Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

SENSORIZACIÓN INTELIGENTE Y APP MÓVIL PARA BICICLETAS

Alumno: Fernando Plaza Quevedo

Tutor: Prof. D.ª Elisabet Estévez Estévez
Dpto: Ingeniería Electrónica y Automática

Junio, 2018

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Electrónica y Automática

Doña Elisabet Estévez Estévez , tutora del Proyecto Fin de Carrera titulado: Sensorización inteligente y App móvil para biciletas, que presenta Fernando Plaza Quevedo, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2018

El alumno:

Fernando Plaza Quevedo

La tutora:

Elisabet Estévez Estévez

Agradecimientos

Mi especial agradecimiento a mis padres y hermano por su apoyo incondicional durante esta etapa.

A Elisabet Estévez, tutora de este Trabajo Fin de Grado, por su gran ayuda e implicación, así como a a mi amigo Antonio.

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Electrónica y Automática

MEMORIA

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. MOTIVACIÓN Y OBJETIVOS

Cada día más personas optan por el uso de la bicibleta como medio para realizar sus actividades diarias en lugar del transporte público, lo cual entraña diversos beneficios: no se contribuye al incremento del tráfico, no contamina y supone un hábito saludable para el usuario.

No obstante, conviene recordar que los ciclistas, junto con peatones y motociclistas, siguen constituyendo el colectivo más vulnerable al circular por ciudades y carreteras. A diferencia de otros países europeos, en España, la mayor concentración de accidentes se registran fuera de las ciudades, lo cual se traduce en que nuestro país encabece el ranking europeo de ciclistas fallecidos en carretera, tal y como se recoge en la *Figura 1* (información obtenida de la Dirección General de Tráfico, DGT). Por su parte, En Europa, la mayor parte de los accidentes se producen en zona urbana, resultando de la colisión con un vehículo en el 50% de los casos.

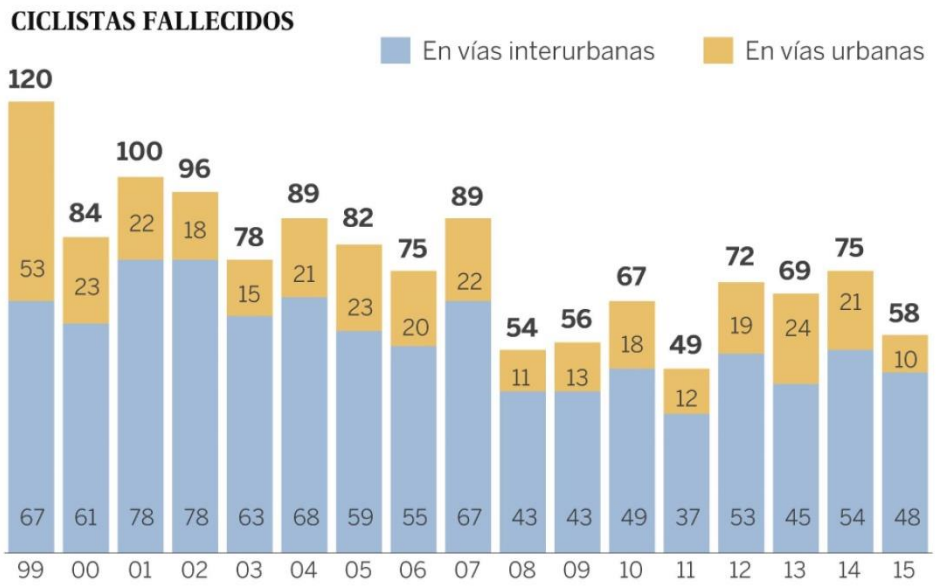


Figura 1. Gráfico de ciclistas fallecidos (1999-2015). [1]

Fenómenos meteorológicos como la lluvia aumentan el riesgo para transitar por las ciudades. Los encharcamientos, por ejemplo, dificultan la percepción de baches, coladeras o cualquier otro tipo de obstáculo. Esto, unido a que las luces de una bicicleta tienen poca notoriedad para los automovilistas, así como una inadecuada

señalización en muchos casos, potencian notablemente el riesgo de accidente para este colectivo vulnerable.

Si bien la concienciación por parte de automovilistas resulta fundamental para reducir el número de accidentes con ciclistas involucrados, el colectivo ciclista también deberá conocer y cumplir la Normativa de Seguridad Vial, destacando el uso del equipamiento obligatorio como casco, luces y chaleco reflectante, entre otros.

A día de hoy, la tecnología ofrece un sinfín de posibilidades para dotar a este colectivo de medios que, por un lado, fomenten el uso de la bicicleta y, por otro, les permita hacerlo de manera más segura. Para satisfacer ambas necesidades, el objetivo del presente Trabajo Fin de Grado radica en la implementación de la sensorización inteligente al campo del transporte en bicicleta.

Para ello se va a diseñar un dispostivo electrónico portable para bicicletas que muestre mediante iluminación tanto las frenadas como el giro realizado por el usuario. Este dispositivo tendrá forma de faro y se iluminará en rojo cuando detecte desaceleraciones y en amarillo cuando se modifique la trayectoria girando a izquierda o a derecha. Además, el faro contiene un GPS que proporcionará los valores de latitud, longitud y altitud, así como un módulo Bluetooth que los enviará a una aplicación móvil, objeto también del presente Trabajo Fin de Grado, para que el usuario disponga de cronología de sus desplazamientos.

2. HARDWARE

La estructura del faro está compuesta por una carcasa, una tapa, un tapón, una tuerca, un tapón hueco, una tuerca hueca y un interruptor, tal y como se puede apreciar en el *Plano Nº2* del documento Planos. El resultado final del faro se muestra en el *Plano Nº1*. Este diseño ha sido realizado con el programa CATIA V5R21.

La tapa presenta un orificio donde se aloja el tapón de M16. Dicho tapón irá roscado a la tuerca, previamente fijada por la parte posterior de la tapa. Esta tuerca va unida a una tuerca hueca y al tapón hueco donde se ubicará el conector USB hembra que permite cargar la batería mediante el cable USB. Por último, en la carcasa se aprecia un orificio de 10 mm de diámetro para colocar el interruptor.

Los elementos internos que constituyen el faro son: una batería de 2,200 mAh de la marca NGS, un HUB *de* cuatro puertos USB 2.0, (ver *Figura 2)*, un cable Micro USB/USB hembra para cargar la batería, dos cables Mini-B/USB y dos Arduino Nano. Para la *iluminación* y *señalización* se utilizará un Arduino con un IMU (Unidad de Medición Inercial) con sus correspondientes leds mientras que para la *App móvil* se empleará un segundo Arduino con un módulo Bluetooth y un GPS.

Figura 2. HUB de 4 puertos USB 2.0. [2]

Dentro de la gran variedad de placas Arduino disponibles en el mercado, en este caso se utilizará el Arduino Nano ya que presenta la misma funcionalidad que el resto, pero con la ventaja de su reducido tamaño, lo cual resulta prioritario para este dispositivo.

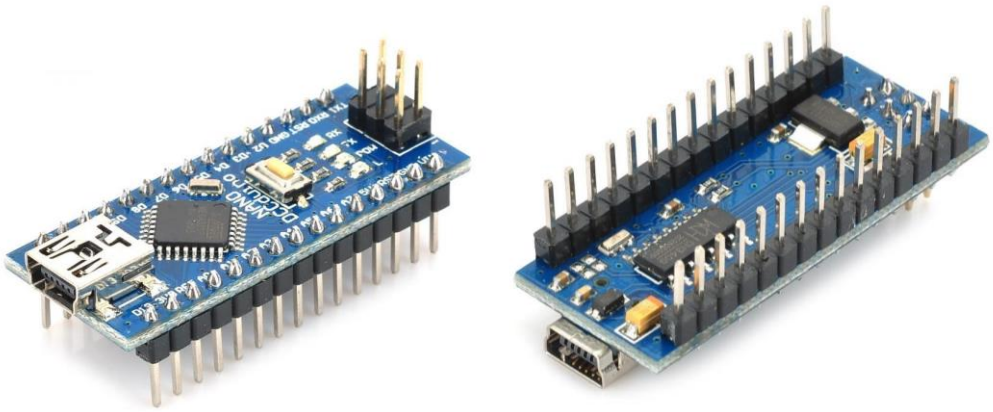


Figura 3. Arduino Nano. [3]

Las características principales del Arduino Nano son las siguientes:

- Microcontrolador: Atmel ATmega328 (ATmega 168 versiones anteriores).
- Tensión de operación (nivel lógico): 5 V.
- Tensión de entrada (recomendado): 7-12 V.
- Tensión de entrada (límites): 6-20 V.
- Pines E/S Digitales: 14 (6 pines – salida PWM).
- Entradas analógicas: 8 – Corriente máxima por cada pin de E/S: 40 mA.
- Memoria Flash: 32 KB (ATmega328) de los cuales 2 KB son usados por el bootloader (16 KB – ATmega168).
- SRAM: 2 KB (ATmega328) / 1 KB (ATmega168).
- EEPROM: 1 KB (ATmega328) / 512 bytes (ATmega168).
- Frecuencia de reloj: 16 MHz.
- Dimensiones: 18.5 mm x 43.2 mm.

El Arduino Nano posee selección automática de la fuente de alimentación y puede ser alimentado a través de:

- Una conexión Mini-B USB.
- Una fuente de alimentación no regulada de 6-20V (pin 30).
- Una fuente de alimentación regulada de 5V (pin 27).

Al alimentar el Arduino a través del Mini USB, el CH340 proporciona una salida de 3.3 V en el pin 16 de la placa. Cuando se conecta a una fuente externa (no USB), los 3.3 V no se encuentran disponibles.

La distribución de los pines del Arduino Nano se muestra en la *Figura 4*.

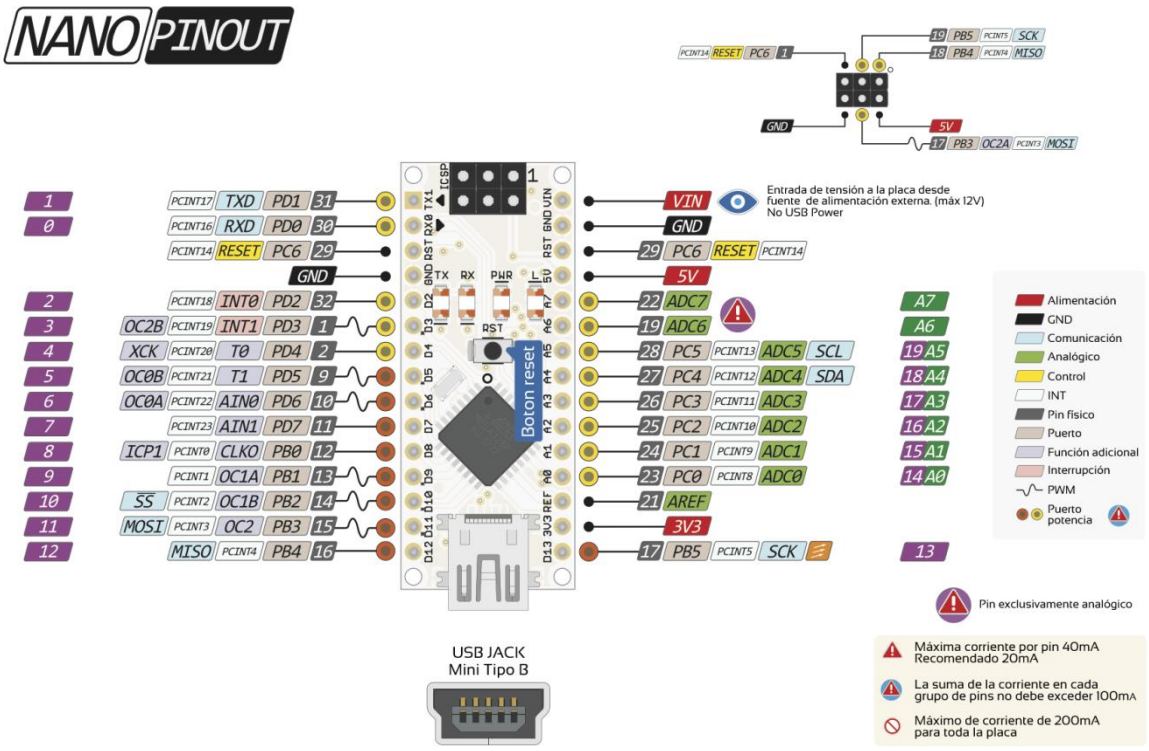


Figura 4. Nano PinOut. [4]

Está basado [5] en el microcontrolador ATmega328P y tiene una entrada Mini-USB tipo B a través de la cual es posible subir el código fuente para la ejecución de los comandos. Consta de 14 puertos digitales de entrada/salida, 8 puertos analógicos, una memoria de 16 KB, 1 KB de SRAM y 512 bytes de EPROM. Su ClockSpeed es 16 MHz. Funciona con un voltaje que puede estar en el rango de 7 a 12 voltios y entrega una corriente de 40 mA.

Para alimentar el resto de elementos se ha optado por utilizar una batería externa USB. Tiene como ventaja que proporciona 5V regulados, por lo que el Arduino se alimenta a través del USB sin preocuparse de la necesidad de regular el voltaje.

Figura 5. NGS Power bank 2,200 mAh. [6]

Las especificaciones principales de la batería son:

- Tecnología de batería: Ión de litio.
- Capacidad de batería: 2,200 mAh.
- Fuente de carga: USB.
- Voltaje de entrada: 5V.
- Voltaje de salida: 5V.

El conexionado de todos los elementos anteriormente mencionados se recoge en la interfaz eléctrica general correspondiente al *Plano UJA-2018-00* del documento Planos, en el apartado de planos eléctricos. La elaboración de los planos eléctricos se ha llevado acabo con AutoCAD Electrical.

En el *Plano UJA-2018-01H01* y el *Plano UJA-2018-02H01* se aprecia el conexionado de los distintos Arduinos con los elementos de iluminación y señalización del faro, así como de la app móvil, lo cual se detallará en los siguientes apartados. Ambos Arduinos se conectarán al HUB USB por medio de un cable Mini-B/USB como puede verse en el *Plano UJA-2018-00H01*. A continuación, el cable de alimentación del HUB USB se conectará al puerto USB de salida de la batería (5V). A este cable se conecta un interruptor para controlar el encendido y apagado del faro. Al puerto USB de entrada de la batería (5V) se conecta el cable Micro USB/ USB hembra, posicionado en el tapón y tuerca huecos mencionados con anterioridad. A través del conector USB hembra se podrá cargar la batería conectándolo al ordenador mediante un cable USB (ver *Plano UJA-2018-00H02)*.

2.1. Iluminación y señalización

El objetivo principal del presente proyecto se basa en el diseño de la iluminación y señalización del giro del faro. Para ello se utiliza un sensor inercial y, en concreto, un IMU (Unidad de Medición Inercial) modelo GY-521 MPU-6050, y los distintos leds para para indicar la señalización. En la *Figura 6* se muestra el esquema completo del montaje, el cual ha sido elaborado con el programa Fritzing.

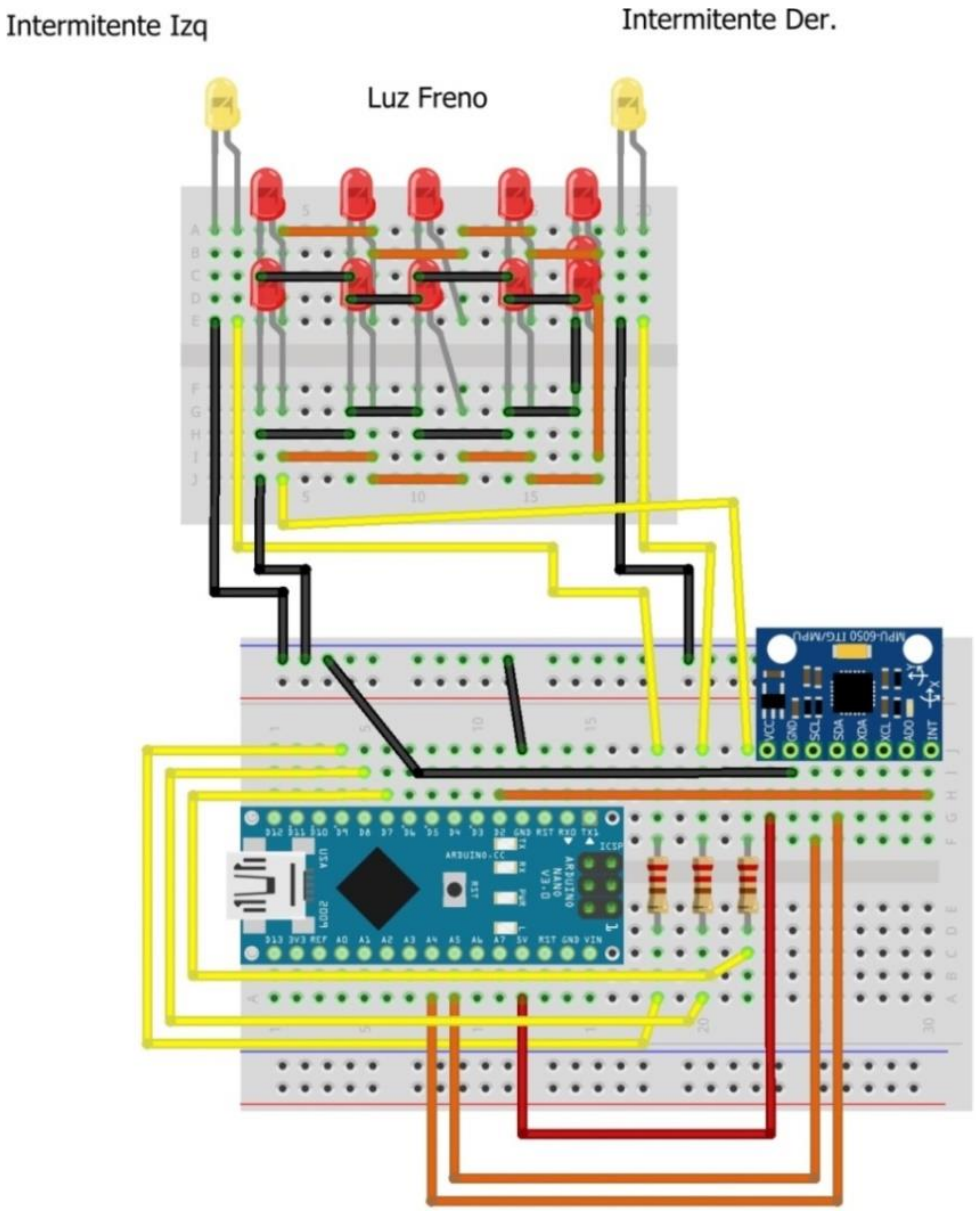


Figura 6. Esquema montaje Arduino para iluminación y señalización del faro.

2.1.1. IMU

Una Unidad de Medición Inercial [7] o Inertial Measurement Unit (IMU) es un dispositivo electrónico capaz de obtener mediciones de velocidad, rotación y fuerzas gravitacionales de un aparato de forma autónoma, mediante una combinación de acelerómetros y giroscopios.

El acelerómetro es un dispositivo que mide la aceleración a la que está sometida el sensor mientras que el giroscopio es un dispositivo totalmente diferencial (no existe una referencia absoluta sino que siempre mide ángulos relativos a una referencia arbitraria) que mide el ángulo de rotación girado por un determinado mecanismo.

Otro aspecto muy importante de los IMUs es la cantidad de grados de libertad (DOF) disponibles, que representan la cantidad de magnitudes independientes que pueden medir. En el presente proyecto se va a utilizar un IMU de 6 DOF, combinando un acelerómetro y un giroscopio, ambos de 3 ejes. El hecho de ser de 3 ejes les permite medir la magnitud y la dirección de la aceleración/rotación que se está ejerciendo en los ejes X, Y, Z del sensor de manera independiente.

El motivo por el que utilizar un IMU es que ambos dispositivos compensan las limitaciones del otro:

- Los acelerómetros no tienen deriva (drift) a medio o largo plazo, ya que realizan la medición absoluta del ángulo que forma el sensor con la dirección vertical, marcada por la gravedad. No obstante, se ven influenciados por los movimientos del sensor y el ruido, por lo que no son fiables a corto plazo. En consecuencia, un acelerómetro no resulta adecuado para determinar la velocidad y la posición.
- Los giroscopios son precisos para movimientos cortos o bruscos. Miden la velocidad angular y obtienen el ángulo por integración respecto al tiempo, acumulando errores y ruido en la medición, por lo que a medio o largo plazo tienen deriva (drift). En conclusión, el giroscopio es un sensor de respuesta rápida y elevada precisión en tiempos cortos.

Por tanto, combinar las mediciones del acelerómetro y del giroscopio permiten al IMU determinar de manera más precisa la orientación si se compara con el empleo de ambos dispositivos por separado.

2.1.1.1. GY-521 MPU-6050

El modelo de sensor inercial [8] empleado es GY-521 MPU-6050 (ver *Figura 7)*. Se trata de un IMU (Unidad de Medición Inercial) de seis grados de libertad (6 DOF), que combina un acelerómetro y un giroscopio ambos de 3 ejes.

La comunicación puede realizarse tanto por SPI como por bus I2C, de modo que resulta sencillo obtener los datos medidos. La tensión de alimentación se encuentra entre 2.4 y 3.6V y, además, incluye un regulador de voltaje que permite alimentar directamente a 5V.

Dispone de convertidor analógico/digital de 16bits por canal para la digitalización de las salidas del acelerómetro. El rango del acelerómetro puede ajustarse a ±2g, ±4g, ±8g, ±16g, y el del giroscopio a ±250, ±500, ±1,000, ±2,000°/sec.

Este elemento consume 3.5 mA, con todos los sensores y el *DMP* (Procesador Digital de Movimiento o Digital Motion Processor) activados. Además dispone de un sensor de temperatura, un reloj de alta precisión e interrupciones programables y puede conectarse a otros dispositivos I2C como maestro (master).

El MPU-6050 incorpora un procesador interno DMP [9] (Procesador Digital de Movimiento) que ejecuta complejos algoritmos de MotionFusion para combinar las mediciones de los sensores internos (acelerómetro y giroscopio), evitando tener que realizar los filtros de forma exterior como, por ejemplo, el filtro complementario. Éste descarga el cálculo de los algoritmos de procesamiento de movimiento desde el procesador host. El cálculo de los algoritmos los realiza directamente en el chip lo cual hace que se reduzca la carga del Arduino.

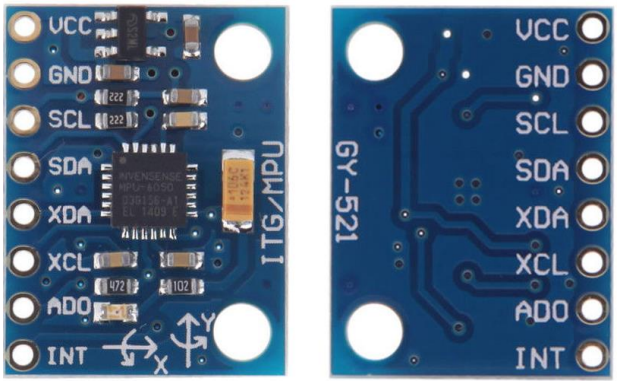


Figura 7. GY-521 MPU-6050. [10]

Para conseguir la iluminación y señalización del faro de la bicicleta se utiliza el DMP del MPU-6050.

A continuación, se muestran las conexiones del Arduino Nano con el IMU MPU-6050.

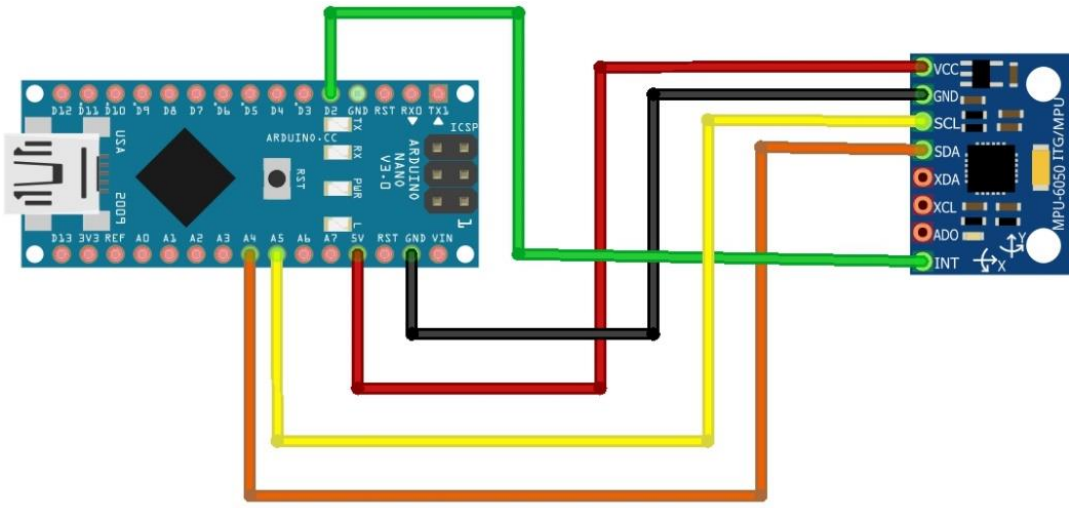


Figura 8. Esquemático GPS MPU-6050.

Arduino Nano	GY-521 MPU-6050
5V	VCC
GND	GND
A5	SCL
A4	SDA
D2	INT

Tabla 1. Conexiones Arduino Nano - MPU-6050

2.2. Aplicación móvil

El segundo propósito radica en el desarrollo de la aplicación móvil. Para ello se recurre a un módulo Bluetooth HC-05 y un GPS GY-NEO6MV2. El GPS obtendrá los valores de la latitud, longitud y altitud que el Bluetooth enviará a la app con el fin de obtener el recorrido realizado por la bicicleta. En la *Figura 9* se recoge el esquema completo del Arduino para el desarrollo de la app móvil.

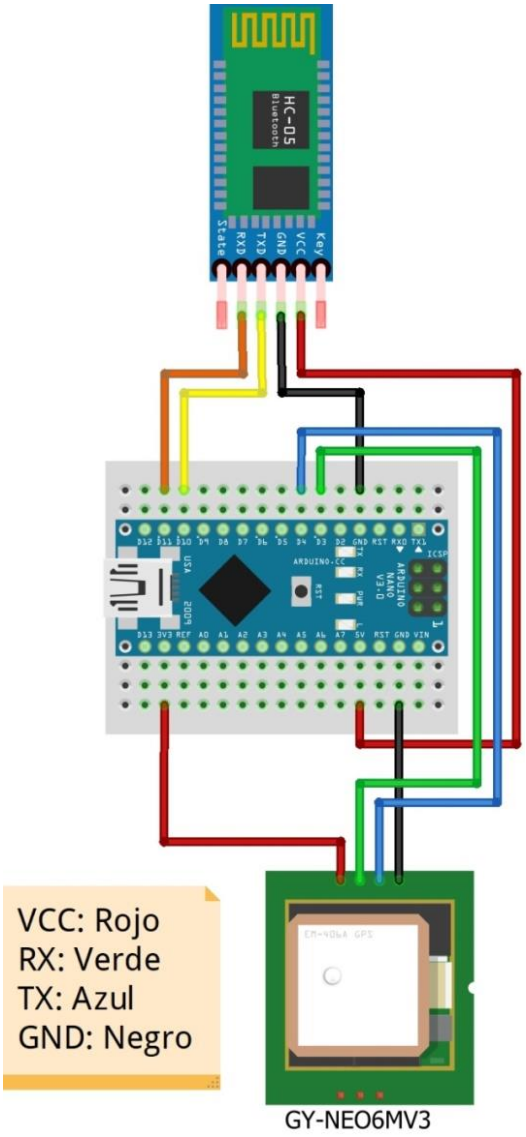


Figura 9. Esquema montaje Arduino para App móvil

2.2.1. GPS GY-NEO6MV2

El Sistema de Posicionamiento Global o Global Position System (GPS) [11] es un Sistema Global de Navegación por Satélite (GNSS) basado en 24 satélites en órbita sobre la Tierra que permite determinar la posición de una persona u objeto con una alta precisión.

Los satélites del GPS circundan la tierra dos veces al día en una órbita muy exacta y transmiten la información a la Tierra. Los receptores del GPS toman esta información y utilizan la triangulación para calcular la ubicación exacta del usuario. El receptor GPS compara el tiempo transcurrido desde que el satélite transmite una señal hasta que es recibida por dicho receptor. Esta diferencia de tiempo indica al receptor GPS la distancia a la que se encuentra el satélite y, realizando más mediciones con otros satélites, el receptor es capaz de determinar la posición (coordenadas) del usuario y mostrarlo en el mapa.

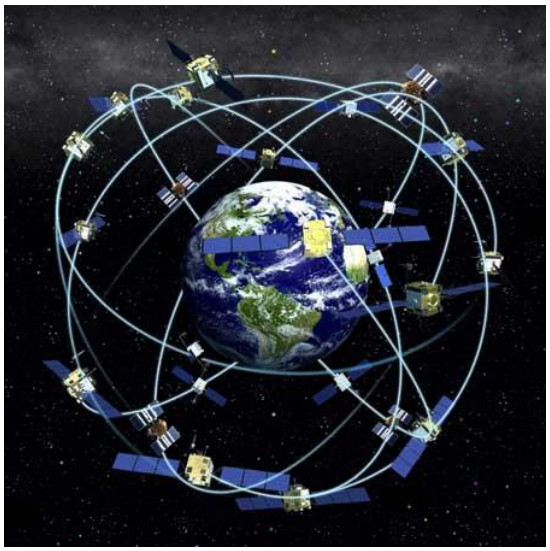


Figura 10. GPS-Satélites.

Un receptor GPS debe estar conectado a la señal de al menos tres satélites para calcular una posición 2D (latitud y longitud). Con cuatro o más satélites, el receptor puede determinar la posición 3D del usuario (latitud, longitud y altitud). Una vez establecida la posición del usuario, la unidad GPS puede calcular otra información, como velocidad, rumbo, precisión, pista, distancia de viaje, distancia al destino, hora de salida y puesta del sol, entre otras mediciones.

El dispositivo objeto de desarrollo permitirá obtener la posición 3D del usuario (latitud, longitud y altitud), determinándose la precisión y número de satélites a los que está conectado en cada momento.

Los dispositivos NEO-6 [12] son fabricados por U-Blox y pueden, como en este caso, conectarse fácilmente a un autómata o Arduino. Además disponen de interface de comunicación UART, SPI, DDC (I2C) y USB y soportan los protocolos NMEA, UBX binary y RTCM.

La tensión de alimentación es de 2.7 a 3.6V aunque puede alimentarse a una tensión de 5V al poseer un regulador integrado. La intensidad de corriente necesaria se encuentra alrededor de los 37 mA en modo continuo.

Figura 11. GPS NEO-6MV2. [13]

2.2.1.1. Características principales

- Receptor GPS independientes.
- Tiempo de encendido cold y warm de 30 segundos aproximadamente.
- Precisión en posición de 2.5m.
- Precisión en velocidad de 0.1 m/s.
- Precisión en orientación de 0.5º.
- SuperSense ® GPS interior: -162 dBm de sensibilidad de seguimiento.
- Tecnología anti-jamming.
- Soporte SBAS (WAAS, EGNOS, MSAS, GAGAN).

- Timepulse.
- Tasa de actualización de la posición 5 Hz.
- Temperatura de funcionamiento: -40 a 85 ºC.
- UART TTL zócalo.
- EEPROM para guardar los ajustes.
- Con antena GPS 18x18mm.
- Compatible con RoHS.

Model	Type					Supply		Interfaces				Features						
	GPS	PPP	Timing	Raw Data	Dead Reckoning	1.75 V - 2.0 V	2.7 V - 3.6 V	UART	USB	SPI	DDC (I²C compliant)	Programmable (Flash) FW update	TCXO	RTC crystal	Antenna supply and supervisor	Configuration pins	Timepulse	External interrupt/ Wakeup
NEO-6G	•					•		•	•	•	•		•	•	o	3	1	•
NEO-6Q	•						•	•	•	•	•		•	•	o	3	1	•
NEO-6M	•						•	•	•	•	•			•	o	3	1	•
NEO-6P	•	•		•			•	•	•	•	•			•	o	3	1	•
NEO-6V	•				•		•	•	•	•	•			•	o	3	1	•
NEO-6T	•		•	•			•	•	•	•	•		•	•	o	3	1	•

o = Requires external components and integration on application processor

Figura 12. Características GPS NEO-6M. [14]

2.2.1.2. Esquemático

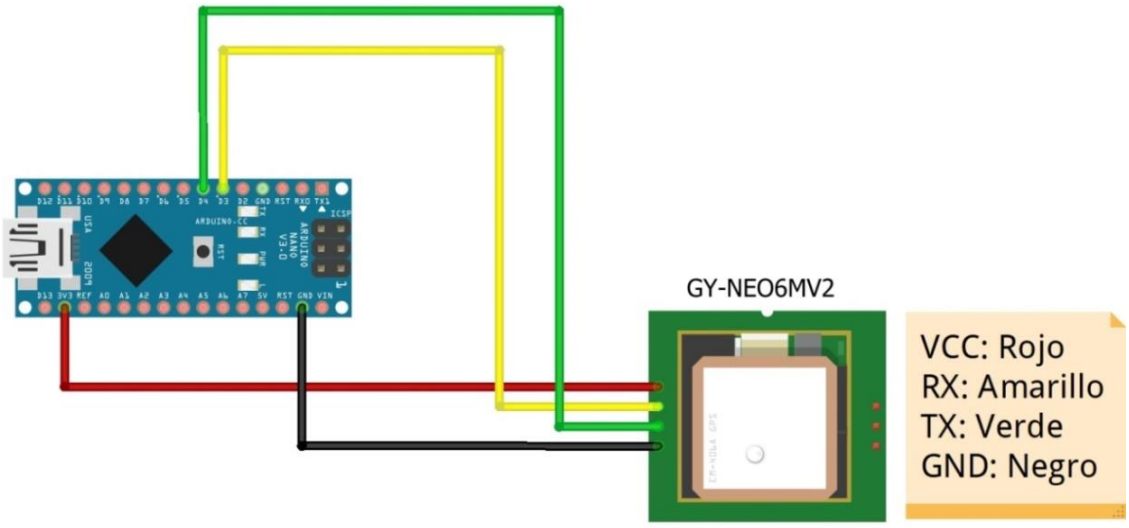


Figura 13. Esquemático GPS NEO-6M.

Arduino Nano	GY-521 MPU-6050
5V	VCC
D3	RX
D4	TX
GND	GND

Tabla 2. Conexiones Arduino Nano – GPS NEO-6M.

2.2.2. Módulo Bluetooth HC-05

El Bluetooth [15] es un estándar de comunicación inalámbrica que permite la transmisión de datos entre diferentes dispositivos a través de radiofrecuencia en la banda de 2.4 GHz. Estos dispositivos se clasifican en tres clases: “Clase 1”, “Clase 2” o “Clase 3” en función de su potencia de transmisión.

La especificación de Bluetooth define un canal de comunicación a un máximo de 720 kbit/s (1 Mbit/s de capacidad bruta) con rango óptimo de 10 m (opcionalmente 100 m con repetidores). Opera en la frecuencia de radio de 2.4 a 2.48 GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir en Full Duplex con un máximo de 1600 saltos por segundo. Los saltos de frecuencia se dan entre un total de 79 frecuencias con intervalos de 1 MHz, lo cual dota de seguridad y robustez.

El hardware que compone el dispositivo Bluetooth consta de dos partes:

- Un dispositivo de radio, encargado de modular y transmitir la señal.
- Un controlador digital, compuesto por una CPU, un procesador de señales digitales (DSP – Digital Signal Processor) llamado Link Controller (o controlador de Enlace) y las interfaces con el dispositivo anfitrión.

Hay dos modos [16] de trabajo en los dispositivos Bluetooth: Maestro (Master) y Esclavo (Slave). La diferencia es que un esclavo solo puede conectarse a un maestro, mientras que un maestro puede conectarse a varios esclavos (hasta un máximo de 7) y recibir y solicitar información de todos ellos.

Cada uno de los dispositivos que se identifican vía Bluetooth presenta una dirección única de 48 bits, un nombre de dispositivo para identificarlo y un PIN de conexión que debe teclearse para tener acceso al mismo.

Si el emparejamiento se realiza con éxito, ambos suelen guardar el número de identificación del otro y, cuando se encuentran a una distancia relativamente próxima, vuelven a vincularse sin necesidad de emparejarlos de nuevo con su PIN de identificación.

El módulo HC-05 [17] (ver *Figura 14)* permite conectar y comunicar con Arduino vía Bluetooth. Se configura mediante comandos AT y puede hacerlo funcionar tanto en modo maestro como esclavo.

El módulo Bluetooth HC-05 puede alimentarse con una tensión de entre 3.3 y 6V (normalmente 5V). Dispone de un pulsador para entrar en modo comandos, aunque también puede hacerse a través de software mediante el pin EN.

Incorpora un LED que indica el estado de la conexión y si se ha efectuado el emparejamiento en función de la velocidad de parpadeo.

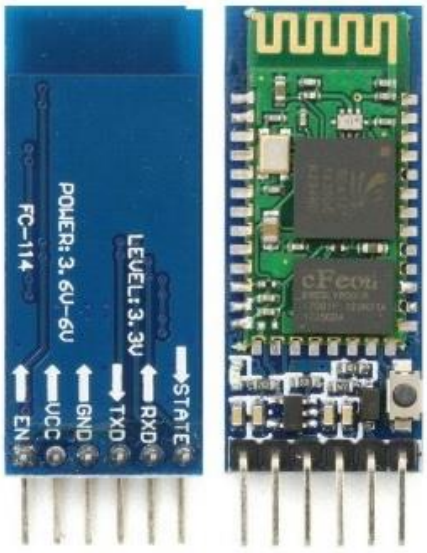


Figura 14. Módulo HC-05. [18]

2.2.2.1. Características principales

- Protocolo Bluetooth: v1.1 / 2.0.
- Frecuencia: banda ISM de 2.4 GHz.
- Modulación: GFSK
- Potencia de transmisión: menos de 4dBm, Clase 2.
- Ratio de asíncronos: 2.1 Mbps (Max) / 160 kbps.
- Perfiles de ayuda: puerto serie Bluetooth (maestro y esclavo).

2.2.2.2. Estado del HC-05

El módulo HC-05 tiene cuatro modos de funcionamiento [19]:

1. Estado Desconectado:

    - Se entra en este estado al alimentar el módulo y cuando no se ha establecido una conexión Bluetooth con otro dispositivo.
    - El led del módulo parpadea rápidamente.
    - En este estado no se pueden interpretar los comandos AT.

2. Estado Conectado o de Comunicación:

    - Para entrar a este estado ha de realizarse una conexión Bluetooth con otro dispositivo.
    - El led emite un doble parpadeo.
    - Todos los datos que se ingresen al módulo HC-05 por el pin RX se trasmiten por Bluetooth al dispositivo conectado. Por su parte, los datos recibidos se devuelven por el pin TX.

3. Modo AT 1:

    - Para acceder a este estado es necesario alimentar el módulo y pulsar el botón de encendido.
    - En este estado, se pueden enviar comandos AT aunque a la misma velocidad a la que está configurado.

- El led del módulo parpadea rápidamente de igual forma que en el estado desconectado.

4. Modo AT 2:

- Se requiere mantener pulsado el botón de encendido antes de alimentarlo para entrar en este estado. Una vez se ha encendido, el módulo permanecerá en dicho estado al dejar de pulsar el botón.
- Para enviar comandos AT es necesario hacerlo a la velocidad de 38,400 baudios.
- El led del módulo en este estado parpadea lentamente.

2.2.2.3. Comandos AT

De fábrica el nombre del módulo es HC-05, su contraseña 1234 y su configuración es en modo esclavo. El módulo puede usarse con esta configuración, aunque para modificar estos valores se precisa utilizar los comandos AT.

Para poder utilizar los comandos se carga el sketch de la *Figura 15* en Arduino:

Figura 15. Sketch comandos AT.

A continuación, se abre el Monitor Serial del IDE de Arduino y en la parte inferior se elige la opción de “Ambos NL & CR” a velocidad de 38,400 baud (velocidad para comunicarse con el HC-05). Por último, se escriben los comandos [19] y “Enviar”.

Los comandos más utilizados para comunicarse con el módulo HC-05 son:

- Comprobar si el Bluetooth responde: AT.
- Cambiar nombre del módulo: AT+NAME=<Nombre>.
- Conocer el nombre del módulo: AT+NAME?.
- Cambiar código de vinculación: AT+PWSD=<PIN>.
- Mostrar el PIN actual del módulo: AT+PWSD?.
- Configurar la velocidad de comunicación: AT+UART=<Baud>, <StopBit>,<Parity>.
- Conocer la configuración actual: AT+UART?.
- Configurar el Role para que trabaje como maestro o esclavo: AT+ROLE=<Role>.
    - 0 -> Esclavo.
    - 1 -> Maestro.
- Para saber el modo de trabajo actual: AT+ROLE.
- Configurar el modo de conexión: AT+CMODE=<Mode>.
    - 0 -> Conectarse a un dispositivo con la dirección especificada.
    - 1 -> Conectar el módulo a cualquier dirección disponible (aleatorio).
- Para saber el modo de actual de conexión: AT+CMODE?.
- Especificar la dirección del dispositivo al que se va a conectar: AT+BIND=<Address>.
- Conocer la dirección actual del módulo: AT+BIND?.
- Obtener la versión del firmware: AT+VERSION?.
- Obtener la dirección del módulo: AT+ADDR?.
- Resetear el módulo: AT+RESET.
- Restablecer valores por defecto: AT+ORGL.

2.2.2.4. Esquemático

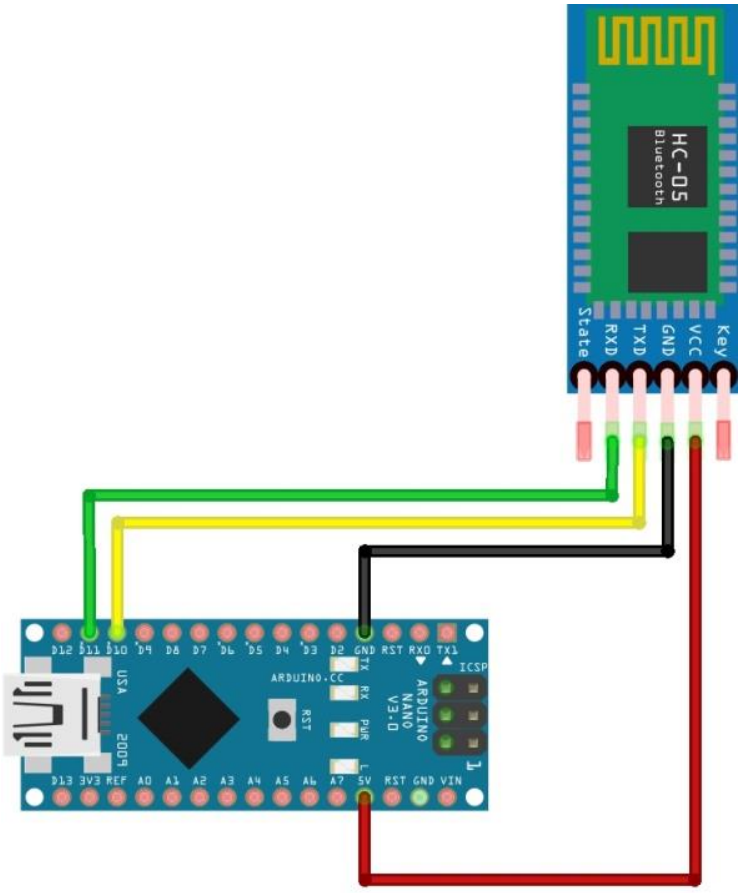


Figura 16. Esquemático HC-05.

Arduino Nano	HC-05
5V	VCC
GND	GND
D10	TXD
D11	RXD

Tabla 3. Conexiones Arduino Nano – HC-05.

3. SOFTWARE

3.1. Iluminación y señalización

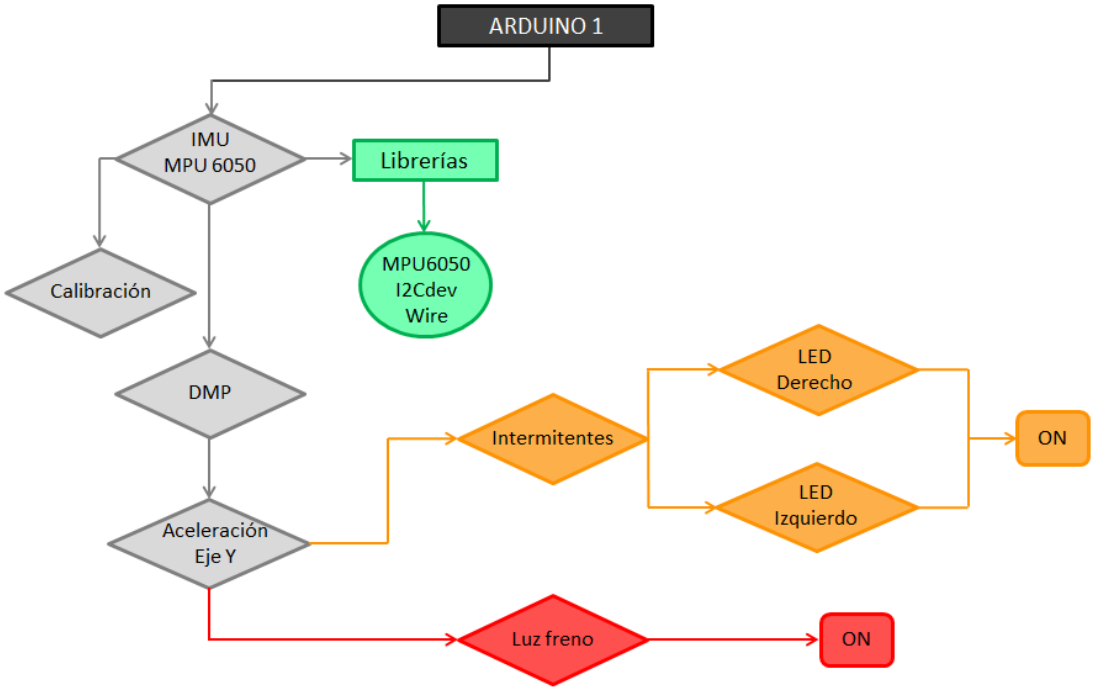


Figura 17. Flujograma Arduino para iluminación y señalización del faro (I).

Para obtener la iluminación y señalización del faro se va a utilizar un Arduino Nano con un IMU MPU-6050, 10 leds rojos para el freno y 2 leds amarillos para los giros a derecha y a izquierda. Las librerías empleadas pertenecen al IMU MPU6050 y son las siguientes: *"MPU6050.h"*, *"I2Cdev.h"* y *"Wire.h"*.

Antes de utilizar el sketch para la obtención de la iluminación y señalización, compuesto principalmente por el DMP, han de determinarse los valores de calibración del IMU (ver *Figura 18*), necesarios para la implementación del sketch del DMP. Los valores utilizados en el Sketch del DMP son:

- GiroX: 260.
- GiroY: 186.
- GiroZ: 2.
- AcelZ: 541.

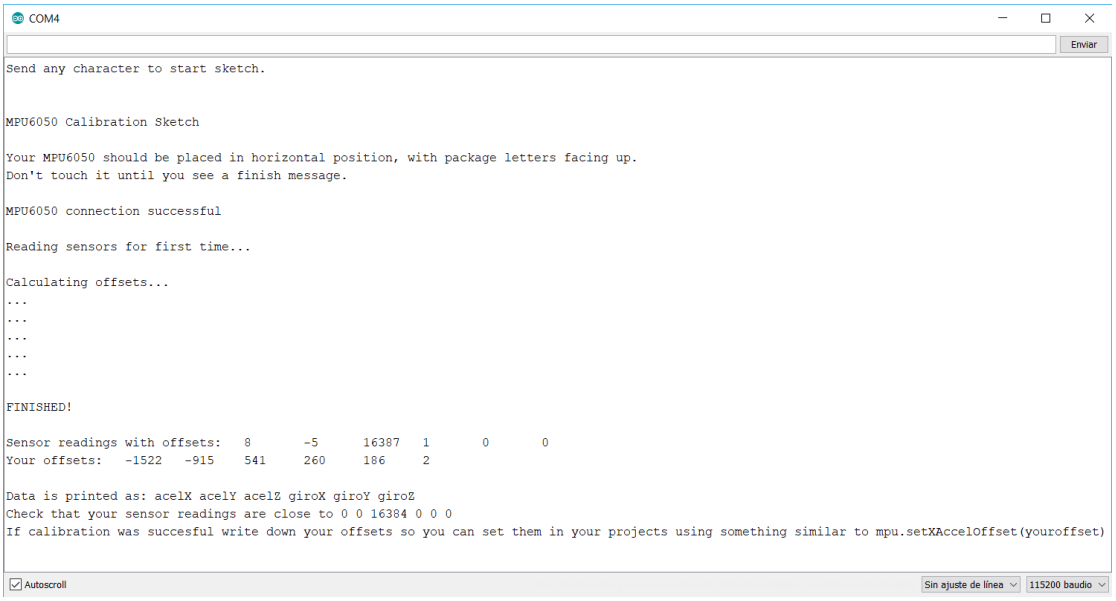


Figura 18. Valores calibración

El proceso completo se muestra en detalle en la *Figura 19*. En primer lugar, se abre el puerto serie y se fija la velocidad en baudios (115,200) para la transmisión de los datos en serie y *Wire.begin()* que permite la comunicación I2C con el IMU.

A continuación, se comprueba si el IMU se ha iniciado correctamente y, de ser así, ha de activarse el DMP (Procesador Digital de Movimiento). Tal y como se ha comentado anteriormente, se utiliza el DMP del IMU para ejecutar algoritmos que combinan las mediciones del acelerómetro y giroscopio, obteniendo los ángulos y aceleraciones en los ejes X, Y, Z. Una vez compilada esta información, se obtienen continuamente valores de estas seis magnitudes, aunque la que verdaderamente cubre nuestras necesidades es la aceleración en el eje Y. Para encender el led derecho la aceleración en el eje Y debe ser mayor de 1100 y para el led izquierdo menor de -1100. Por último, para activar el led de freno se emplea la ecuación "Frenazo" de la *Figura 19,* que tiene en cuenta la aceleración del ciclo anterior para evitar la existencia de saltos demasiado grandes entre ellos, así como guardar una correlación entre todas las aceleraciones. Los valores se obtienen en valor absoluto y deberán ser superiores a 700.

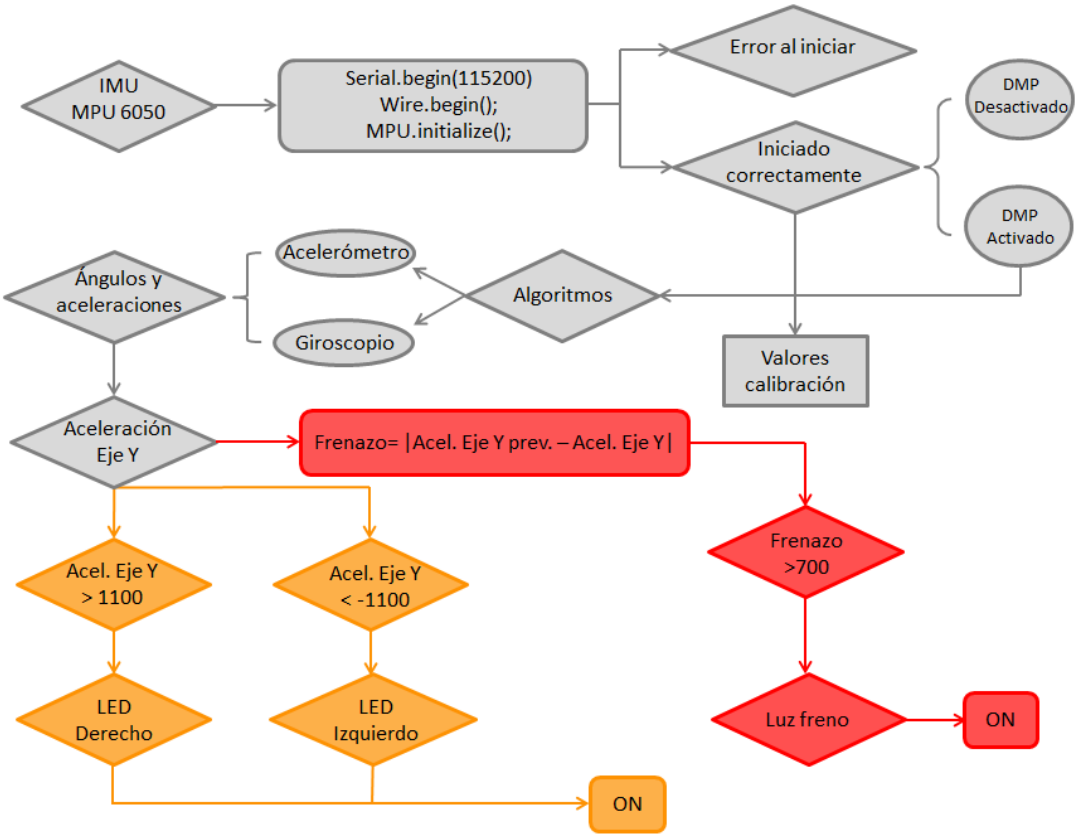


Figura 19. Flujograma Arduino para iluminación y señalización del faro (II).

3.2. Aplicación móvil

3.2.1. Software del microcontrolador

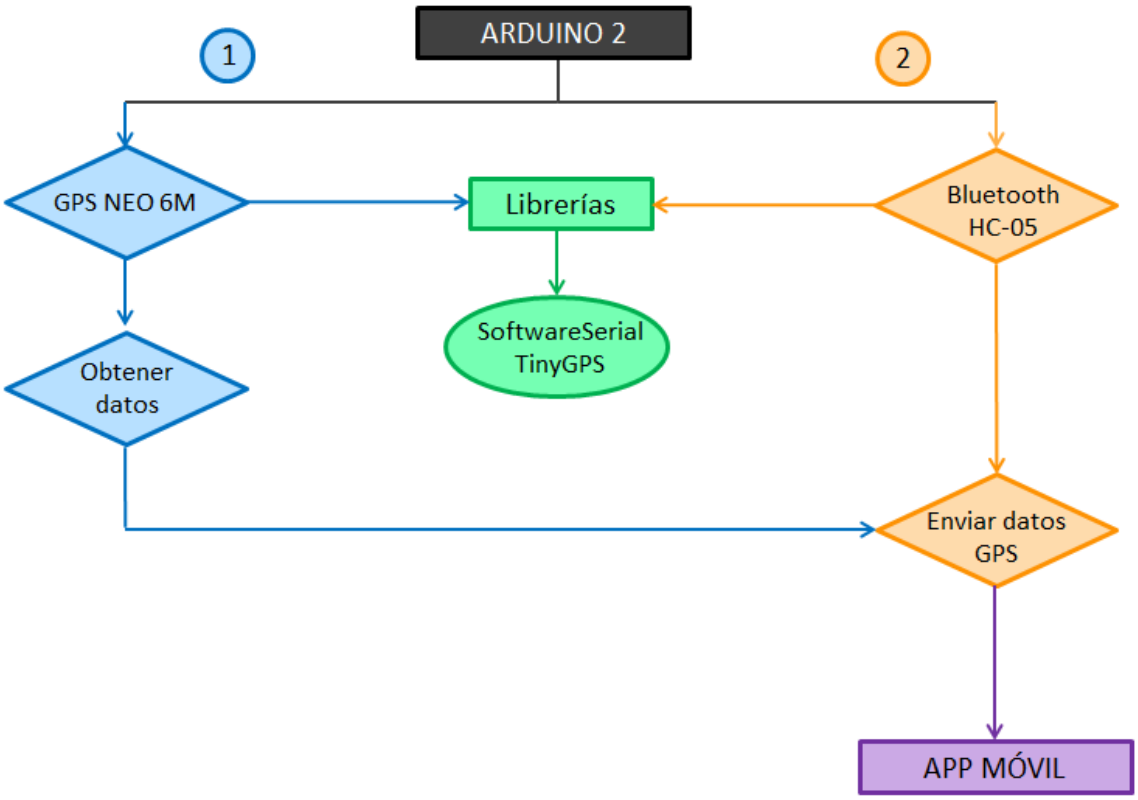


Figura 20. Flujograma Arduino para App móvil (I).

Para desarrollar la aplicación móvil, en primer lugar se recurre a un Arduino Nano con un módulo Bluetooth HC-05 para enviar los datos (latitud, longitud y latitud) obtenidos por el GPS NEO 6M. Las librerías utilizadas para el GPS son <*TinyGPS.h*> y <*SoftwareSerial.h*> para la comunicación serie en los pines, empleándose también esta última para el Bluetooth HC-05. Posteriormente, se programa internamente la aplicación para obtener el recorrido de la bicicleta.

Inicialmente se comprueba que el GPS se ha iniciado correctamente y, de ser así, se inicia la comunicación serie *(SoftwareSerialGPS.begin(9600))*. A continuación, el GPS toma un segundo para recibir señal, y una vez conseguida, pueden obtenerse todos los valores (latitud, longitud, altitud, precisión y número de satélites) o que el dispositivo no encuentre satélites. Aquí terminaría la comunicación serie del GPS y comenzaría la comunicación del módulo HC-05

(SoftwareSerialBT.begin(9600)). El HC-05 puede enviar “No hay satélites” si el GPS no los encuentra o enviar los datos en una cadena separados por este símbolo “|” (latitud|longitud|altitud|precisión|satélites). Se utiliza este símbolo para que, una vez desarrollada la app móvil, se guarde en cada posición del vector uno de los datos, para lo cual se empleará la función “Split” que permite cortar cada valor por dicho símbolo (por ejemplo, |latitud| - posición 0 del vector). Finalmente se termina la comunicación serie del Bluetooth y se repite el proceso en el loop del Arduino.

El hecho de utilizar una comunicación serie para cada dispositivo es debido a que si se utiliza la misma para ambos, estos colisionarían dejando de funcionar uno de ellos.

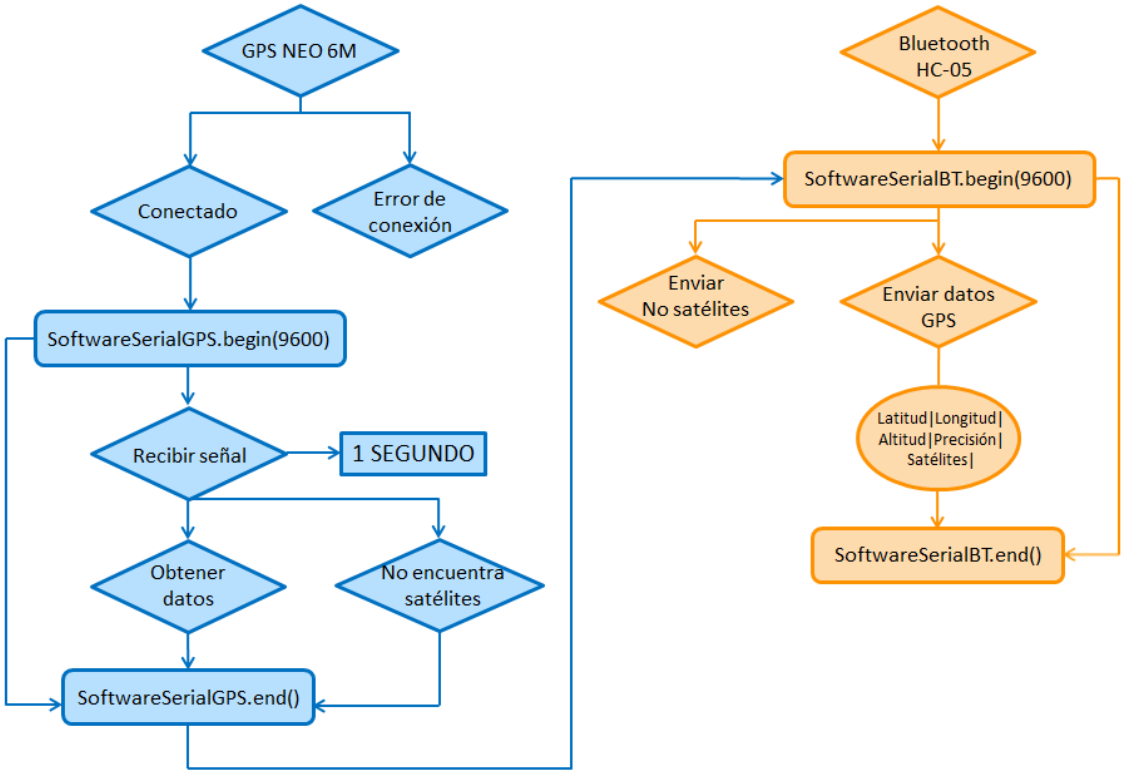


Figura 21. Flujograma Arduino para App móvil (II).

3.2.2. Aplicación Android

3.2.2.1. Prototipo

A continuación, se presenta una idea inicial de la estructura de la aplicación a desarrollar.

La pantalla de inicio de la app será la siguiente:

Figura 22. Pantalla de inicio.

La pantalla principal dispondrá de dos botones, “conectar” y “mapa directo”.

- Conectar: al pulsar este botón conectará con el Bluetooth HC-05 del Arduino.
- Mapa directo: al pulsar este botón nos direccionará a una nueva ventana.

Además, debajo del botón de conectar, emergerá un texto indicando si la conexión con el dispositivo Bluetooth se ha realizado con éxito.

Al pulsar el *botón mapa directo* se accede a una nueva pestaña que permite visualizar un mapa donde se irá generando cada 'x' tiempo una coordenada a tiempo real. Dicha coordenada ofrecerá la información de latitud, altitud, longitud, precisión y satélites. Esta sucesión de puntos generará el recorrido realizado por la bicicleta.

Figura 23. Mapa en directo.

3.2.2.2. Apache Cordova

Apache Cordova es un marco de desarrollo móvil de código abierto que permite usar tecnologías web estándar: HTML5, CSS3 y JavaScript para el desarrollo multiplataforma. Las aplicaciones se ejecutan dentro de envolturas dirigidas a cada plataforma, y dependen de los enlaces API (Interfaz de Programación de Aplicaciones o Application Programming Interface) que cumplen con los estándares

para acceder a las capacidades de cada dispositivo, como sensores, datos, estado de la red, etc.

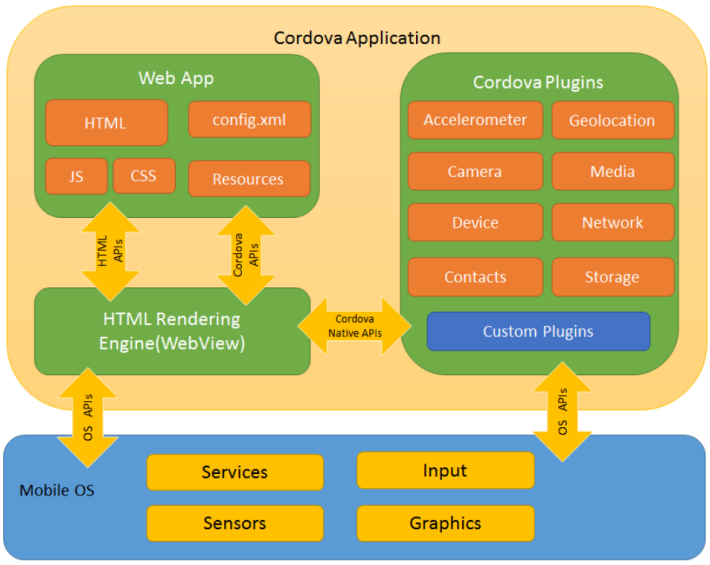


Figura 24. Arquitectura Apache Cordova. [20]

La instalación de Apache Cordova se subdivide a su vez en tres intalaciones: en primer lugar, Cordova; a continuación, Java; por último, Android Studio.

Para instalar Cordova se seguirán los siguientes pasos:

1) Instalar Node.js. Cordova se ejecuta en la plataforma Node.js, que debe instalarse como primer paso.

Node.js® es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos.

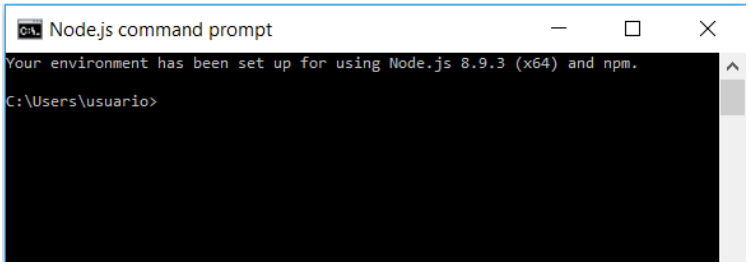


Figura 25. Plataforma Node.js.

2) Instalar Cordova. Cordova se instala utilizando el Administrador de paquetes de nodo (npm). Para ello se escribe `npm install -g cordova` en la ventana de comandos.

Además de Cordova, para el desarrollo de aplicaciones Android se requiere la instalación de Android SDK y Java.

En segundo lugar, se instala *Java*. Android SDK precisa la instalación de *Java Development Kit* (JDK).

1) Descargar Java Standard Edition JDK para “Windows x64” (Windows de 64 bits).

2) Al instalar, se seleccionan las opciones predeterminadas y se toma nota del directorio en el que se instala JDK para agregarlo a la ruta en una etapa posterior.

3) A continuación, se actualiza la ruta para incluir JDK. Se accede al *Panel de control/Sistema y Seguridad/Sistema/Cambiar configuración*, que abrirá la ventana de *Propiedades del sistema*. Se selecciona la pestaña de Opciones avanzadas, y se hace clic en el botón de *Variables de entorno*, como se puede apreciar en la *Figura 26*.

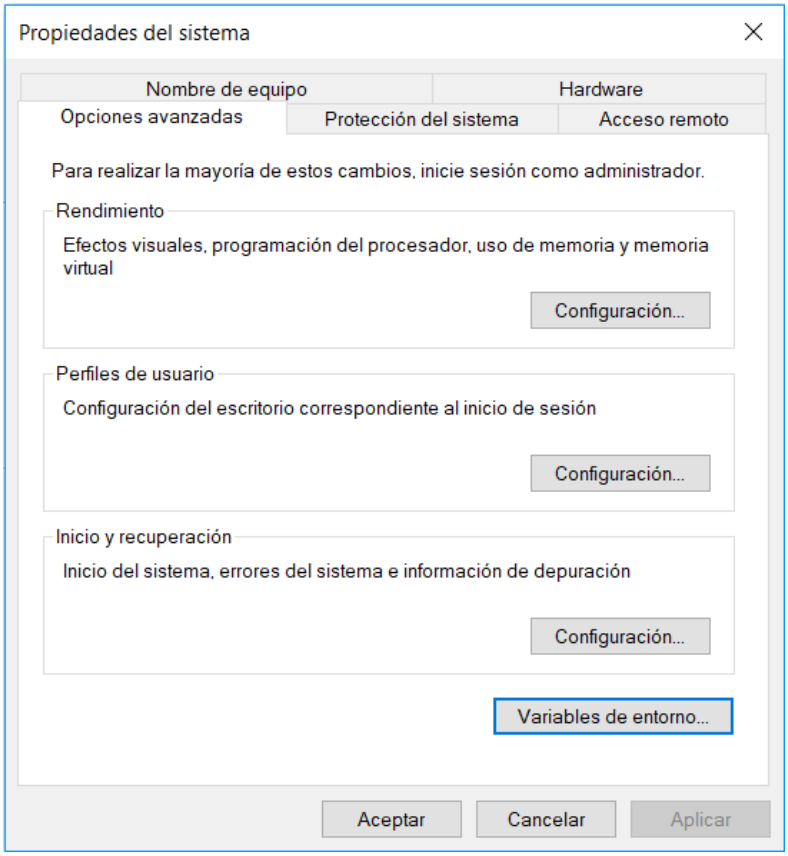


Figura 26. Propiedades del sistema.

4) En la lista *Variables del sistema* se seleccione la ruta y se hace clic en el botón *Editar*. Si no hay una entrada PATH en la lista, pulsar el botón Nuevo para crear una. Al final del campo, se agrega un punto y coma seguido de la ruta del directorio bin de la instalación JDK.

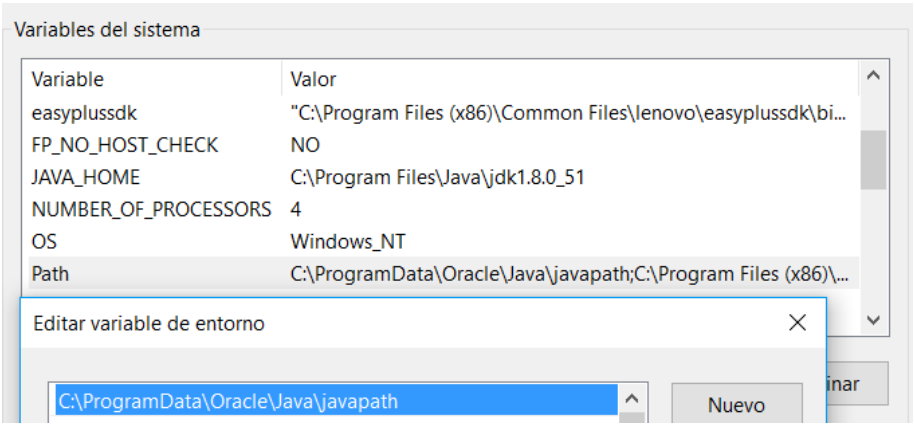


Figura 27. PATH_JAVA.

5) Por último, se agrega la variable JAVA_HOME si no está presente (si lo está, es posible que haya que actualizar su valor mediante la opción *Editar*). Se pulsa en *Nuevo* y en el campo *nombre de la variable* se indica JAVA_HOME. (Ver *Figura 28*.)

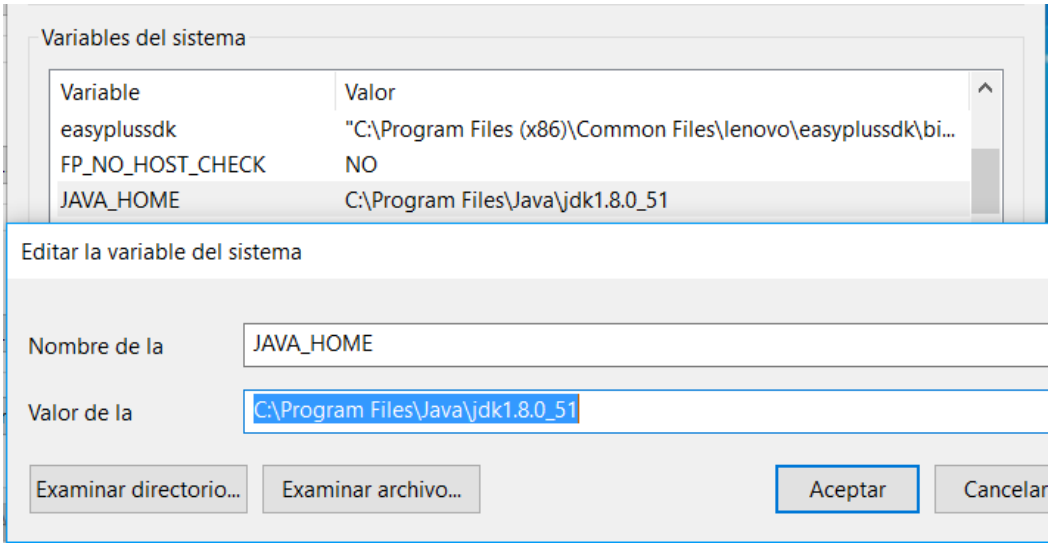


Figura 28. JAVA_HOME.

Por último, se instala Android Studio. Para instalar las herramientas que permitan construir la aplicación de Android con Cordova, es necesario recurrir a las herramientas de Android SDK.

1) Descargar e instalar Android Studio.

2) Agregar la ruta de las herramientas SDK (herramientas de directorios y herramientas de plataforma a la variable PATH del sistema). Se accede al *Panel de control/Sistema y Seguridad/Sistema/Cambiar configuración*, que abrirá la ventana de *Propiedades del sistema*. Se selecciona la pestaña Opciones avanzadas y se hace clic en el botón de *Variables de entorno*, como se muestra en la *Figura 26*.

3) En la lista *Variables del sistema* seleccioner ruta y clicar en *Editar*. Al final del campo, agregar un punto y coma seguido de la ruta a las herramientas de directorios y herramientas de plataforma de la instalación de Android SDK.

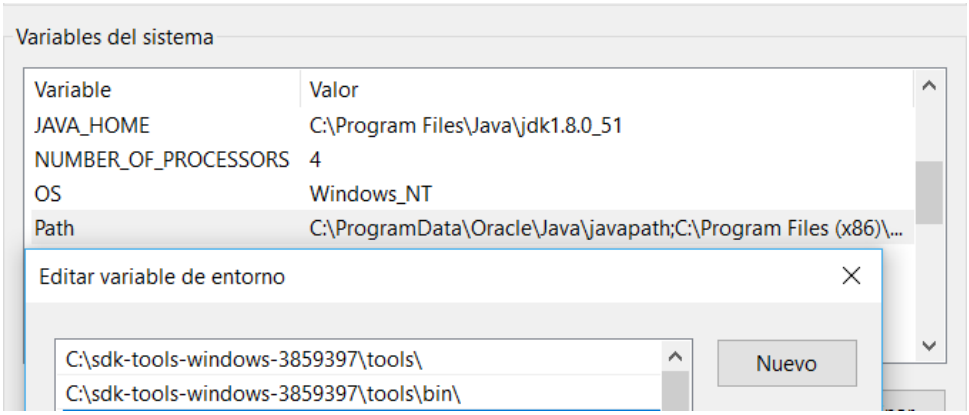


Figura 29. PATH_SDK.

4) Por último, se agrega la variable de entorno de ANDROID_HOME en la configuración del sistema, tal y como se llevó a cabo con la variable JAVA_HOME.

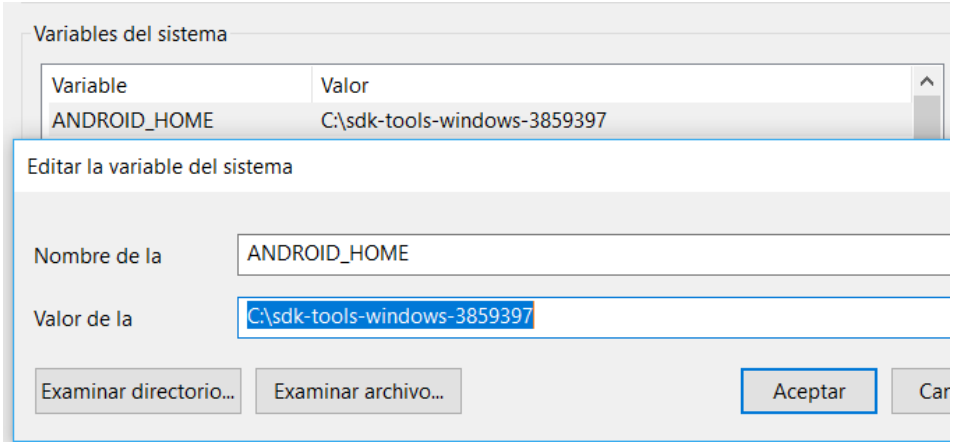


Figura 30. ANDROID_HOME.

Por medio de Node.js se realizan una serie de tareas fundamentales para generar la aplicación Android. En primer lugar, se crea la aplicación con el comando que se muestra en la *Figura 31*.

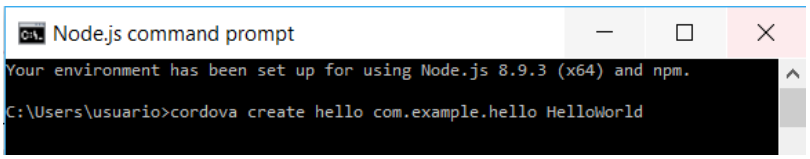


Figura 31. Crear aplicación. [21]

El primer argumento “*hello*” especifica un directorio que se genera para nuestro proyecto. El segundo argumento “*com.example.hello*” proporciona un identificador de dominio para el proyecto. Este argumento es opcional, siempre y cuando también se omita el tercer argumento, ya que los argumentos son posicionales. Se puede editar este valor más adelante en el *config.xml.* Por último, el tercer argumento “*HelloWorld*” da el título de pantalla de la aplicación. Éste es opcional y se puede editar en el config.xml.

La aplicación se creó inicialmente con el nombre “app_bici”, aunque finalmente se modificó a través de config.xml. renombrándola como “UJABike”.

Seguidamente se añade una plataforma Android para poder trabajar en él. [21]

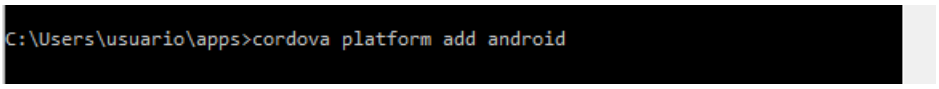


Figura 32. Crear proyecto.

Finalmente se construye la aplicación por medio del presente comando en node.js. [21]

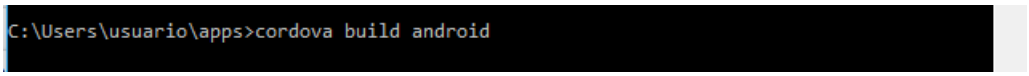


Figura 33. Construir aplicación.

3.2.2.3. NetBeans

Para desarrollar la aplicación se utiliza un programa de código abierto llamado *NetBeans*. NetBeans [22] es un entorno de desarrollo integrado libre, creado principalmente para el lenguaje de programación Java.

La plataforma NetBeans permite que las aplicaciones se desarrollen a partir de un conjunto de componentes de software llamados *módulos*. Un módulo es un archivo Java que contiene clases de JavaScript para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Puesto que los módulos pueden ser desarrollados de manera

independiente, las aplicaciones basadas en la plataforma NetBeans pueden ser fácilmente extendidas por otros desarrolladores de software.

3.2.2.4. PhoneGap

PhoneGap es [23] un framework para el desarrollo de aplicaciones móviles. Principalmente permite desarrollar aplicaciones para dispositivos móviles empleando herramientas como JavaScript, HTML5 y CSS3. Las aplicaciones resultantes serán híbridas, es decir, no son aplicaciones nativas al dispositivo (ya que el renderizado se realiza mediante vistas web y no con interfaces gráficas específicas de cada sistema), pero tampoco constituyen aplicaciones web (se trata de aplicaciones que son empaquetadas para poder ser desplegadas en el dispositivo incluso trabajando con el API del sistema nativo).

PhoneGap maneja API que permiten el acceso a elementos como el acelerómetro, la cámara, los contactos en el dispositivo, la red, el almacenamiento, las notificaciones, etc. Estas API se conectan al sistema operativo usando el código nativo del sistema huésped a través de una interfaz de funciones en JavaScript.

Además, permite el desarrollo ejecutando las aplicaciones en el navegador web (no se necesita un simulador dedicado a esta tarea), así como su visualización en el dispositivo móvil.

PhoneGap es una distribución de Apache Cordova. Ambos sistemas tienen funciones casi idénticas con la salvedad de que PhoneGap tiene acceso a servicios de compilación en la nube. Al tratarse de softwares de código abierto, tanto uno como otro pueden utilizarse libremente en cualquier aplicación sin necesidad de atribución o licencia de ningún tipo.

3.2.2.5. Ripple Emulator

Ripple Emulator permite ejecutar tu aplicación PhoneGap desde el navegador web ya que las API de JavaScript de PhoneGap están disponibles con Ripple.

3.2.2.6. Desarrollo de la aplicación móvil

En esta parte reside el código de la aplicación. La aplicación móvil [24] en sí misma se implementa como una página web, que hace referencia a CSS, JavaScript, imágenes, archivos multimedia u otros recursos necesarios para ejecutarse.

Este contenedor tiene un archivo muy importante - archivo *config.xml* que proporciona información sobre la aplicación y especifica los parámetros que afectan a su funcionamiento.

Config.xml es un archivo de configuración global [24] que controla numerosos aspectos del comportamiento de una aplicación Cordova. Este archivo .xml se organiza según las especificaciones de las aplicaciones web empaquetadas (widgets) del W3C, y se amplía para especificar las funciones, los complementos y las configuraciones específicas de la plataforma Core Cordova API.

Los elementos principales del archivo config.xml son:

- Atributo ***id*** del elemento <widget> proporciona el identificador de la aplicación y la versión de su número de versión completa.
- ***<description>*** y ***<author>*** especifican metadatos e información del contacto que puede aparecer en la app.
- ***<name>*** especifica nombre formal de la aplicación y cómo aparece en la pantalla principal del dispositivo.
- ***<plugin>*** permite acceder a características nativas del sistema operativo desde nuestro código JavaScript.
- ***<platform>*** especifica la configuración que sólo debe aparecer en un archivo específico de plataforma única config.xml.

```
<widget id="info.fernando.appbici" version="1.0.0" xmlns="http://www.w3.org/ns/widgets" xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>App Bici</name>
    <description>
        A sample Apache Cordova application that responds to the deviceready event.
    </description>
    <author email="dev@cordova.apache.org" href="http://cordova.io">
        Apache Cordova Team
    </author>
    <content src="index.html" />
    <plugin name="cordova-plugin-whitelist" spec="1" />
    <access origin="*" />
    <allow-intent href="http://*/*" />
    <allow-intent href="https://*/*" />
    <allow-intent href="tel:*" />
    <allow-intent href="sms:*" />
    <allow-intent href="mailto:*" />
    <allow-intent href="geo:*" />
    <platform name="android">
        <allow-intent href="market:*" />
    </platform>
    <platform name="ios">
        <allow-intent href="itms:*" />
        <allow-intent href="itms-apps:*" />
    </platform>
    <plugin name="cordova-plugin-mapbox" spec="^1.2.3">
        <variable name="ACCESS_TOKEN" value="pk.eyJ1IjoiYXJlaXEiLCJhIjoic2tGTkJtOCJ9.7GII_yFyIOZfkJPN00A0ZA" />
    </plugin>
</widget>
```

Figura 34. Ejemplo config.xml.

Para el desarrollo de la aplicación móvil cabe destacar también la bibiclioteca *jQuery* [25]. Se trata de una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos .html, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es software libre y de código abierto que posee una doble licencia (Licencia MIT y Licencia Pública General de GNU v2) que permite su uso en proyectos libres y privados jQuery. Ofrece una serie de funcionalidades basadas en JavaScript que, de otro modo, requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menor tiempo y espacio.

3.2.2.7. Plugins

Los *plugins* [26] son una parte integral del ecosistema de Cordova. Proporcionan una interfaz para Cordova, componentes nativos para comunicarse entre ellos y enlaces a API de dispositivos estándar. Esto le posibilita invocar código nativo desde JavaScript.

El proyecto de Apache Cordova mantiene un conjunto de complementos llamados Plugins que permiten acceder a características nativas del sistema operativo desde nuestro JavaScript.

Los plugins utilizados para el desarrollo de la aplicación móvil son:

- ***Mapbox Cordova Plugin***

Este plugin [27] permite integrar mapas. Se instala con CLI de Cordova (Command Line Interface), conocido como Interfaz de Línea de Comandos, a través de node.js:

```
cordova plugin add cordova-plugin-mapbox --variable ACCESS_TOKEN=pk.eyJ1IjoiYXJlaXEiLCJhIjoic2tGTkJtOCJ9.7GII_yFyIOZfkJPN00A0ZA
```

Para utilizar cualquiera de las herramientas, API o SDK de Mapbox, se requiere un token de acceso a Mapbox. Mapbox usa tokens de acceso para asociar solicitudes API con su cuenta y permite limitar el número de usos de los mapas por parte de los usuarios.

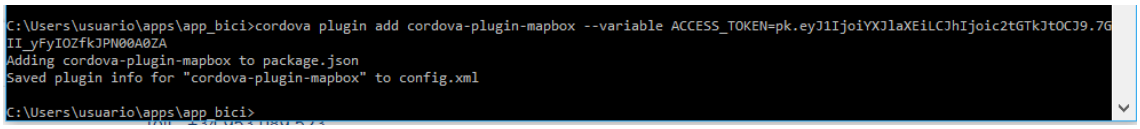


Figura 35. Instalación mapbox cordova plugin.

- ***Bluetooth Serial Plugin***

Este plugin [28] permite la comunicación en serie a través de Bluetooth. Se instala con CLI (Command Line Interface) de Cordova, a través de node.js:

```
cordova plugin add cordova-plugin-bluetooth-serial
```

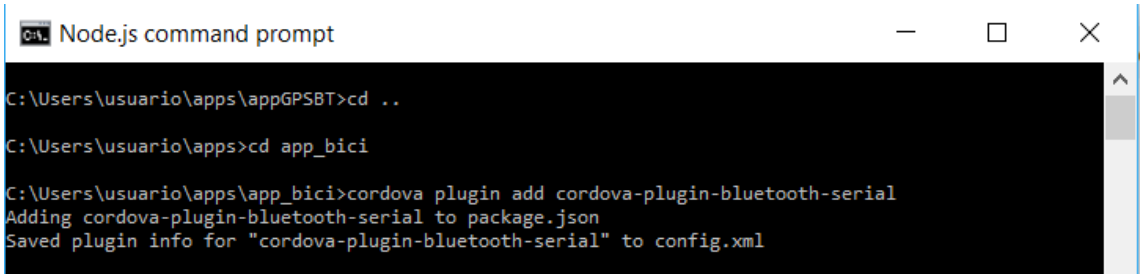


Figura 36. Instalación Bluetooth serial plugin.

- ***Splash-Screen Cordova Plugin***

Este plugin [29] muestra y esconde una pantalla de bienvenida durante el inicio de la aplicación. Se instala a través de node.js:

```
cordova plugin add cordova-plugin-splashscreen
```

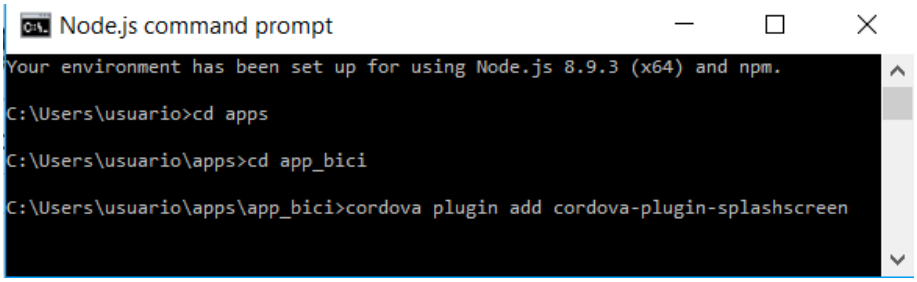


Figura 37. Instalación Splash-Screen plugin.

3.2.2.8. Flujograma

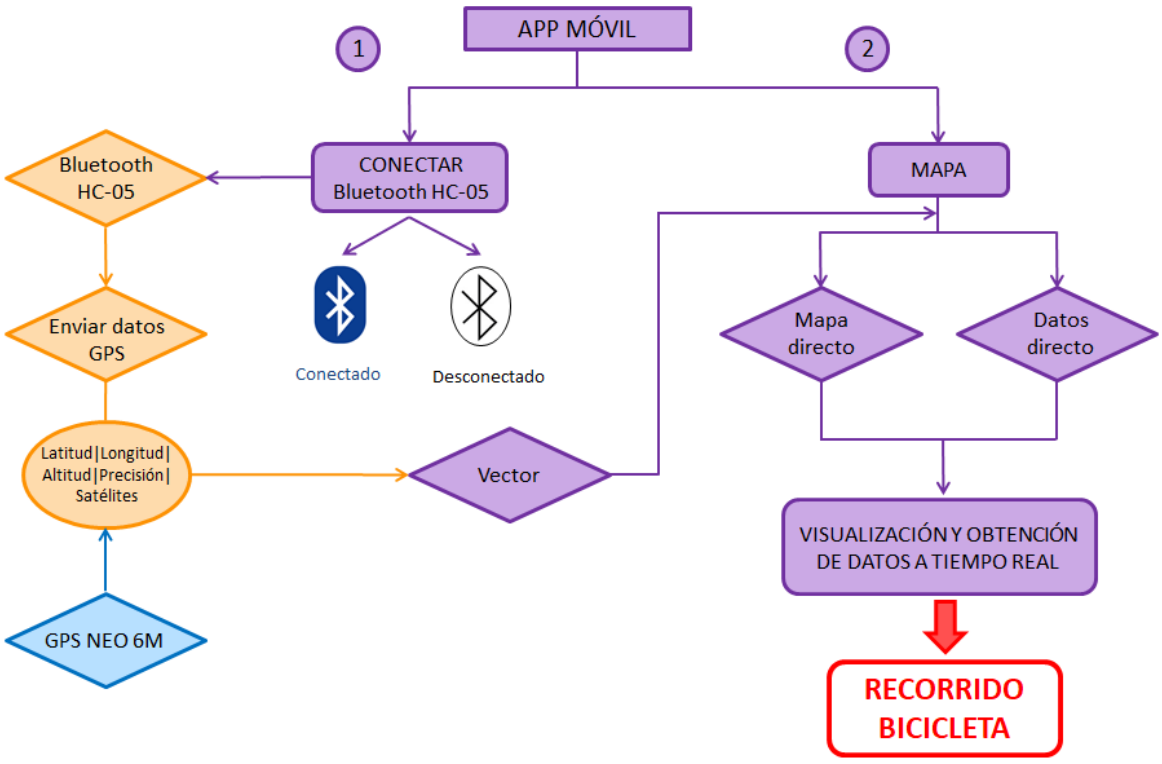


Figura 38. Flujograma APP móvil.

La aplicación móvil conecta con el Bluetooth HC-05, que se encarga de enviar los datos obtenidos por el GPS. Estos valores pueden visualizarse a tiempo real y mostrarse en el mapa trazando el recorrido de la bicicleta.

A continuación, se explican los archivos donde se realiza la programación de la app. Los tipos de archivos son: *.html* (HyperText Markup Language) que se encargan de la visualización de la aplicació; *.js* (JavaScript) responsables de la funcionalidad; *.css* (Cascading Style Sheets) proporcionan estilo a la app y *.xml* (eXtesible Markup Language) tratan la configuración de la misma.

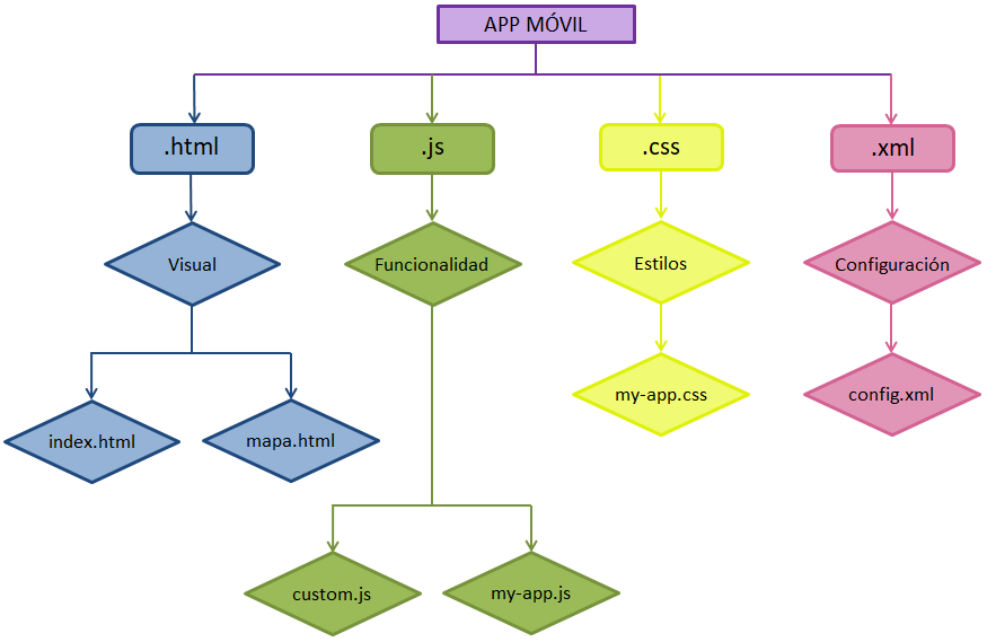


Figura 39. Documentos App.

Para comenzar con el desarrollo de la app se descargó una plantilla con estilos ya definidos. Partiendo de esta base, se fueron implementando modificaciones en cada uno de los archivos (ver *Figura 39*) para satisfacer las necesidades de la aplicación.

Los archivos .html constan de dos documentos: *index.html* y *mapa.html*. El index.html, a su vez, está dividido en <*head*> y <*body*>:

- El <head> es la estructura global del documento .html. Entre los distintos elementos destacan los <*links*>, enlaces a .css, y el <*title*>, inicialmente nombrado como “app_bici” al crear la aplicación Android.
- El <body> es el cuerpo de la aplicación, es decir, donde se ubica el contenido. A partir de * * se empieza a dar forma a la página principal de la aplicación móvil. Para ello se utilizan los diferentes comandos que se muestran en la *Figura 40*.

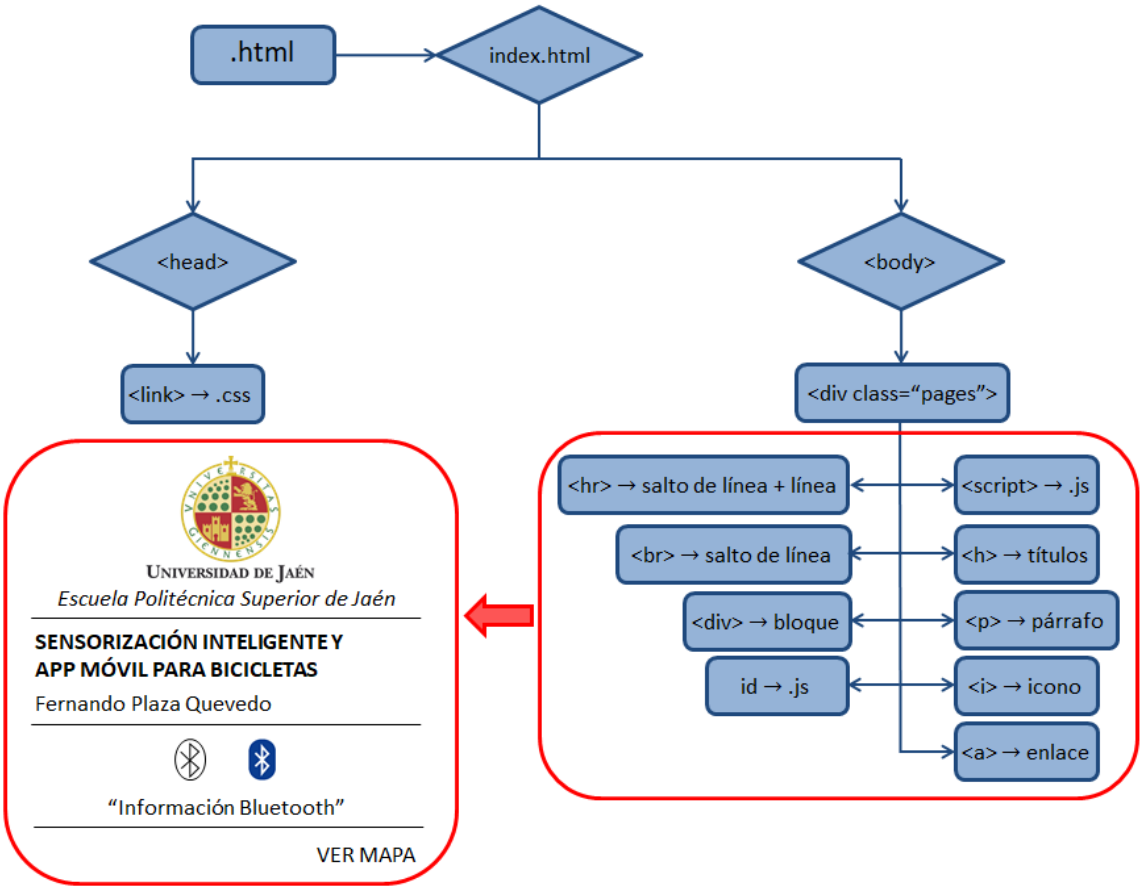


Figura 40. Index.html

A continuación, se elabora el documento mapa.html y se realiza la parte del mapa. Tal y como se aprecia en la *Figura 41*, este documento empieza en * * al igual que en el index.html. Esto se debe a que el contenido del index.html a partir de esa línea es sustituido por esta parte del mapa.

Llegados a este punto, se diferencian tres partes:

- ***navbar.*** Es la barra de inicio (arriba) de la página.
  - En la parte izquierda (left) aparece una flecha para ir hacía atrás (index.html).
  - En la parte central (center) emerge un texto "Mapa GPS" realizado con los comandos vistos anteriormente.
  - En la parte derecha (right) se indica el estado de conexión del Bluetooth. La parte del .js (funcionalidad) decide cuándo debe mostrarse cada uno de ellos.

- ***Mapa.*** Se enlaza con el mapa que se instalará en el documento custom.js.
- ***Datos Arduino.*** Se elaboran los textos recogidos en la *Figura 41* a través de los comandos <hr>, col-33, col-50, así como los anteriormente citados.

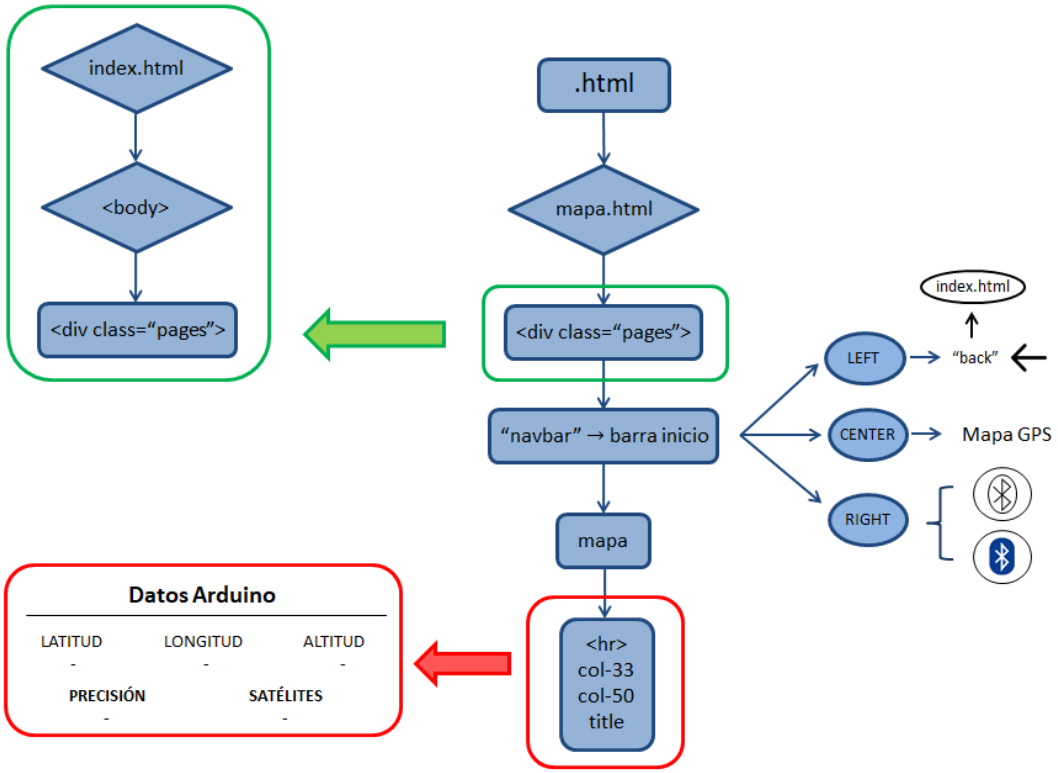


Figura 41. Mapa.html.

En segundo lugar, se trabaja con los archivos .js, constituidos por el *custom.js* y el *my-app.js*., en los cuales se basa la funcionalidad de la app y que, por tanto, representan el core del desarrollo de la app.

El custom.js está constituido principalmente por:

- ***disconnect.*** Envía a la función setStatus el texto "Desconectando…" y se utiliza el plugin del Bluetooth para ordenar que se desconecte con los siguientes parámetros: bluetoothSerial.disconnect(ondisconnect).
- ***onconnect.*** Envía el texto "Bluetooth Conectado" y el logo del Bluetooth conectado (azul) tanto a la página principal como a la del mapa, es decir,

a los .html (index.html y mapa.html). En los .html ya se hizo referencia a este hecho.

- ***ondisconnect.*** Envía a la función setStatus el texto "Bluetooth Desconectado" y el logo de Bluetooth desconectado (negro).
- ***setStatus.*** Envía la información del Bluetooth facilitada por el disconnect y el ondisconnect a los documentos .html.

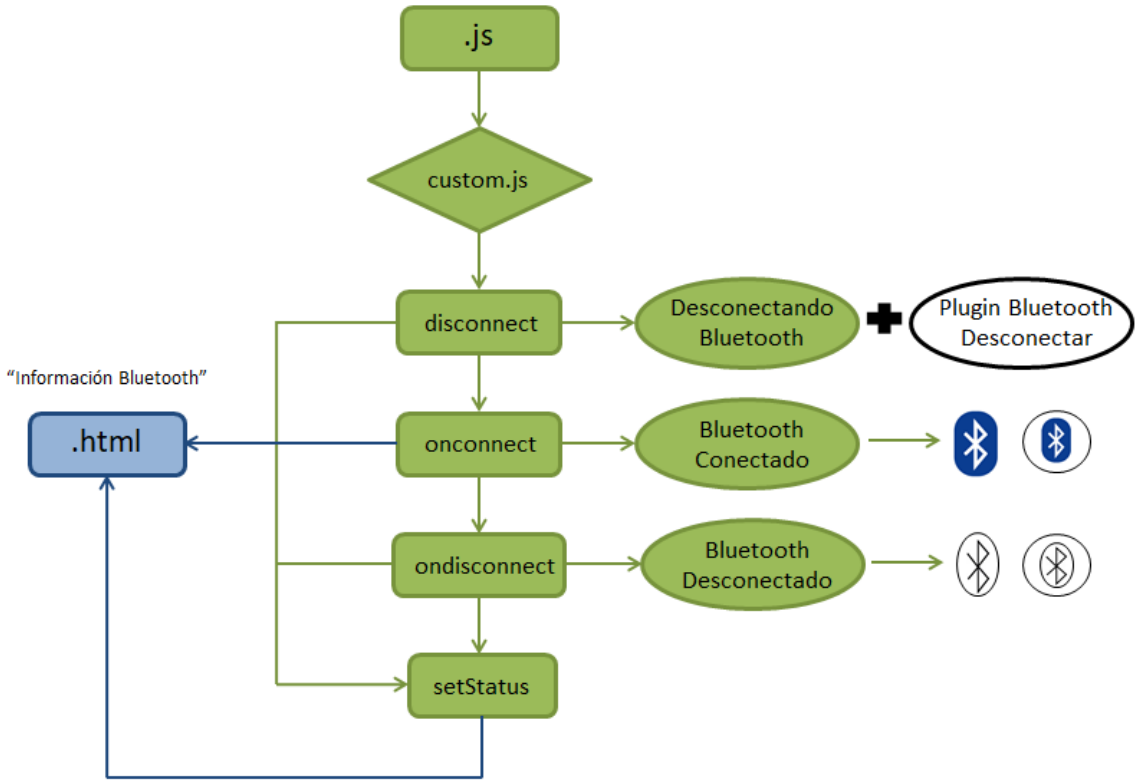


Figura 42. Custom.js

A continuación, se detalla el documento *my-app.js*.

- ***deviceready.*** Permite conocer si el dispositivo está operativo. Se indica la MAC del Bluetooth y envía "Conectando a MAC" a setStatus del custom.js, que lo manda, a su vez, al .html. Se utiliza, además, el plugin del Bluetooth (previamente instalado compilando en la consola de node.js) para conectarlo con el dispositivo de parámetros ya definidos en el custom.js *(bluetoothSerial.connect(MAC, onconnect, ondisconnect))*. El otro caso que puede tener lugar es que el dispositivo esté desconectado (disconnect).

- ***Page mapa.*** Se instala el mapa que irá incluido en el mapa.html.
- ***setInterval.*** Se utiliza para hacer repeticiones de instrucciones cada cierto tiempo. Primero, se generan las variables latitud, longitud, altitud, precisión y satélites, así como el vector, para, a continuación, enviarlos desde el Bluetooth con el formato siguiente: latitud|longitud|altitud|precisión|satélites. La función "Split" separará cada dato por el símbolo "|", disponiendo cada uno de ellos en una posición del vector. Por tanto, si el array es mayor 1, es decir, si hay más de un dato, significa que el GPS ha localizado satélites y está enviando todos los datos. En caso contrario, enviará "No hay satélites". Todos estos datos del Arduino se enlazan al documento mapa.html. A su vez, esta información está dentro de un setInterval, por lo que se va ejecutando en un intervalo de tiempo concreto. El intervalo de tiempo se ha definido en 5 segundos, es decir, en la app se mostrarán los datos cada 5 segundos, tanto en el mapa como en Datos Arduino.

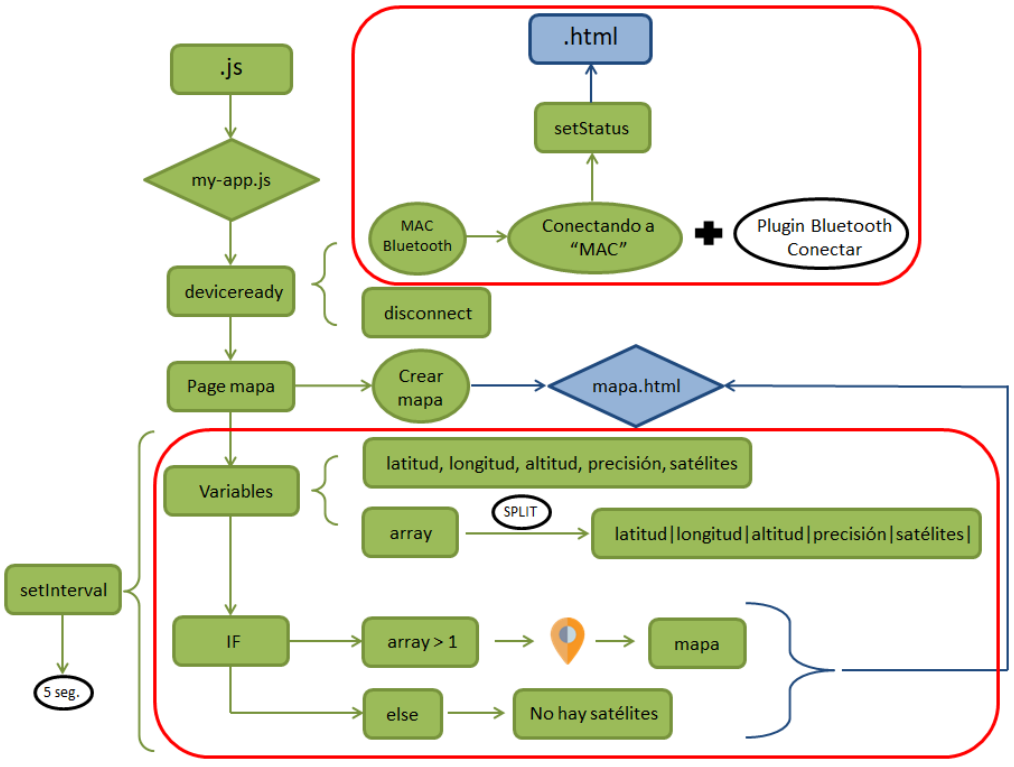


Figura 43. My-app.js

En tercer lugar, se hace referencia a los archivos .css y, en particular, al documento *my-app.css*. En él se definirán los estilos (posicionamiento, tamaños,

etc., como se aprecia en la *Figura 44*) de los textos de la información del Bluetooth, el logo del Bluetooth y del mapa.

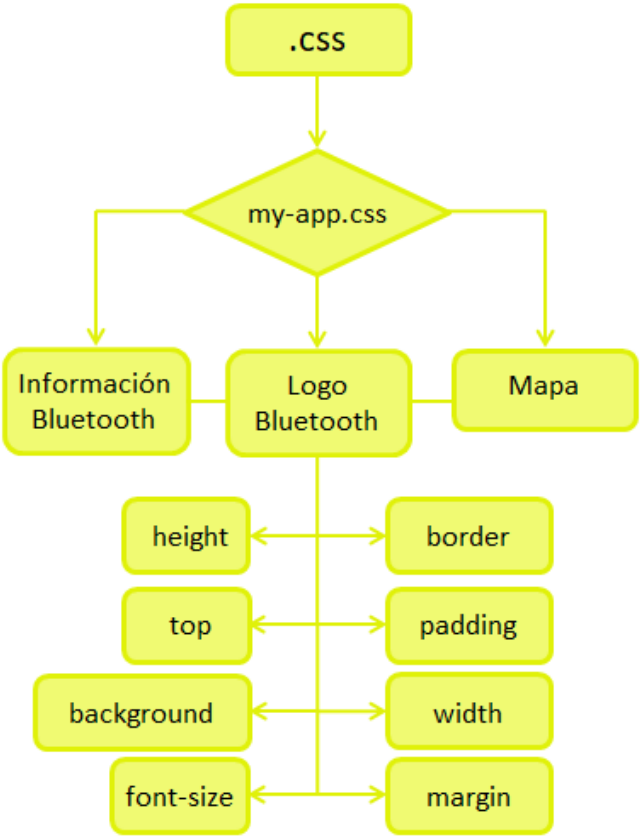


Figura 44. My-app.css

Por último, mencionar el archivo .xml. El documento que compone este archivo es el config.xml., que se genera automáticamente al compilar la aplicación desde la consola de node.js. Está constituido por diferentes elementos:

- ***Plugin.*** Las líneas de código se generan automáticamente al instalar los plugin (Bluetooth y mapa) por medio de node.js.
- ***Name.*** Al crear la aplicación desde node.js se decide nombrarla como "app_bici", nombre que aparece en el <title> de index.html. Sin embargo, el nombre que aparece en la app (UJABike) puede configurarse en el config.xml.
- ***Id.*** Es el identificador de la aplicación.

- ***Orientation.*** Se configura para que la app sólo pueda visualizarse en vertical en el dispositivo móvil. De esta forma se evita que los elementos se muevan si la orientación cambia a horizontal.
- ***Icono y Splash.*** Se decide utilizar dos imágenes, una para el icono de la app (Escudo Universidad de Jaén) y otra para el Splash (portada del Trabajo Fin de Grado). Para incluirlas se utiliza una página web (ver *Figura 45*) que genera automáticamente el código para el config.xml. Ambas imágenes, también generadas por la web, deben incluirse en la carpeta "res" de la aplicación.

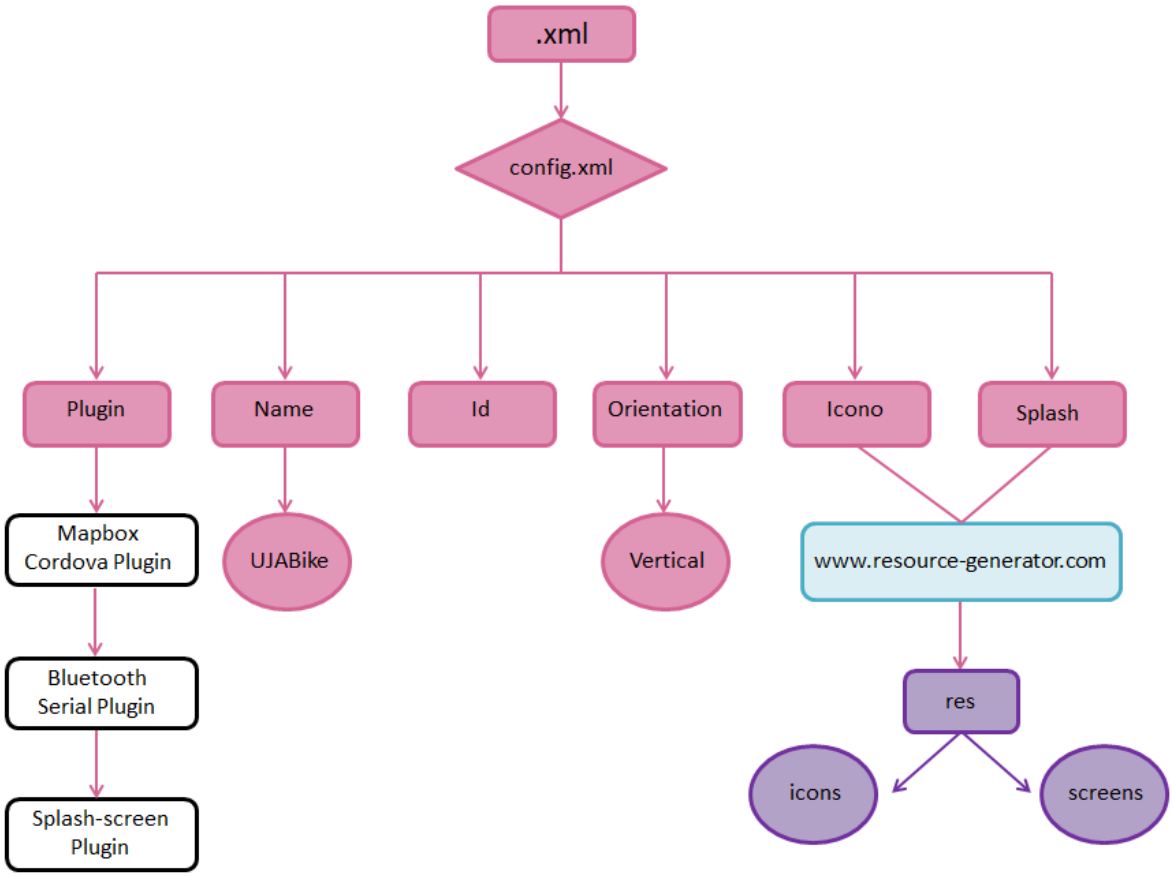


Figura 45. Config.xml

3.2.2.9 UJA Bike

Para disponer de la app UJABike en el móvil hay que descargar el archivo *app_debug.apk* e instalarlo. Este archivo se encuentra en la carpeta UJABike, ubicada, a su vez, en siguiente dirección: *UJABike/app_bici/plataforms/android/app/build/outputs/apk/debug*.

Antes de usar la app UJABike ha de emparejarse el dispositivo móvil con el Bluetooth HC-05, para lo cual basta con introducir el pin 1234.

El icono de la aplicación móvil es el escudo de la Universidad de Jaén.

Figura 46. Icono.

En el dispositivo móvil se mostrará conforme a la *Figura 47*.

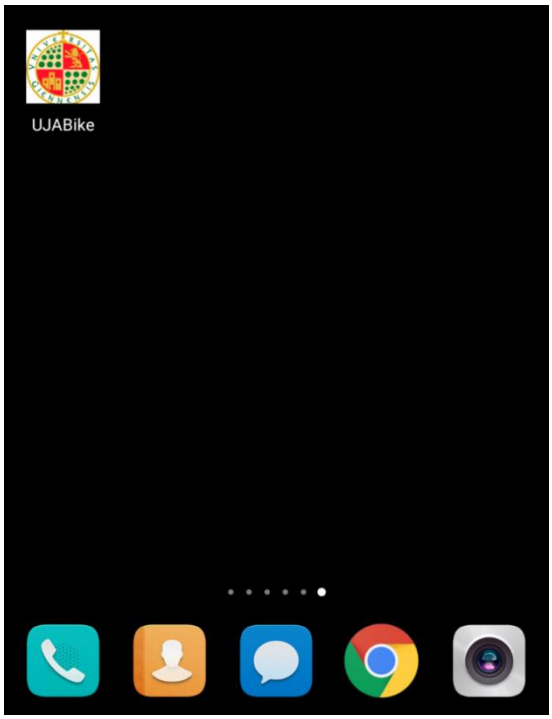


Figura 47. Pantalla móvil – Icono/Nombre.

En este momento es posible acceder a UJABike. Inicialmente aparecerá el Splash con la portada del Trabajo Fin de Grado durante 2 segundos.

Figura 48. Splash.

A continuación, aparecerá la página principal de la aplicación. En ella se distingue los logos tanto del Bluetooth desconectado como intentando conectar con el Bluetooth HC-05 mediante su MAC. En la parte inferior, el botón “VER MAPA” enlaza directamente con el mapa en directo.

Figura 49. Principal.

A partir de la *Figura 49* , existen dos escenarios posibles:

- El Bluetooth HC-05 esté desconectado.
- El Bluetooth HC-05 esté conectado.

En la *Figura 50*, el Bluetooth está desconectado, apareciendo el logo de color negro. Al entrar en la pantalla del mapa, en la barra de inicio, también se observa que el logo del Bluetooth se muestra desconectado (ver *Figura 51*).

Figura 50. Principal – Bluetooth Desconectado.

Destacar que en la página del mapa se muestra la barra de inicio con la flecha para volver a la página principal, el título de la página y el logo indicando el tipo de

conexión Bluetooth a la derecha. En el centro de la pantalla aparece el mapa y en la parte inferior los datos del Arduino que podrán visualizarse posteriormente.

Figura 51. Mapa - Bluetooth Desconectado.

En el segundo caso, escenario más habitual, el logo del Bluetooth está conectado (azul). Al igual que en la página principal, en la página del mapa también aparece el logo del Bluetooth conectado (ver *Figura 52* y *Figura 53).*

Figura 52. Principal – Bluetooth Conectado.

Figura 53. Mapa – Bluetooth Conectado.

Una vez que el Bluetooth HC-05 está conectado y que el GPS recibe señal y encuentra satélites, se mostrarán los datos de los parámetros latitud, longitud y altitud a tiempo real en el apartado Datos Arduino, los cuales, a su vez, pueden visualizarse simultáneamente en el mapa (ver en la *Figura 54*).

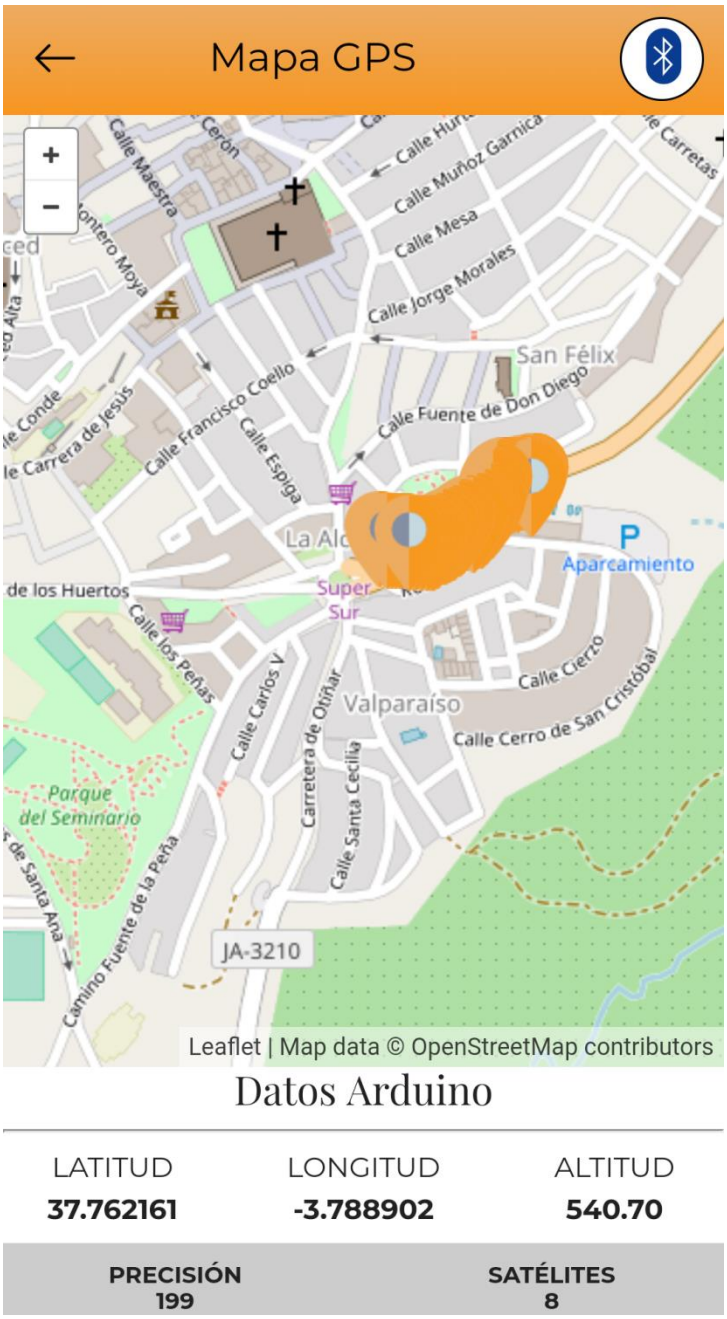


Figura 54. Mapa – Bluetooth Conectado – GPS.

En el caso de que el Bluetooth esté conectado, pero el GPS no sea capaz de localizar satélites, este mensaje se mostrará en el apartado Datos Arduino (ver *Figura 55).*

Figura 55. Mapa – Bluetooth Conectado – No hay satélites.

4. CONCLUSIONES

En el presente proyecto se ha llevado a cabo el diseño y fabricación de un dispostivo electrónico portable para bicicletas que ilumina mediantes leds las frenadas y los giros efectuados por el usuario.

Para corroborar el correcto funcionamiento del dispositivo electrónico, tanto de la iluminación y la señalización como de la aplicación móvil, se llevaron acabo ensayos reales en bicicleta (ver *Figura 56*).

Figura 56. Bicicleta.

Para ello, se diseñó un soporte constituido por dos ángulos de 15x450mm, una varilla roscada de M8 y una base de 80x160mm (ver *Figura 57*), que permitió sostener el teléfono móvil que realizaba la grabación del principio de funcionamiento del dispositivo.

Figura 57. Soporte.

Entre las principales dificultades encontradas destacan el establecimiento de la comunicación vía Bluetooth, así como la implementación de dos Arduinos para controlar tanto la iluminación y señalización del faro como la aplicación móvil. Al intentar programarlo todo en un mismo código (un único Arduino), aparecían errores en la obtención de datos del IMU ya que el MPU-6050 utiliza interrupciones de reloj para su correcto funcionamiento mientras que el GPS emplea la librería <*SoftwareSerial*>, la cual genera problemas con estas interrupciones, sin olvidar que el GPS necesita un segundo para encontrar señal. La velocidad de funcionamiento del IMU es mayor a un segundo, es decir, cada 'x' milisegundos lanza una nueva ejecución. Como esta secuencia forma parte del loop, al ejecutar el IMU, el flujo del programa se encuentra detenido a la espera del GPS durante ese segundo donde busca señal, de modo que el IMU devuelve valores de overflow (valores desbordados).

Sin embago, resulta muy motivador y gratificante poner la tecnología y, en concreto, la sensorización inteligente, al servicio de un medio de transporte alternativo y saludable, así como de un colectivo especialmente vulnerable. Fomentar el uso de la bicicleta y que los usuarios sientan una mayor seguridad, han constituido los pilares fundamentales para desarrollar este Trabajo Fin de Grado.

Destacar que este trabajo ha permitido adquirir nuevos conocimientos entre los que destacan la creación de una aplicación móvil con sistema Android utilizando lenguajes de programación como HTML, JavaScript y CSS, así como la utilización e implementación de distintos elementos tales como un sensor inercial, un Bluetooth y un dispositivo GPS. Además se ha ampliado conocimientos de programación en Arduino y en el manejo de softwares de diseño (CATIA, AutoCAD Electrical) y de otras herramientas como Fritzing.

5. REFERENCIAS

[1] *Dirección General de Tráfico.* (s.f.). Obtenido de http://www.dgt.es/es/

[2] *LogiLink.* (s.f.). Obtenido de http://www.logilink.eu/Products_LogiLink/Notebook-Computer_Accessories/USB_Hubs/USB_20_Hub_4-Port-Smile-blue_UA0136.htm

[3] *ElectroCrea.* (s.f.). Obtenido de https://electrocrea.com/products/arduino-nano

[4] *Arduino.* (s.f.). Obtenido de https://forum.arduino.cc/index.php?topic=147582.0

[5] *Circuits Today*. (s.f.). Obtenido de http://www.circuitstoday.com/arduino-nano-tutorial-pinout-schematics

[6] *NGS.* (s.f.). Obtenido de https://www.ngs.eu/es/tabletas-telefonos/baterias-externas/ngs-powerbank-powerpump-2200-grape/TABLETS_SMARTPHONES/TP-POWER-0017/

[7] *Luis Llamas*. (s.f.). Obtenido de https://www.luisllamas.es/medir-la-inclinacion-imu-arduino-filtro-complementario/

[8] *Luis Llamas*. (s.f.). Obtenido de https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/

[9] *Arduino*. (s.f.). Obtenido de https://playground.arduino.cc/Main/MPU-6050

[10] *PROBOTS*. (s.f.). Obtenido de https://probots.co.in/index.php?main_page=product_info&products_id=497

[11] *Blog*. (s.f.). Obtenido de https://verbuencine.wordpress.com/2010/05/07/%C2%BFcomo-funciona-el-sistema-gps/

[12] *Luis Llamas*. (s.f.). Obtenido de https://www.luisllamas.es/localizacion-gps-con-arduino-y-los-modulos-gps-neo-6/

[13] *Addicore*. (s.f.). Obtenido de https://www.addicore.com/NEO-6M-GPS-p/231.htm

[14] *u-blox*. (s.f.). Obtenido de https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf

[15] *Arquitectura de Computadoras*. (s.f.). Obtenido de https://conceptosarquitecturadecomputadoras.wordpress.com/bluethoot/

[16] *Arduino123*. (s.f.). Obtenido de http://arduino123.blogspot.com.es/p/bluetooth.html

[17] *PROMETEC*. (s.f.). Obtenido de https://www.prometec.net/bt-hc05/

[18] *Art of Circuits*. (s.f.). Obtenido de http://artofcircuits.com/product/hc-05-bluetooth-serial-pass-through-master-slave-module

[19] *Naylamp Mechatronics*. (s.f.). Obtenido de https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html

[20] *Apache Cordova*. (s.f.). Obtenido de https://cordova.apache.org/

[21] *Apache Cordova*. (s.f.). Obtenido de https://cordova.apache.org/docs/es/latest/guide/cli/

[22] *NetBeans*. (s.f.). Obtenido de https://netbeans.org/index_es.html

[23] *arsys*. (s.f.). Obtenido de https://www.arsys.es/blog/programacion/diseno-web/que-es-phonegap/

[24] *Apache Cordova*. (s.f.). Obtenido de https://cordova.apache.org/docs/en/8.x/config_ref/index.html

[25] *jQuery*. (s.f.). Obtenido de https://jquery.com/

[26] *Apache Cordova*. (s.f.). Obtenido de https://cordova.apache.org/docs/en/8.x/guide/platforms/android/plugin.html

[27] *GitHub*. (s.f.). Obtenido de https://github.com/mapbox/mapbox-plugins-android

[28] *GitHub*. (s.f.). Obtenido de https://github.com/don/BluetoothSerial

[29] *GitHub*. (s.f.). Obtenido de https://github.com/apache/cordova-plugin-splashscreen/blob/master/doc/es/index.md

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Electrónica y Automática

ANEXO 1: PLANOS

6. PLANOS

Planos del Proyecto

Planos mecánicos

Plano Nº1. Conjunto.

Plano Nº2. Despiece.

Plano Nº3. Tapa.

Plano Nº4. Carcasa.

Plano Nº5. Tapón.

Plano Nº6. Tuerca.

Plano Nº7. Tapón hueco.

Plano Nº8. Tuerca hueca.

Plano Nº9. Interruptor.

Planos eléctricos

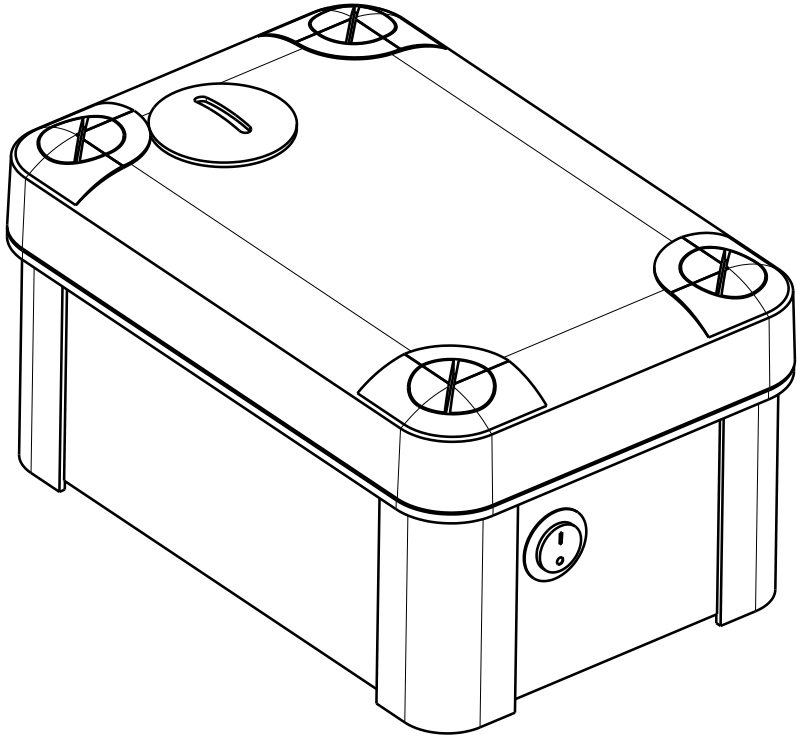
Plano UJA-2018-00.

Plano UJA-2018-00H01.

Plano UJA-2018-00H02.

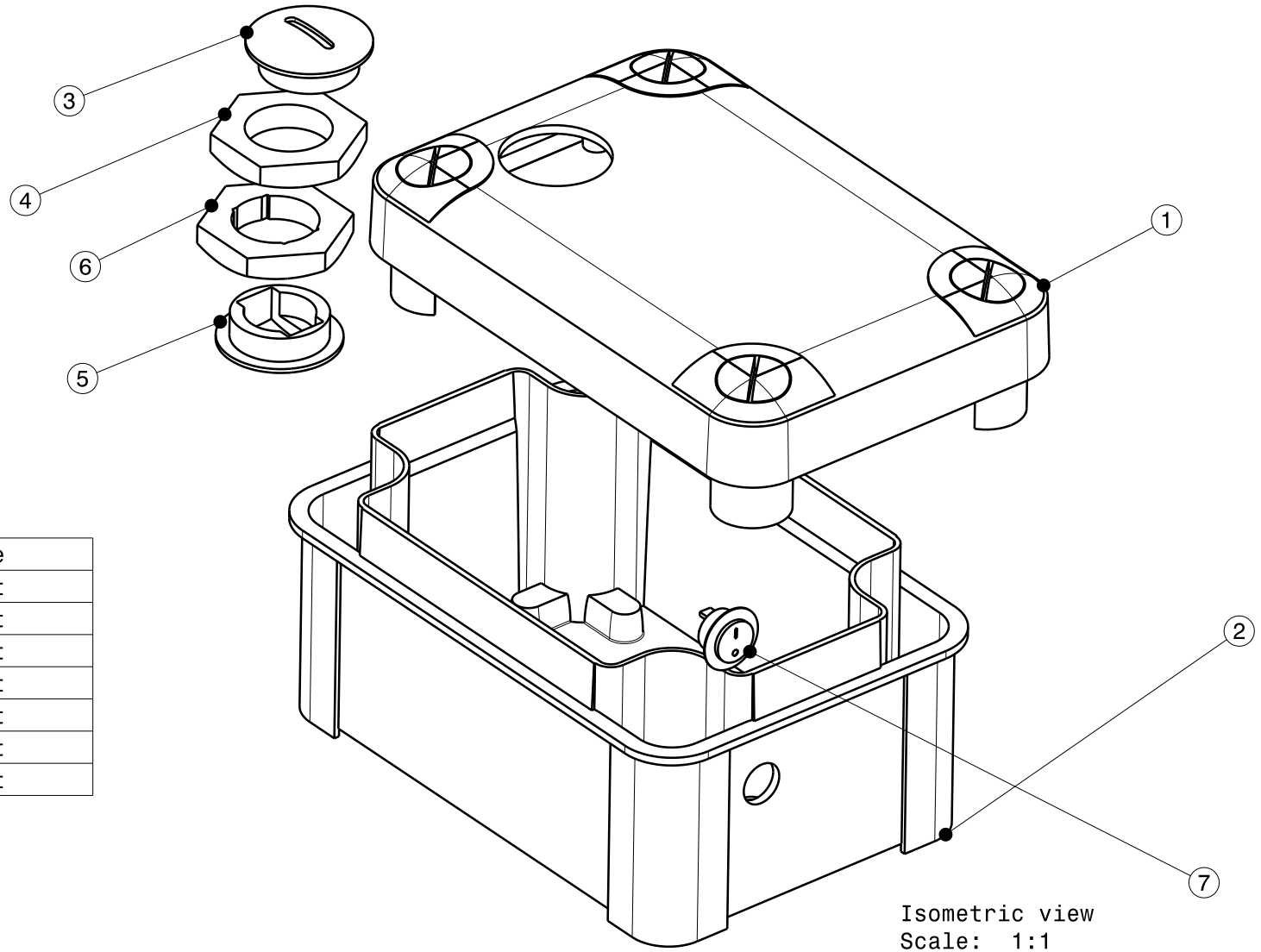
Plano UJA-2018-01H01.

Plano UJA-2018-02H01.



Isometric view
Scale: 1:1

DESIGNED BY: F.PLAZA DATE: 07/05/2018	UNIVERSIDAD DE JAÉN		
CHECKED BY: XXX DATE: XXX			
SIZE A3	FARO		
SCALE 1:1	WEIGHT (kg) XXX	DRAWING NUMBER Conjunto	SHEET 1/9



Isometric view
Scale: 1:1

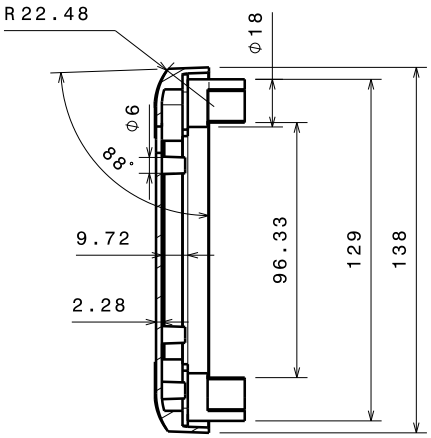
Bill of Material: Faro

Quantity	Part Number	Type
1	Tapa	Part
1	Carcasa	Part
1	Tapón	Part
1	Tuerca	Part
1	Tapon_hueco	Part
1	Tuerca_hueca	Part
1	Interruptor	Part

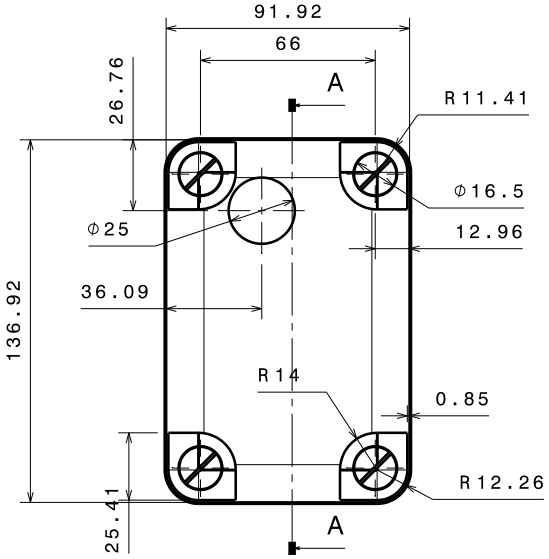
Recapitulation of: Faro
Different parts: 7
Total parts: 7

Quantity	Part Number
1	Tapa
1	Carcasa
1	Tapón
1	Tuerca
1	Tapon_hueco
1	Tuerca_hueca
1	Interruptor

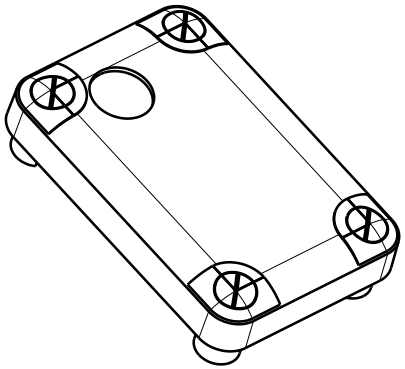
DESIGNED BY: F.PLAZA DATE: 07/05/2018	UNIVERSIDAD DE JAÉN		
CHECKED BY: XXX DATE: XXX			
SIZE A3	FARO		
SCALE 1:1	WEIGHT (kg) XXX	DRAWING NUMBER Despiece	SHEET 2/9



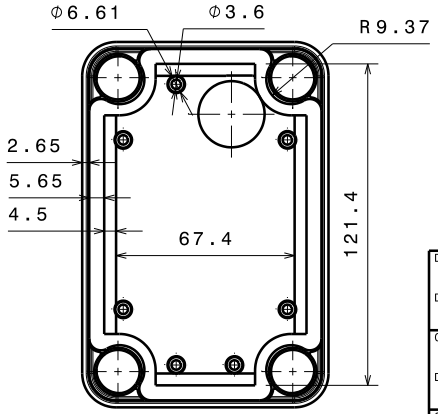
Section view A-A
Scale: 1:2



Front view
Scale: 1:2



Isometric view
Scale: 1:2

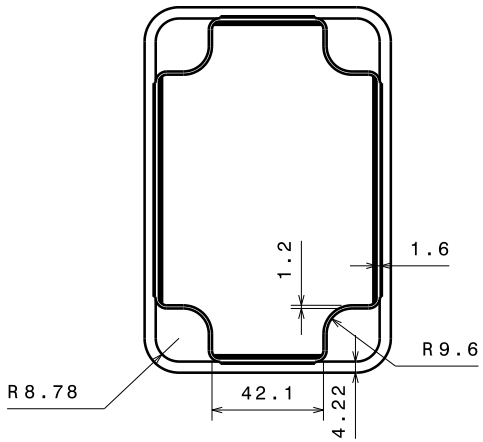


Bottom view
Scale: 1:2

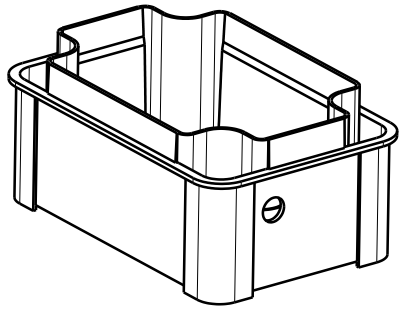
DESIGNED BY: F.PLAZA DATE: 07/05/2018	UNIVERSIDAD DE JAÉN		
CHECKED BY: XXX DATE: XXX			
SIZE A3	FARO		
SCALE 1:2	WEIGHT (kg) XXX	DRAWING NUMBER Tapa	SHEET 3/9

This drawing is our property; it can't be reproduced or communicated without our written agreement.

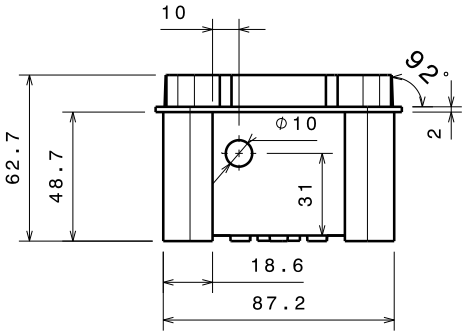
I	_
H	_
G	_
F	_
E	_
D	_
C	_
B	_
A	_



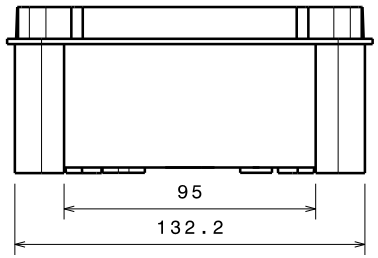
Top view
Scale: 1:2



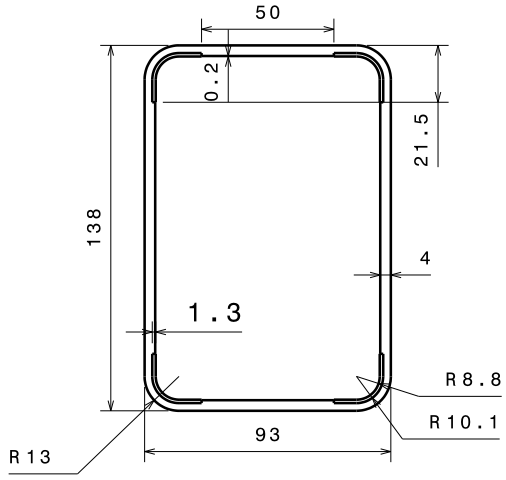
Isometric view
Scale: 1:2



Front view
Scale: 1:2



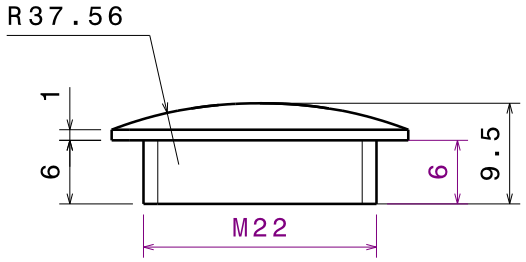
Left view
Scale: 1:2



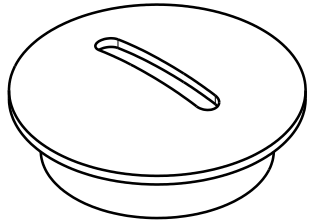
Bottom view
Scale: 1:2

DESIGNED BY: F.PLAZA DATE: 07/05/2018	UNIVERSIDAD DE JAÉN		
CHECKED BY: XXX DATE: XXX			
SIZE A3	FARO		
SCALE 1:2	WEIGHT (kg) XXX	DRAWING NUMBER Carcasa	SHEET 4/9

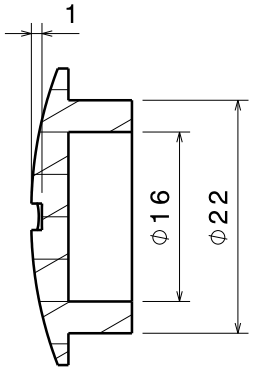
This drawing is our property; it can't be reproduced or communicated without our written agreement.



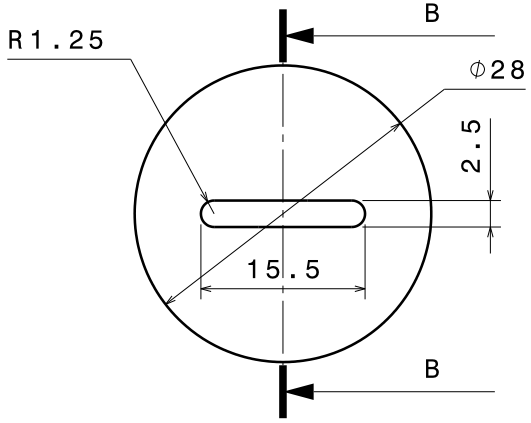
Front view
Scale: 2:1



Isometric view
Scale: 2:1

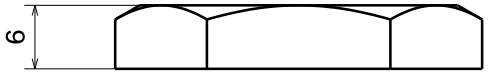


Section view B-B
Scale: 2:1

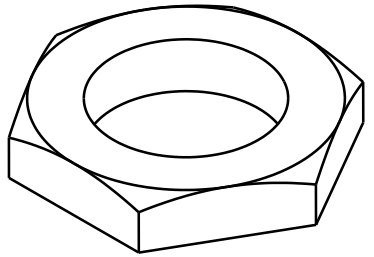


Top view
Scale: 2:1

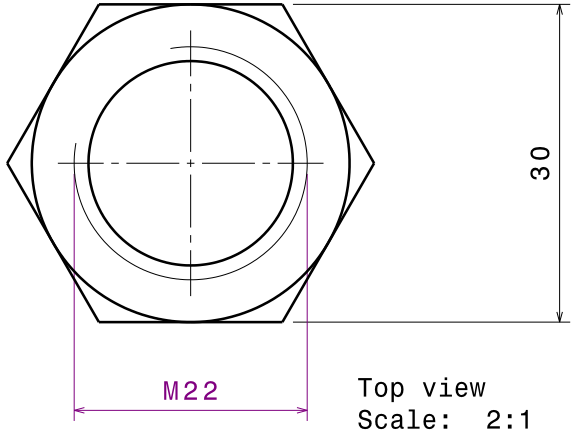
DESIGNED BY: F.PLAZA DATE: 07/05/2018	UNIVERSIDAD DE JAÉN		
CHECKED BY: XXX DATE: XXX			
SIZE A3	FARO		
SCALE 2:1	WEIGHT (kg) XXX	DRAWING NUMBER Tapón	SHEET 5/9



Front view
Scale: 2:1

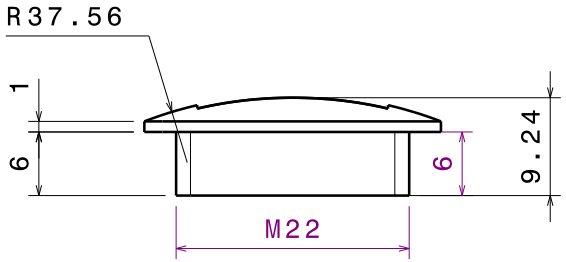


Isometric view
Scale: 2:1

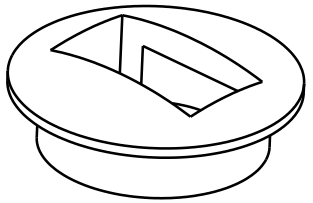


Top view
Scale: 2:1

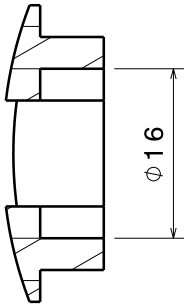
DESIGNED BY: F.PLAZA DATE: 07/05/2018	UNIVERSIDAD DE JAÉN		
CHECKED BY: XXX DATE: XXX			
SIZE A3	FARO		
SCALE 2:1	WEIGHT (kg) XXX	DRAWING NUMBER Tuerca	SHEET 6/9



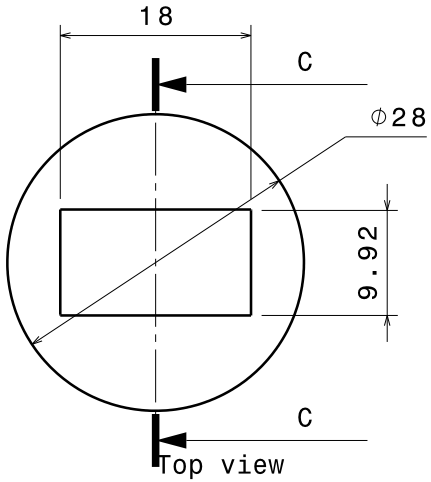
Front view
Scale: 2:1



Isometric view
Scale: 2:1

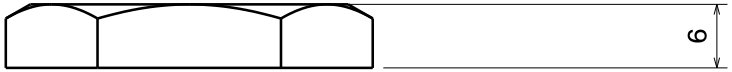


Section view C-C
Scale: 2:1

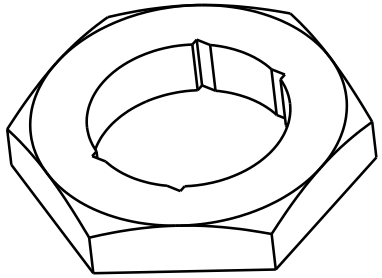


Top view
Scale: 2:1

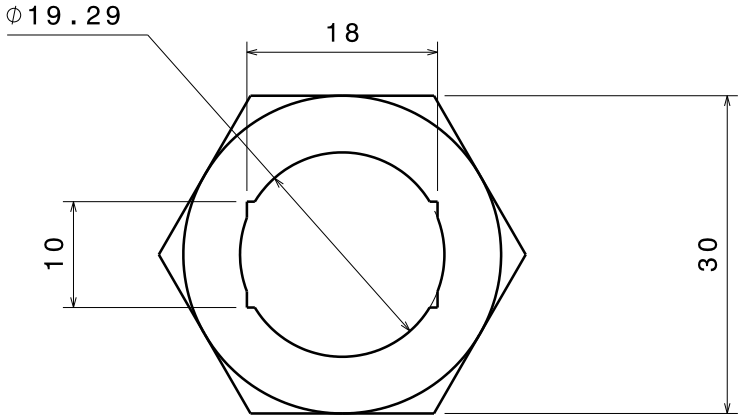
DESIGNED BY: F.PLAZA DATE: 07/05/2018	UNIVERSIDAD DE JAÉN		
CHECKED BY: XXX DATE: XXX			
SIZE A3	FARO		
SCALE 2:1	WEIGHT (kg) XXX	DRAWING NUMBER Tapón hueco	SHEET 7/9



Front view
Scale: 2:1

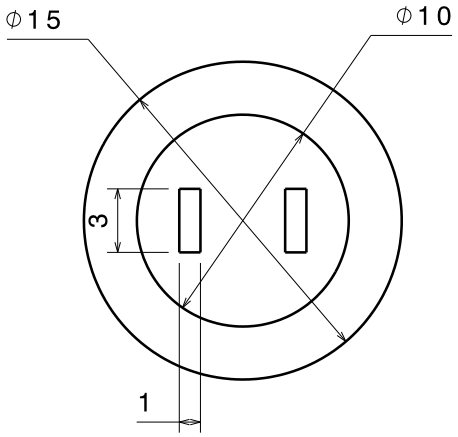


Isometric view
Scale: 2:1

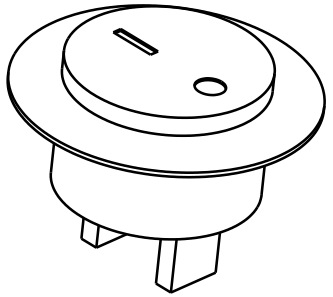


Top view
Scale: 2:1

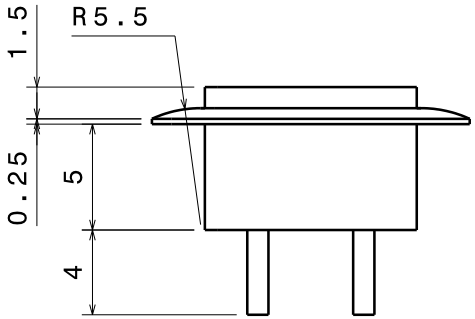
DESIGNED BY: F.PLAZA DATE: 07/05/2018	UNIVERSIDAD DE JAÉN		I _
CHECKED BY: XXX DATE: XXX			H _
			G _
SIZE A3	FARO		F _
			E _
			D _
SCALE 2:1	WEIGHT (kg) XXX	DRAWING NUMBER Tuerca hueca	SHEET 8/9 C _ B _ A _



Bottom view
Scale: 4:1



Isometric view
Scale: 4:1



Front view
Scale: 4:1

DESIGNED BY: F.PLAZA DATE: 07/05/2018	UNIVERSIDAD DE JAÉN		
CHECKED BY: XXX DATE: XXX			
SIZE A3	FARO		
SCALE 4:1	WEIGHT (kg) XXX	DRAWING NUMBER Interruptor	SHEET 9/9

TYPE :FARO LUZ/INTERMITENTES
MODEL :ELABORACIÓN PROPIA
P/N :UJA2018-00

USB-A
EXT-P1
5VDC
DATA -
DATA +
GND

IN259

MICRO USB INPUT
BT01-P2
5VDC
DATA -
DATA +
GND
NC

REF :BT01
TYPE :POWERBANK 2200
MODEL :NGS
P/N :POWERPUMP2200GRAPE

USB-A OUTPUT
BT01-P1
5VDC
DATA -
DATA +
GND

S1

COMMERCIAL WIRE INCLUDED IN HUB USB

REF :HUB01
TYPE :HUB 4 PUERTOS USB 2.0
MODEL :LogiLink
P/N :UA0136

5V
5VDC
DATA-
DATA+
GND

USB-A
HUB01-P1
5VDC
DATA -
DATA +
GND

USB-A
HUB01-P2
5VDC
DATA -
DATA +
GND

REF :ARD-IS
TYPE :ARDUINO PARA ILUMINACIÓN/SEÑALIZACIÓN DEL FARO
MODEL :ELABORACIÓN PROPIA
P/N :UJA-2018-01

REF :ARD01
TYPE :ARDUINO 1
MODEL :ARDUINO
P/N :A000005

MINI USB
ARD01-P1
5VDC
DATA -
DATA +
GND
NC

COMMERCIAL WIRE INCLUDED IN ARDUINO NANO

D13 D12
3V3 D11
REF D10
A0 D9
A1 D8
A2 D7
A3 D6
A4 D5
A5 D4
A6 D3
A7 D2
+5V GND
RST RST
GND RXD
VIN TX1

REF :IMU01
TYPE :MPU6050
MODEL :SODIAL
P/N :014802

VCC
GND
SCL
SDA
XDA
XCL
ADO
INT

R1 220
L1_IZQ
YE

R2 220
L2_DER
YE

R3 220
L3_FRENO
RD

COMMERCIAL WIRE INCLUDED IN ARDUINO NANO

REF :ARD-APP
TYPE :ARDUINO PARA APP MÓVIL
MODEL :ELABORACIÓN PROPIA
P/N :UJA-2018-02

REF :ARD02
TYPE :ARDUINO 2
MODEL :ARDUINO
P/N :A000005

MINI USB
ARD02-P1
5VDC
DATA -
DATA +
GND
NC

REF :GPS01
TYPE :GPS GY-NEO6MV2
MODEL :TOOGOO(R)
P/N :070350

VCC
RX
TX
GND

REF :BT01
TYPE :BLUETOOTH HC-05
MODEL :SODIAL
P/N :037708

STATE
RXD
TXD
GND
VCC
EN

REF :ARD–IS
TYPE :ARDUINO PARA ILUMINACIÓN/SEÑALIZACIÓN DEL FARO
MODEL :ELABORACIÓN PROPIA
P/N :UJA–2018–01

REF :ARD01
TYPE :ARDUINO 1
MODEL :ARDUINO
P/N :A000005

MINI USB
ARD01-P1

1	5VDC
2	DATA -
3	DATA +
4	GND
5	NC

COMMERCIAL WIRE INCLUDED IN ARDUINO NANO

USB-A
HUB01-P1

5VDC	1
DATA -	2
DATA +	3
GND	4

REF :HUB01
TYPE :HUB 4 PUERTOS USB 2.0
MODEL :LogiLink
P/N :UA0136

REF :ARD–APP
TYPE :ARDUINO PARA APP MÓVIL
MODEL :ELABORACIÓN PROPIA
P/N :UJA–2018–02

REF :ARD02
TYPE :ARDUINO 2
MODEL :ARDUINO
P/N :A000005

MINI USB
ARD02-P1

1	5VDC
2	DATA -
3	DATA +
4	GND
5	NC

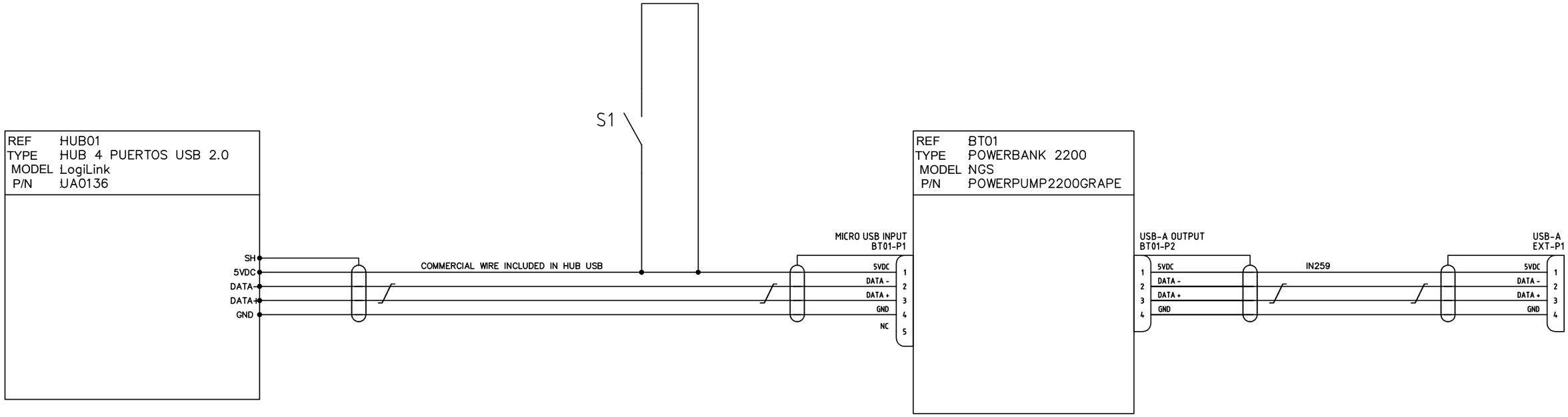
COMMERCIAL WIRE INCLUDED IN ARDUINO NANO

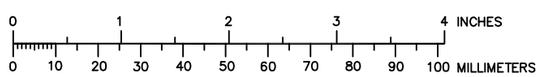
USB-A
HUB01-P2

5VDC	1
DATA -	2
DATA +	3
GND	4

0 1 2 3 4 INCHES
0 10 20 30 40 50 60 70 80 90 100 MILLIMETERS

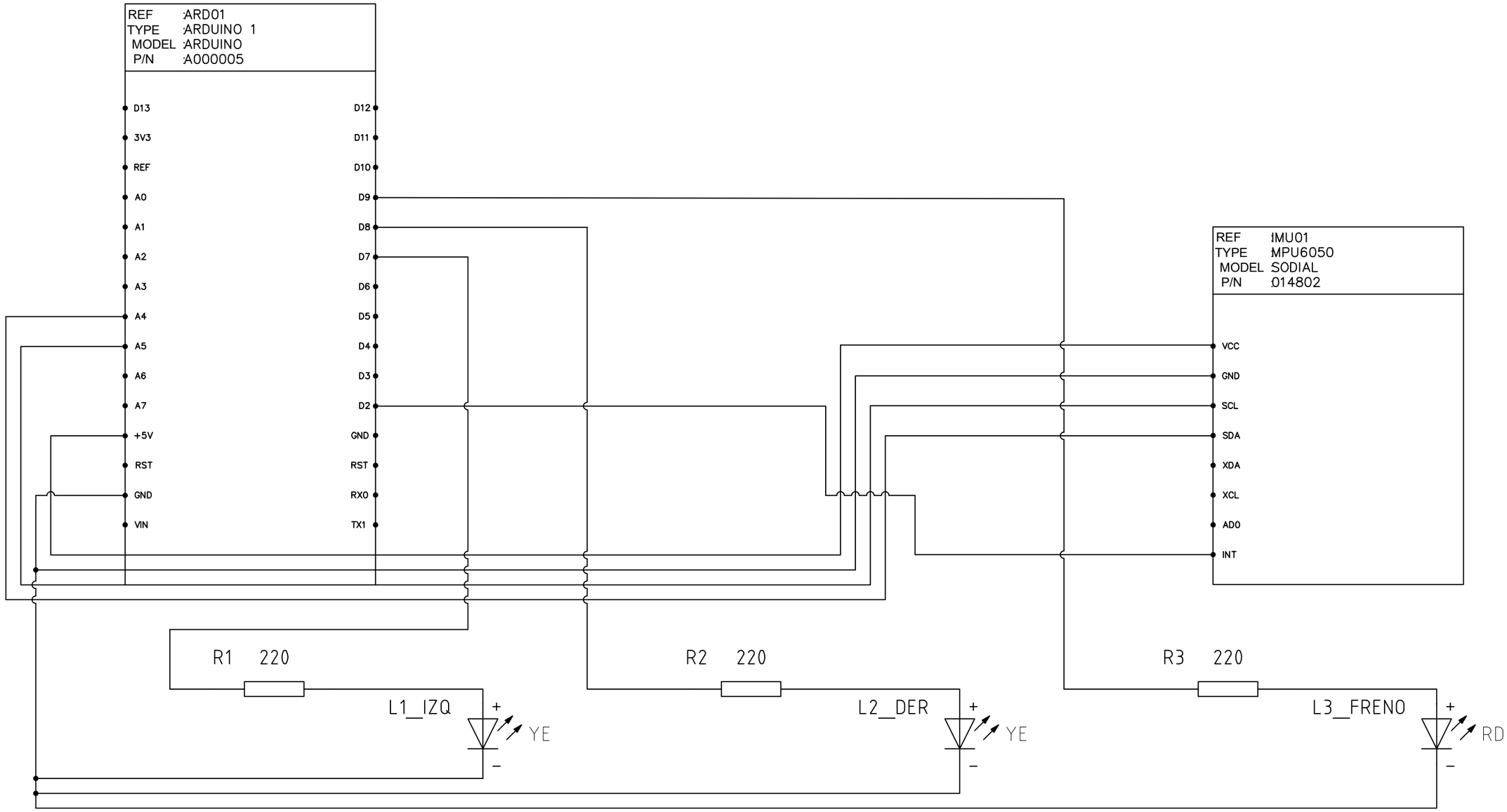
UNIVERSIDAD DE JAÉN	Dibujado: FERNANDO PLAZA QUEVEDO	A3	CAD: AutoCad Electrical
	Proyecto: SENSORIZACIÓN INTELIGENTE Y APP MÓVIL PARA BICICLETAS		
Descripción: DISEÑO FARO INTERFAZ ELÉCTRICA	N° Plano: UJA–2018–00H01	Fecha: 03–05–2018	Hoja: 01/02





UNIVERSIDAD DE JAÉN	Dibujado: FERNANDO PLAZA QUEVEDO	A3	CAD: AutoCad Electrical
	Proyecto: SENSORIZACIÓN INTELIGENTE Y APP MÓVIL PARA BICICLETAS		
Descripción: DISEÑO FARO INTERFAZ ELÉCTRICA		N° Plano: UJA-2018-00H02	Fecha: 03-05-2018
			Hoja: 02/02

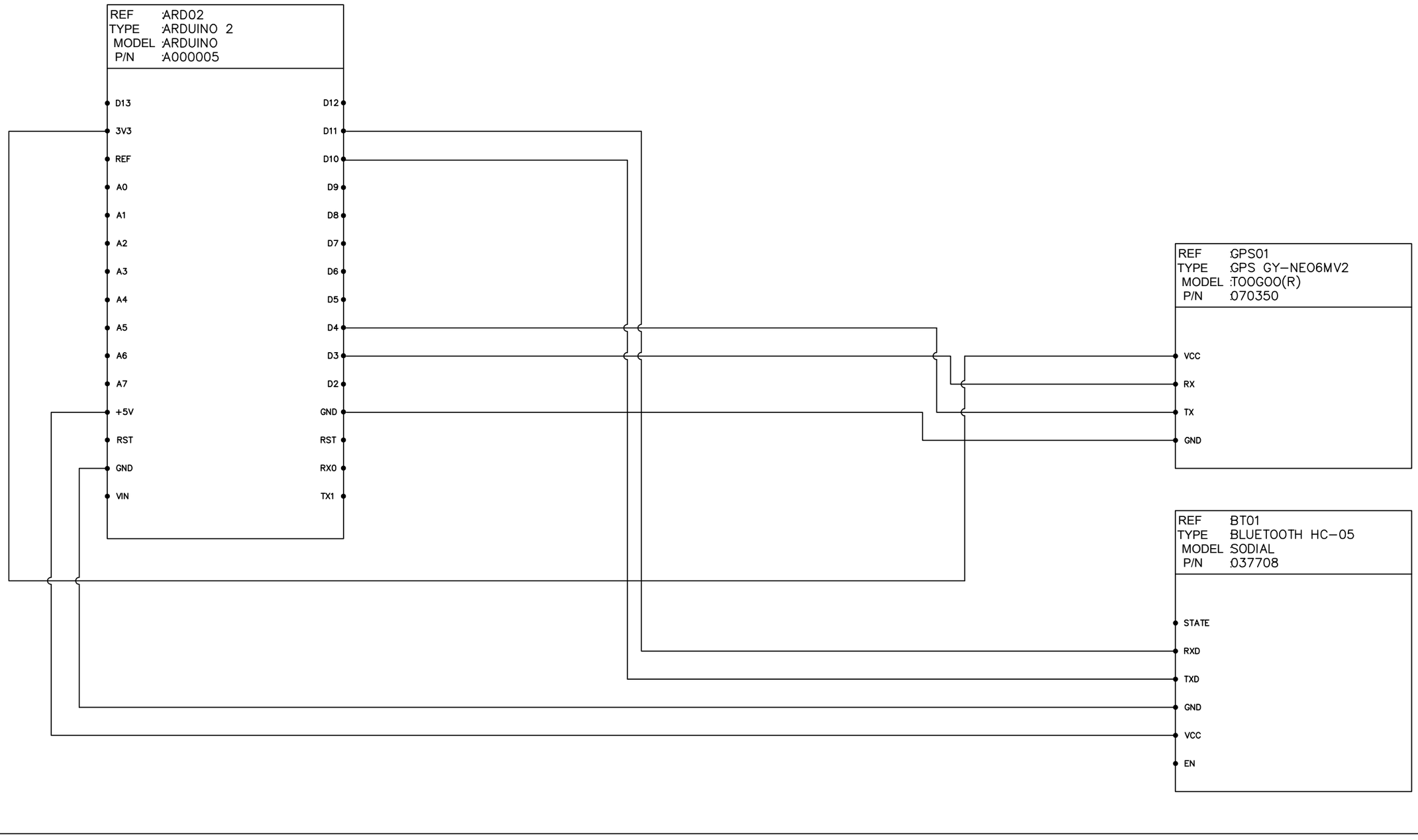
REF :ARD–IS
TYPE :ARDUINO PARA ILUMINACIÓN/SEÑALIZACIÓN DEL FARO
MODEL :ELABORACIÓN PROPIA
P/N :UJA–2018–01



0 1 2 3 4 INCHES
0 10 20 30 40 50 60 70 80 90 100 MILLIMETERS

UNIVERSIDAD DE JAÉN	Dibujado: FERNANDO PLAZA QUEVEDO	A3	CAD: AutoCad Electrical
	Proyecto: SENSORIZACIÓN INTELIGENTE Y APP MÓVIL PARA BICICLETAS		
Descripción: DISEÑO FARO INTERFAZ ELÉCTRICA	Nº Plano: UJA–2018–01H01		Fecha: 03–05–2018
			Hoja: 01/01

REF :ARD-APP
TYPE :ARDUINO PARA APP MÓVIL
MODEL :ELABORACIÓN PROPIA
P/N :UJA-2018-02



0 1 2 3 4 INCHES
0 10 20 30 40 50 60 70 80 90 100 MILLIMETERS

UNIVERSIDAD DE JAÉN	Dibujado: FERNANDO PLAZA QUEVEDO	A3	CAD: AutoCad Electrical
	Proyecto: SENSORIZACIÓN INTELIGENTE Y APP MÓVIL PARA BICICLETAS		
Descripción: DISEÑO FARO INTERFAZ ELÉCTRICA	N° Plano: UJA-2018-02H01		Fecha: 03-05-2018
			Hoja: 01/01