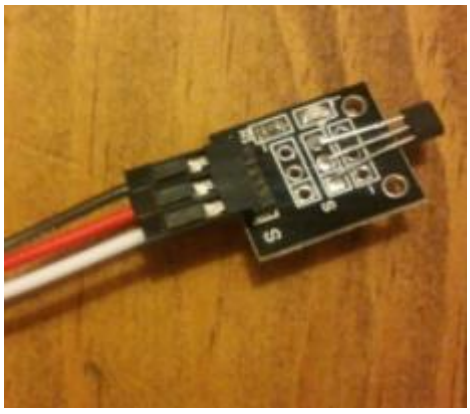


- Sensor de ultrasonidos, en el lateral izquierdo



- A la derecha del usuario están el cable USB, con el que se alimenta el Arduino, y la batería externa.

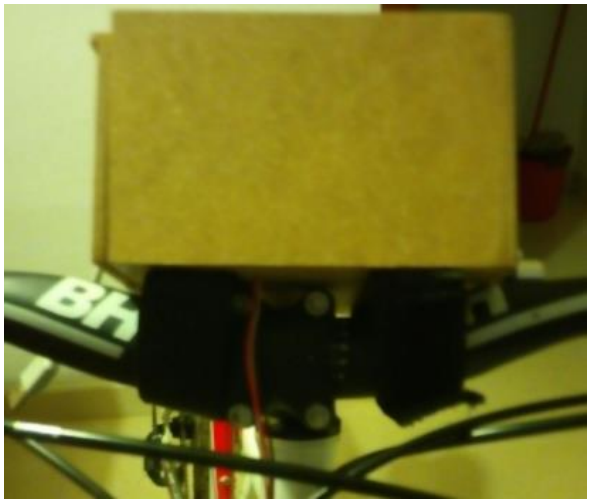
- Sensor de efecto Hall, en la parte inferior de la caja.



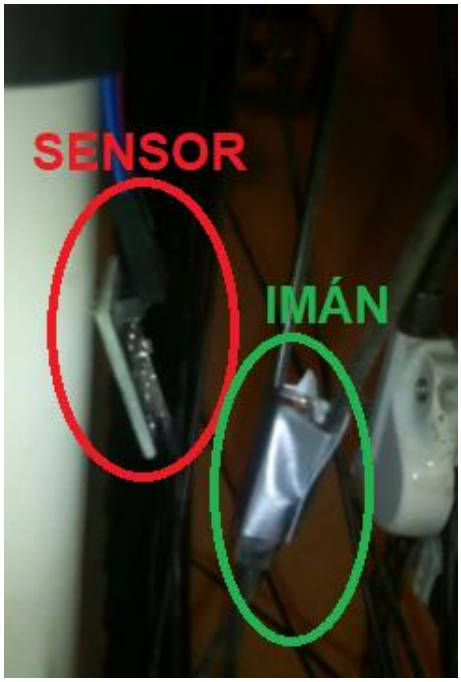
- Por último y también en la parte inferior de la carcasa, se encuentran las tiras de velcro, para sujetar el ordenador de a bordo.

Una vez se sabe dónde está cada elemento del ordenador de a bordo, se deben seguir los siguientes pasos para instalar el sistema:

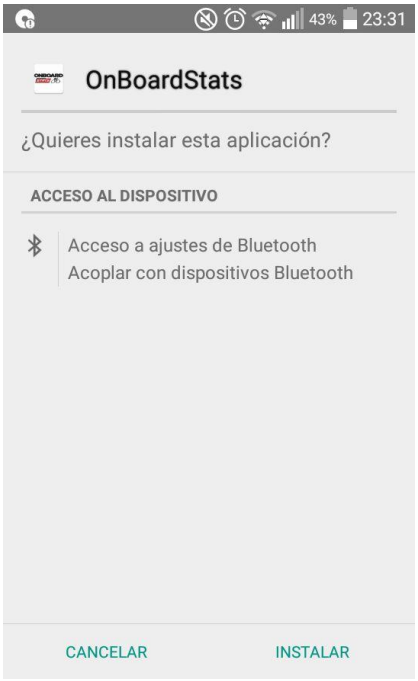
1. Colocar el dispositivo al manillar de la bicicleta y sujetarlo a él con las tiras de velcro.



2. Fijar el sensor de efecto Hall a la horquilla de la rueda delantera y hacer lo propio con el imán de neodimio a un radio. **IMPORTANTE:** tanto el sensor como el radio deben estar a la misma altura para que el primero detecte al segundo.



3. Instalar la aplicación en el móvil. Se piden los permisos que son necesarios para el correcto funcionamiento de la aplicación.



4. Por último, emparejar el *smartphone* con el ordenador de a bordo. La clave de emparejamiento del dispositivo es **9600**.

Estos serían los pasos a seguir para instalar el ordenador de a bordo y la aplicación. Ahora se explicará cómo usar ambos correctamente para tomar los datos de la ruta y almacenarlos en el teléfono móvil.

Manual de Uso

A continuación, se van a detallar los pasos que se deben seguir para realizar un trayecto con el sistema completo.

Lo primero y más importante es guardar la longitud de la rueda de nuestra bicicleta si no lo habíamos hecho antes o si vamos a utilizar una bici distinta que tenga una longitud de rueda diferente a la de la que teníamos guardada. Para ello la forma más fácil de obtener esta medida es a través de la marca de tamaño de la rueda, como aparece en la imagen.

Aquí hay que hacer un breve paréntesis para poner en contexto al lector. Hoy en día, los tamaños de los neumáticos de bicicleta son calificaciones que establece la norma ISO 5775 [33] para marcar los neumáticos y llantas de bicicleta. Sin embargo, los marcados tradicionales inglés y francés continúan hasta ahora en uso, y por ello se muestran las características de cada uno de estos marcados de rueda [34]:

- **Marcado internacional (ISO 5775):** la normativa ISO especifica 2 valores siempre en milímetros, anchura de la goma y diámetro de la llanta. Viéndolo con un ejemplo: ISO 50-559.

- **Marcado inglés:** diámetro exterior (llanta y goma) por el ancho del neumático, ambas medidas en pulgadas. Ejemplo: 26 x 2.00”.
- **Marcado francés:** muy similar al inglés, pero en milímetros. Diámetro exterior por la anchura. Ejemplo: 700 x 35C

Una vez conocemos los posibles marcados de nuestro neumático, lo que hay que hacer encontrar en Internet (existen tablas de equivalencias entre marcados) su equivalencia al marcado internacional si no está en este marcado, e insertar los valores de la anchura de la goma y del diámetro de la llanta en la aplicación. Esta se encarga de realizar la conversión a longitud de la rueda

y coloca el valor calculado en el cuadro de texto para que solo se tenga que pulsar el botón ‘Guardar’.

Si no encontramos la equivalencia del marcado de nuestro neumático al marcado internacional, nuestra rueda no tiene marcado o este ya no se ve por el desgaste, tenemos que medir el diámetro manualmente.

Ahora sólo queda multiplicar este diámetro por el número π para obtener la longitud de la (circunferencia) rueda:

$$Longitud\ de\ la\ rueda = Diámetro\ de\ la\ rueda * 3.1416$$

Ahora ya podemos almacenar la longitud de la rueda en la aplicación y circular con el ordenador de a bordo.

Una vez hemos guardado esta medida, todo está listo para comenzar, por lo que pegamos la batería a la carcasa mediante el velcro, y la conectamos con el cable USB para alimentar la placa Arduino. Nos aparecerá un mensaje de bienvenida en la pantalla LCD.

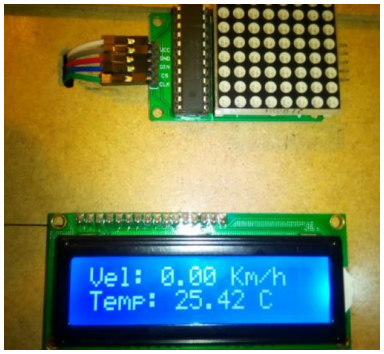
A continuación, debemos coger la aplicación previamente instalada en el móvil y pulsar el botón ‘Comenzar’ en la pantalla inicial. Si el Bluetooth no está activado, la aplicación pedirá permiso para hacerlo. Para continuar hay que tener activado el Bluetooth.

Una vez el Bluetooth está activado hay que seleccionar el dispositivo **OnBoardStats** y enviarle a este la longitud de la rueda que hemos guardado anteriormente.

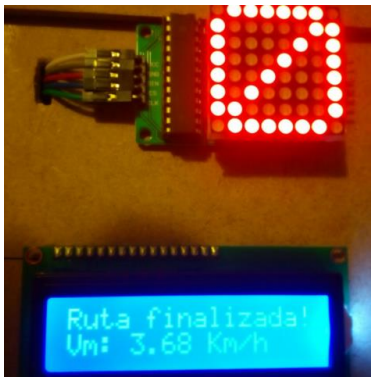
IMPORTANTE: el móvil debe estar cerca del ordenador de a bordo para que se pueda establecer una comunicación entre ambos, tal y como aparece en la siguiente imagen.

Ahora debemos acercar la mano al ultrasonido situado a nuestra izquierda, casi tocándolo, para que detecte presencia. La pantalla nos avisará para que nos preparemos y la matriz realizará una cuenta atrás.

Cuando termina esta cuenta atrás, la matriz hará parpadear una flecha hacia delante los primeros tres segundos y medio, mientras que la pantalla mostrará los datos actualizados del estado de la ruta.



Cuando se termine el trayecto, hay que volver a acercar la mano al ultrasonido para que se pare de contabilizar el tiempo empleado y la distancia recorrida. Al detectar presencia, el ordenador de a bordo mostrará una señal de parada en la matriz y los datos finales de la ruta en la pantalla.



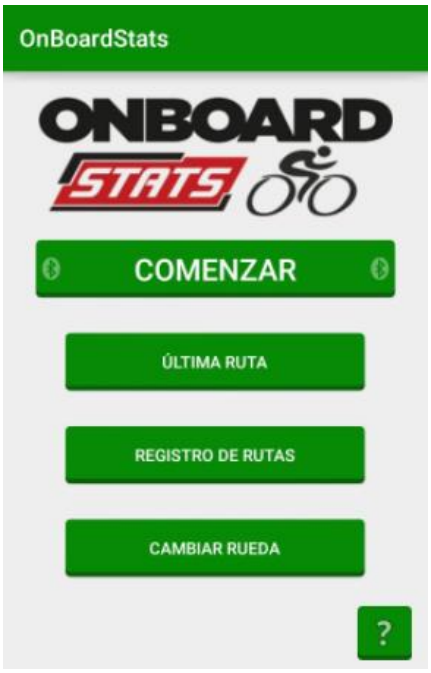
En este momento existen dos opciones, enviarle los datos calculados durante el trayecto o no hacerlo. Para realizar el envío hay que acercar la mano al ultrasonido y esperar unos instantes. El ordenador de a bordo notificará de ello.

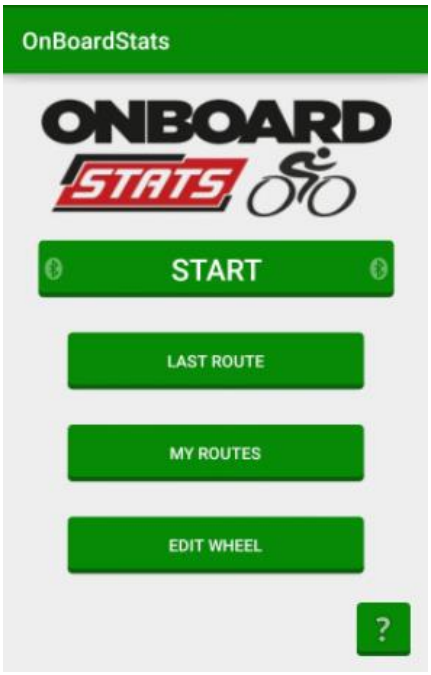
Si no se quiere enviar la información de la ruta, hay que **mantener** la mano cerca del sensor de ultrasonidos hasta que en la pantalla aparezca el mensaje '*Acerque la mano al ultrasonido'* y una señal de OK en la matriz LED.

En cuanto a la aplicación, durante el trayecto solo el botón 'Finalizar' estará habilitado.

Si se reciben los datos de la ruta estos se almacenarán y la aplicación volverá a su estado normal. Para ver esta información recibida bastará con dirigirnos al listado de rutas a través del botón 'Registro de Rutas', o bien clicar el botón 'Última Ruta' para ver los detalles de esta última.

Por último, destacar que los textos de la aplicación están tanto en español como en inglés (Estados Unidos), dependiendo del idioma que tenga configurado el usuario en su móvil. Si el *smartphone* no está configurado en ninguno de estos dos, los textos aparecerán en español.





BIBLIOGRAFÍA

[1] Ciclismo urbano, ventajas para el medio ambiente, de Wikipedia

https://es.wikipedia.org/wiki/Ciclismo_urbano#Ventajas_para_el_medio_ambiente

[2] El uso de la bici en Málaga, una tendencia en auge, del diario local La Opinión de Málaga

http://www.laopiniondemalaga.es/malaga/2017/03/06/bici-malaga-tendencia-auge/914199.html

[3] La bici se acelera en Barcelona, del diario nacional El País

https://elpais.com/ccaa/2017/06/03/catalunya/1496505839_193912.html

[4] Efecto Hall, de Wikipedia

https://es.wikipedia.org/wiki/Efecto_Hall

[5] Build a Readable Bicycle Computer, de David Schneider para la web de IEEE Spectrum

http://spectrum.ieee.org/geek-life/hands-on/build-a-readable-bicycle-computer

[6] Ordenador de a bordo (visualización de velocidad, distancia, temperatura…), de Carlos Cupeiro Durán, Javier López González y Daniel Redondo Arroyo, alumnos de la Universidad Rey Juan Carlos.

https://drive.google.com/file/d/0B7_mgnHy7uaLZEhmU3lmb0VJbFE/view

[7] Cuentakilómetros B'TWIN 100 con cable B'TWIN, de Decathlon

https://www.decathlon.es/cuentakilometros-btwin-100-con-cable-id_8382193.html

[8] Gama TOPLINE 2012 y ciclo computadores, de la marca Sigma

http://www.sigmasport.com/es/produkte/fahrrad-computer

[9] Ciclo-computadores de la firma Garmin

https://buy.garmin.com/es-ES/ES/fitness-y-outdoor/ciclismo/cIntoSports-cCycling-p1.html

[10] Introducción a Arduino, de la web de Arduino

https://www.arduino.cc/en/Guide/Introduction

[11] Arduino, de Wikipedia

https://es.wikipedia.org/wiki/Arduino

[12] Especificaciones de la placa Arduino UNO, de la tienda oficial de Arduino (ver la pestaña *TECH SPECS*)

https://store.arduino.cc/arduino-uno-rev3

[13] Android, de Wikipedia

https://es.wikipedia.org/wiki/Android

[14] Actividades, de la página de desarrolladores de Android

https://developer.android.com/guide/components/activities.html?hl=es-419

[15] Cuota de mercado de smartphones de febrero a abril de 2017, de la consultoría Kantar Worldpanel

http://es.kantar.com/tech/m%C3%B3vil/2017/junio-2017-cuota-de-mercado-de-smartphones-en-espa%C3%B1a-2017/

[16] About SQLite, de la propia web de SQLite

https://www.sqlite.org/about.html

[17] SQLite, de Wikipedia

https://es.wikipedia.org/wiki/SQLite

[18] Tema 2: Técnicas de acceso y control de enlace, apuntes de los profesores de la asignatura Redes y Sistemas Distribuidos.

[19] Bluetooth, de Wikipedia

https://es.wikipedia.org/wiki/Bluetooth

[20] Módulo 2. Tecnologías de comunicaciones para la IoT, apuntes de Mercedes Amor Pinilla para el curso Samsung-UMA de Desarrollo de Aplicaciones para Internet de las Cosas.

[21] Proyecto 11. Bola de cristal, del libro de proyectos de Arduino.

[22] Proyecto 3. Love - O – Meter, del libro de proyectos de Arduino.

[23] Sensor ultrasónico, de Wikipedia

https://es.wikipedia.org/wiki/Sensor_ultras%C3%B3nico

[24] Opciones para alimentar Arduino con baterías, de la web de Luís Llamas.

https://www.luisllamas.es/alimentar-arduino-baterias/

[25] Librería LedControl, de la web de Arduino

http://playground.arduino.cc/Main/LedControl

[26] Extensible Markup Language, de Wikipedia

https://es.wikipedia.org/wiki/Extensible_Markup_Language

[27] Configuración del módulo Bluetooth HC-06 usando comandos AT, de Naylamp Mechatronics

http://www.naylampmechatronics.com/blog/15_Configuraci%C3%B3n--del-m%C3%B3dulo-bluetooth-HC-06-usa.html

[28] ScrollView, de la página de desarrolladores de Android

https://developer.android.com/reference/android/widget/ScrollView.html

[29] ¿Qué es un DHT11 / DHT22?, de Luis Llamas.

https://www.luisllamas.es/arduino-dht11-dht22/

[30] Medir presión del aire y altitud con Arduino y barómetro BMP180, de Luis Llamas.

https://www.luisllamas.es/medir-presion-del-aire-y-altitud-con-arduino-y-barometro-bmp180/

[31] Localización GPS con Arduino y los módulos GPS NEO-6, de Luis Llamas.

https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/

[32] NMEA 0183, de Wikipedia.

https://es.wikipedia.org/wiki/NMEA_0183

[33] Estándar ISO 5775, de Wikipedia

https://es.wikipedia.org/wiki/ISO_5775

[34] Marcas de tamaño de neumáticos de bicicletas, de Wikipedia

https://es.wikipedia.org/wiki/Neum%C3%A1tico_de_bicicleta#Marcas_de_tama.C3.B1os