UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

SISTEMAS DE SENSORES INTELIGENTES PARA DETECCIÓN DE AUTOMÓVILES

Daniel Gómez Fernández

Abril 2014

SISTEMAS DE SENSORES INTELIGENTES PARA DETECCIÓN DE AUTOMÓVILES

AUTOR: Daniel Gómez Fernández
TUTOR: Eduardo Boemo Scalvinoni

Digital System Lab
Dpto. Tecnología Electrónica y de comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Abril 2014

Resumen

Existen ciertos grupos de colectivos sociales que se ven excluidos de la posibilidad de resolver incomodidades relativas a situaciones cotidianas por medio de los avances tecnológicos, por el simple hecho de suponer una minoría consumista.

Es este motivo precisamente, el que propicia que los productos de esta índole presentes en el mercado tengan asociados unos costes fuera del alcance de los bolsillos de un portentaje muy alto de estos usuarios y desemboque, por consiguiente ,en la desaparición del interés en dicho producto tanto en su oferta como en su demanda.

El objetivo principal de este proyecto es el de desarrollar un prototipo de producto comercial empleando tecnologías punteras de bajo coste, específicamente para un sector minoritario como es el de las personas con discapacidad visual, y explotar sus características para hacer de una tarea cotidiana que suponga ciertas dificultades extraordinarias, más fácil y realizable.

En concreto, el proyecto se ha materializado en un sistema de reconocimiento inteligente de paso de vehículos en pasos de peatones sin señalización con semáforos, que mediante la infraestructura oportuna y gracias a comunicación Bluetooth 4.0 con el Smartphone del usuario, permita no sólo advertir de un posible peligro por paso de vehículos, si no de una serie de condiciones inherentes del entorno en el momento del cruce que permitan al usuario realizar el cruce con total seguridad.

Para ello se desarrollará y fabricará tanto el hardware de bajo coste de la infraestructura y el firmware asociado a la misma, como el software de la aplicación de usuario.

Palabras clave:

- Discapacidad visual
- Teléfonos inteligentes
- Bluetooth 4.0
- BLE
- Detección de vehículos
- iOS
- Arduino

Abstract

Nowadays there are certain social groups that are excluded of the possibility of solve usual activities through technological advances, just by the fact of belong to a consumer minority.

This is the main reason to increase the prices of market of this kind of products, and it means not to be economically accessible to a huge percentage of these users, and consecuently the commercial interest on them disappear.

The main target of this Project, is to develop and manufacture a prototype of commercial product, based on edge-low cost technologies, specifically for a minority group as blind people, and take advantage of its characteristics to make their daily tasks more accessible and attainable.

In this case, the project purposeb is to materialize an intelligent crosswalk assitant system to detect vehicles on pedestrian crossings without traffic lights, to allow, with the appropriate infrastructure and Bluetooth 4.0 wireles communications, not only to warn the user because of a possible car crossing, but also for some environmental conditions that make the user's crossing absolutely safe.

To carry out with it, it will be developed and manufactured both low cost infrastructure hardware and firmware involved with it, and user application software.

Keywords:

- Blind people
- Smartphones
- Bluetooth 4.0
- BLE
- Crosswalk assitant system
- iOS
- Arduino

Agradecimientos

Esta etapa de mi vida ha sido única, imprescindible y apasionante. Con momentos en la cumbre y momentos en valles profundos.

Aún así, todos ellos han merecido la pena, haciendo posible que en el momento en el que escribo estas líneas todos los valles parecezcan montañas muy muy altas.

Gracias a mis padres y mis hermanos, por aguantar todos mis defectos y manías; por su apoyo y comprensión (en ocasiones incomprensible) y por su amor incondicional.

Gracias a mis amigos, por hacer que los momentos difíciles fuesen más llevaderos y los momentos de alegría fueran festejados por todo lo alto y compartidos de la manera más sincera que conozco.

Gracias en especial a Tomás, Diego e Ignacio, por haberme dado la oportunidad de formar de un grupo inolvidable, que ha trabajado codo con codo en situaciones a veces inverosímiles para sacar esto adelante.

Gracias a Cris, por ser quien es, y por su apoyo, su comprensión, su cariño. Sin ti, esto no habría sido igual.

Gracias a mis profesores por todo lo que he aprendido, hoy soy quien soy en gran parte gracias a todos y cada uno de ellos.

En especial gracias a mi tutor Eduardo, por haberme dado la oportunidad de trabajar con él y haberme abierto las puertas de este proyecto.

Gracias.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 Motivación

La electrónica de consumo presenta una de las crecidas más importantes en las últimas décadas en la economía de los países desarrollados. Sin ir más lejos, 2013 vuelve a crecer un 4% respecto a 2012, lo que supone más de 1,1 billón de dólares a nivel mundial, según la CEA (Consumer Electronic Association).[1]

Éstas cifras demuestran la importante inversión que acometen numerosas empresas con el fin de presentar el producto tecnológicamente más avanzado del mercado, en el que sólo smartphones y tablets suponen un 40% del consumo total.

Por tanto, las tecnologías y más en concreto las relacionadas con la electrónica de consumo, forman parte constitutiva de la sociedad actual.

Según el artículo "Tecnología y discapacidad visual: necesidades tecnológicas y aplicaciones en la vida diaria de las personas con ceguera y deficiencia visual"[2] elaborado por la ONCE en los últimos años, esta premisa supone que las personas que no tienen una relación normalizada con estas tecnologías corren el riesgo de quedar excluidas o marginadas de la sociedad e incluso de la relación que la persona tenga con las mismas dependerá la posición final que ocupe en ella.

Éste artículo apunta además, que pese a que tradicionalmente las tecnologías han sido concebidas, proyectadas, producidas y aplicadas al patrón de persona media, sin tener en cuenta las diferencias con respecto a personas con discapacidades como la invidencia, cada vez este hecho está cambiando más en el apartado sobre todo de aplicación.

Esto quiere decir que los productos tecnológicos, por norma general, se siguen concibiendo, proyectando y produciendo para el patrón de persona media, pero sin embargo, las características asociadas permiten una adaptabilidad y una capacidad de desarrollo que contribuye a una mejor accesibilidad universal y usabilidad general.

Ésta mejora se antoja excesivamente simple en más del 80% de los casos de aplicaciones específicas para este reducido sector.

En varias entrevistas a invidentes[3] [4], éstos como usuario final, califican de realmente útil los esfuerzos de las compañías de terminales móviles en cuestión de accesibilidad en sus productos, sin embargo, esto no es así en las aplicaciones específicas mencionadas.

[1] http://www.cea.org

[2] sid.usal.es/libros/discapacidad/7033/8-1/tecnologia-y-discapacidad-visual-necesidades-tecnologicas-y-aplicaciones-en-la-vida-diaria-de-las-personas-con-ceguera-y-deficiencia-visual.aspx

[3] http://barrapunto.com/article.pl?sid=10/09/20/0753242

[4] http://www.20minutos.es/noticia/1379061/0/invidente/usa/iphone/

Éstas aplicaciones, en la gran mayoría de ocasiones, resultan de poca utilidad y pasan a formar parte del listado de apps de entretenimiento del usuario y no presentan un soporte real para una actividad rutinaria.

Las bases de un producto potencialmente válido para una mayor accesibilidad de las personas con discapacidad están sentadas, simplemente hay que dar con la tecla de una aplicación que transforme una tarea tediosa en una simple, con el aprendizaje correspondiente.

En este proyecto hemos querido dar una vuelta de tuerca a la infinidad de "aplicaciones lazarillo" para el guiado de usuarios invidentes, que no explotan las capacidades de los terminales inteligentes adecuadamente, y por ello no suponen una solución efectiva.

El entorno físico, además del geográfico, es un factor imprescindible a tener en cuenta en este tipo de aplicaciones.

Sin lugar a dudas, la visión global de este proyecto es muy ambiciosa, y por ello, hemos querido poner la primera piedra centrándonos en una sub-tarea de las más complicadas de una aplicación de este alcance.

Uno de los retos más complejos del guiado de invidentes es el cruce de pasos de peatones no regulados con semáforos, sin ningún tipo de accesibilidad con señalización acústica, condición que presentan casi un 100% de estos cruces. Por ello, nuestra aplicación utilizará la tecnología externa más adecuada para, de manera eficiente poner en conocimiento del usuario, a tiempo real, el estado del cruce en cuestión, notificando si se aproximan vehículos y la velocidad de los mismos, si el asfalto puede estar resbaladizo e indicando donde comienza y acaba el cruce.

1.2 Objetivos

Consecuentemente con el punto anterior, en este proyecto se pretende diseñar, desarrollar y fabricar un sistema de bajo coste, para continuar con el compromiso de acercar la tecnología al mayor número de usuarios posible, basado en equipos electrónicos y smartphones como unidad central de proceso y dotado de sensores inteligentes para ofrecer al usuario la información necesaria para realizar un cruce sin semáforos con total seguridad.

Además el proyecto es la primera piedra de una serie de aplicaciones desarrolladas para smartphones, cuyos resultados aportarán un primer acercamiento a las tecnologías empleadas y del cual se sacarán las conclusiones oportunas gracias a los resultados de pruebas de viabilidad.

Por estos motivos, el producto final constituirá un prototipo "Open Source" y "Open Hardware" sobre el que continuarán los desarrollos citados.

Cabe destacar, además, que este proyecto cuenta con financiación de la fundación Vodafone España

1.3 Metodología y plan de trabajo

Se ha diseñado un plan de trabajo basado en los siguientes hitos:

1. **Estado del arte:** Realización de un estudio del estado del arte referido a equipos relacionados con la idea conceptual planteada.
2. **Diseño de requisitos de aplicación:** Diseño de requisitos de la aplicación y los equipos basados en el punto anterior.
3. **Diseño preliminar:** Realización de un diseño preliminar del Hardware y la mecánica.
4. **Diseño final basado en componentes comerciales:** Tecnologías y módulos implementados, en base a la oferta tecnológica actual.
5. **Estimación de costes y planificación.**
6. **Desarrollo:** La fase de desarrollo se dividirá en 3 sub-tareas predeterminadas:
    a. Desarrollo Hardware: Se integrará en una sola electrónica, todos los componentes necesarios para la adaptación de alimentación y señales relativas a los módulos y sensores empleados.
    b. Desarrollo Firmware: Se desarrollará el firmware del PIC seleccionado, empleando los algoritmos y librerías necesarios para el correcto funcionamiento del hardware.
    c. Desarrollo software: Se desarrollará la aplicación específica para el control del hardware y la comunicación con el mismo (envío y recepción de mensajes) que posibiliten la interacción ususario - hardware de reconocimiento, mediante un Smartphone.
7. **Integración pruebas y resultados:** Una vez especificados los requisitos y desarrollado el prototipo en cuestión de software, firmware y hardware, se realizarán unas pruebas de campo para validar la funcionalidad de la aplicación y verificar que se cumplen dichos requisitos.
8. **Conclusiones técnicas y trabajo futuro.**

1.4 Organización de la memoria

La memoria se ha dividido en tantos puntos como muestra el punto metodología y plan de trabajo, enumerados de la misma manera.

Dicha enumeración está definida respetando el orden cronológico de cumplimiento de tareas.

2 ESTADO DEL ARTE

2.1 Aplicaciones accesibles

Los dispositivos móviles pueden presumir de ser cada vez más accesibles. Los principales fabricantes de dispositivos móviles centran parte de sus esfuerzos en conseguir terminales que lleguen a todos los públicos, independientemente de sus capacidades físicas. Por este motivo, daremos por supuesta esta condición de los terminales actuales y centraremos este estudio en el enfoque de funcionalidades y aplicaciones para usuarios con discapacidad visual.

Los Smartphones comienzan a ver en sus tiendas cada vez con más frecuencia aplicaciones basadas en reconocimiento de voz e información acústica con bases de datos lo suficientemente elaboradas como para que el terminal "capte" el significado de una conversación cotidiana. A continuación se muestran algunos ejemplos:

- **Siri:**

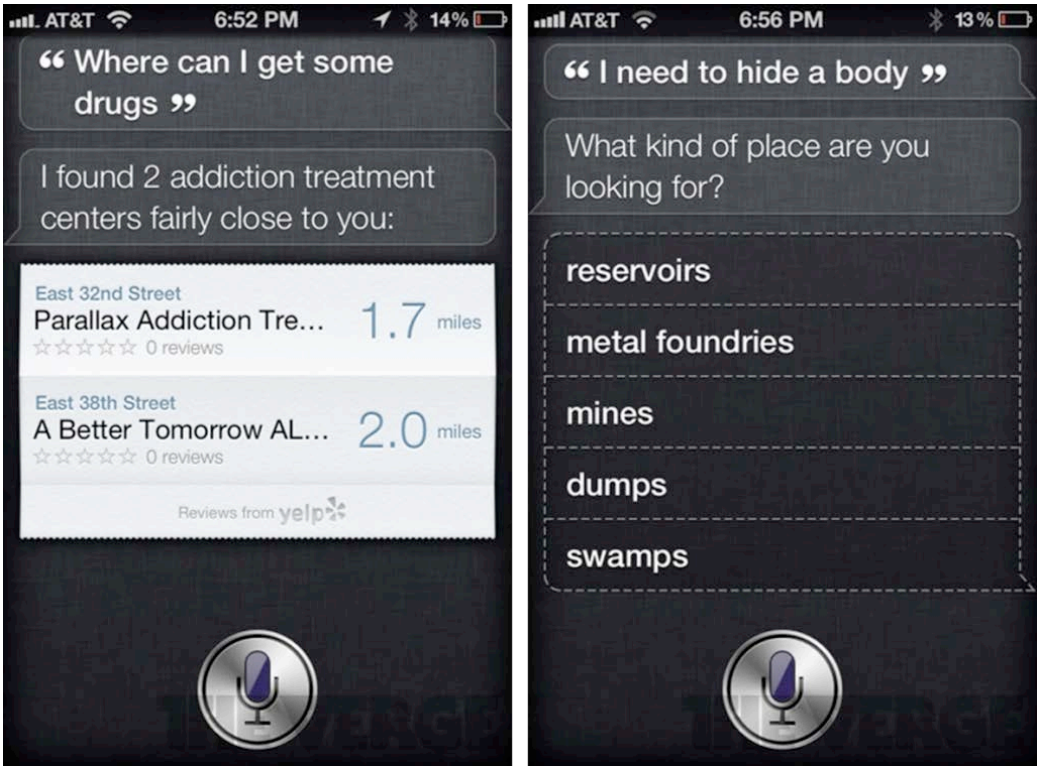


Ilustración 1. Siri para iOS

Esta funcionalidad de iOS permite, mediante la pulsación de un botón activar un asistente por comandos de voz, mediante el cual se pueden enviar mensajes, programar reuniones, hacer llamadas y otras muchas operaciones cotidianas. Es una funcionalidad pre-instalada en los terminales de Apple.

- **Sherpa:**

Ilustración 2. Sherpa para Android

Sherpa es el equivalente de Siri en los terminales de Android, sin embargo, Sherpa es una aplicación en lugar de una funcionalidad nativa como Siri. Por lo que requiere controlar el menú e iniciar la aplicación.

Estas aplicaciones o funcionalidades facilitan el control del dispositivo al usuario, incluyendo aquellos con discapacidad visual, sin embargo, el objeto de estudio más interesante en este escrito se ubica en las aplicaciones que tratan de mejorar la calidad de vida más allá del uso del propio dispositivo como teléfono y gestor de tareas.

En este sentido, los terminales hacen uso de los sensores integrados en el dispositivo para interaccionar con el mundo exterior, ya sea con la cámara, el sensor GPS, el acelerómetro, … etc. Y de estas características se pueden obtener grandes beneficios, para todos los perfiles de usuario, pero quizá con mayor repercusión en los usuarios con discapacidades visuales:

- **LookTel Money Reader:**[5]

Mediante esta aplicación y la cámara del dispositivo se escanea un billete y mediante visión artificial ésta reconoce su valor.

[5] http://www.looktel.com/moneyreader

Ilustración 3. Vista de LookTel Money Reader

- **Color identifier:** [6]

  Gracias a esta aplicación, haciendo uso también de la cámara del dispositivo, se puede conocer el color cualquier objeto con un toque en la pantalla táctil.

- **NAVATAR: Navegador indoor:**[7]

  Mediante esta aplicación se puede realizar una navegación paso a paso en el interior de un edificio. Para ello se utiliza un mapa 3D alojado en la aplicación y el acelerómetro del terminal, para ir calculando el avance en el trayecto.

2.2 Bluetooth y accesibilidad

La tecnología Bluetooth ha superado una barrera importante para meterse de lleno en el mundo de las WSN (Wireless Sensor Networks) o RSI(Redes de sensores inalámbricas), con el nuevo Bluetooth 4.0 [8](también llamado BLE o Bluetooth Smart). Básicamente el desarrollo de esta nueva tecnología ha permitido disminuir sustancialmente el consumo energético y los costes, manteniendo el alcance de la comunicación. Además otra ventaja reseñable, es la de no necesitar autenticación para establecer el emparejamiento de dispositivos.

[6] https://itunes.apple.com/us/app/color-identifier/id363346987?mt=8

[7] http://www.nbcnews.com/technology/smartphone-app-could-help-blind-navigate-indoors-780588

[8] http://www.pcworld.com/article/242517/bluetooth_4_0_becomes_smart_what_it_means_for_you.html

Apple, ha ideado un sistema basado en Bluetooth 4.0 llamado iBeacons[9] [10] mediante el cual se establecen balizas posicionales con módulos BLE, cuya intensidad de señal se utiliza para realizar una triangulación a modo de GPS y conocer la posición del teléfono.

Estos sistemas permiten desarrollar sistemas de posicionamiento indoor de bajo coste, con precisiones más que interesantes y sin necesidad de realizar instalaciones complicadas.
De nuevo, esta característica vuelve a ser un paso de gigante para los usuarios con discapacidad visual.

Existen distintas aplicaciones específicas desarrolladas para BLE y iBeacons en el AppStore, que presentan una magnífica gama de posibilidades para este perfil de usuario.

- **Indoo.rs:**

  Indoo.rs [11]es un sistema basado en iBeacons para smartphones y tabletas de Apple para la localización y guiado indoor. A través de su página web se puede comprar un Kit de Evaluación valorado en 520$ que incluye dos paquetes de software con aplicaciones distintas y un set de 10 balizas iBeacons de reducidas dimensiones, con baterías que duran hasta un año en operación continua, pesan menos de 30gr., y poseen un radio de acción de 40m.

- **Doors activation:**

  Desde la presentación de iOS7[12] dónde se expusieron todas las capacidades del sistema iBeacons, se planteó, como implantación del sistema, una aplicación basada en Módulos BLE, instalados en cierres de puertas, que permite abrirlas una vez detectada la señal del dispositivo del usuario.

9iBeacons: http://appleinsider.com/articles/13/06/19/inside-ios-7-ibeacons-enhance-apps-location-awareness-via-bluetooth-le

[10] http://www.businessinsider.com/beacons-and-ibeacons-create-a-new-market-2013-12

[11] http://indoo.rs/bluetoothkit/

[12] http://appleinsider.com/articles/13/06/19/inside-ios-7-ibeacons-enhance-apps-location-awareness-via-bluetooth-le

2.3 "Internet of things" aplicado a usuarios con discapacidad

El concepto de Internet of Things o (IoT) [13] apareció por primera vez en el Auto-ID Center del MIT a raíz de la tecnología RFID, NFC y códigos QR.

IoT se basa en la idea de identificar por micro-módulos emisores de ondas de radio, todos los objetos y/o personas que nos rodean en la vida cotidiana.

A través del concepto de IoT, codificando, registrando y rastreando los objetos y personas de nuestro alrededor se podría tener constancia de la ubicación y el estado de los mismos, y en definitiva, tener constancia de estos en cualquier lugar y en cualquier momento sin necesidad de percibirlos o comunicarnos con ellos.

Proyectos como IOTOPE o Tikitag buscan facilitar a los desarrolladores explotar sus ideas basadas en este concepto, poniendo sobre la mesa recursos como servidores de almacenamiento con bases de datos con capacidad para miles de millones de objetos, o etiquetas RFID programables.

Ilustración 4. Etiquetas RFID Tikitag para desarrollo de IoT

El proyecto Alcatel-Lucent[14] es un ejemplo de producto basado en este concepto. Posibilitando, por ejemplo, que un médico conozca el historial de su paciente en el mismo instante que su dispositivo detecta la etiqueta RFID que el usuario lleva en su DNI o cartera.

Nuevamente, la definitiva implantación de esta tecnología, abriría numerosas puertas al desarrollo de equipos, aplicaciones y dispositivos para usuarios invidentes.

[13] http://www.mckinsey.com/insights/high_tech_telecoms_internet/the_internet_of_things

[14] http://www.spainmovil.es/alcatel/noticias/n28440/Por-que-el-futuro-se-llama-4G-LTE.html

El colofón tecnológico en el ámbito de un público con discapacidad visual, se plantea la unión de todas las tecnologías citadas en este apartado.

De esta manera, se obtiene una circunstancia en la que se tiene constancia del entorno y sus propiedades, y el usuario se puede comunicar con él de forma automática y sin identificaciones, desde un dispositivo accesible.

Sin embargo, debido a que algunas de estas tecnologías se encuentran a día de hoy en desarrollo, éste concepto aún debe evolucionar para ser viable en un sentido práctico.

3 ESPECIFICACIÓN DE REQUISITOS DE LA APLICACIÓN

Una vez especificado el objetivo de este proyecto, y analizados los puntos planteados en el estado del arte, se profundizará en las características de la aplicación a desarrollar y se fijarán unos objetivos concretos relativos a cada punto técnico del proyecto.

Los requisitos de la especificación se dividirán en los siguientes apartados:

- Requisitos del hardware periférico
    - Electrónica
    - Comunicaciones
    - Sensores
- Requisitos del hardware de control de proceso
    - Dispositivo
    - Comunicaciones
- Requisitos del software de aplicación

A continuación se procede a detallar cada uno de los apartados:

3.1 Requisitos del hardware periférico

3.1.1 Electrónica

La tarjeta electrónica encargada de la recogida de las señales, tanto de los sensores, como de la comunicación inalámbrica, contará con al menos 3 entradas analógicas y 12 entradas/salidas digitales con al menos 1 puerto UART y, que permita la simulación de UARTs virtuales mediante E/S digitales.

El micro trabajará al menos a 16MHz y existirá la posibilidad de alimentación a través de la red eléctrica o baterías.

La tensión de trabajo del micro procesador será de 5V.

Aunque no será una característica crítica, se procurará que la electrónica sea lo más compacta posible.

El micro será compatible con el SDK de Arduino, y contará con un puerto ICSP para su programación.

3.1.2 Comunicaciones

El módulo de comunicaciones utilizará tecnología 2,4 GHz Bluetooth 4.0 (BLE). Dicho módulo debe ser lo más compacto posible.
Además presentará un consumo de menos de 25mA en cualquier circunstancia y del orden de micro amperios en condiciones de reposo. Por este motivo, el módulo deberá presentar un módo de hibernación o Sleep.

Además incorporará un sistema que permita actualizar el firmware en caso necesario.

Dicho módulo será Open Source y permitirá realizar desarrollos en su funcionalidad, así como en las características de las tramas enviadas, y todo ello presentará documentación accesible al público.

3.1.3 Sensores

El módulo electrónico periférico contará con sensores que permitan detectar el paso de vehículos en dos puntos distintos, permitiendo así, mediante la diferencia del tiempo transcurrido entre la activación de un sensor y el siguiente, los cuales se encuentran separados una distancia conocida, calcular la velocidad de paso de los mismos.

Estos sensores, serán sensores IR de gran alcance, permitiendo detectar vehículos a velocidades de hasta 90km/h a una distancia de hasta 1,5m.

El sensor llevará embebidos tanto el emisor como el recepto infrarrojo y presentará un consumo de menos de 40mA.

El retardo que presentará el sensor en la obtención de la medida será de menos de 10ms.
El error máximo en la medida de velocidad final mediante el algoritmo utilizado será de 5km/h.

Además, se utilizarán sensores de temperatura y humedad, que permitan detectar heladas. Estos sensores presentarán un consumo de menos de 50mA y a ser posible, ambas medidas (temperatura y humedad) estarán embebidas en un solo sensor digital.

3.2 Requisitos del hardware de control de proceso

El hardware encargado de recoger la información del hardware periférico y mostrarlo al usuario de la manera conveniente deberá incorporar tecnología 2,4GHz Bluetooth 4.0, compatible.

Deberá ser un dispositivo móvil o smartphone con sistema operativo compatible con la aplicación.

Deberá integrar un módulo GPS para la localización geográfica y módulo 3G de datos móviles para la descarga de mapas.

Además incorporará altavoces y micrófono integrados, así como un sistema que permita la accesibilidad a usuarios con discapacidad visual mediante comandos de voz y dictado de texto escrito.

3.3 Requisitos del software de aplicación

Se diseñará un protocolo para la comunicación Bluetooth 4.0 de tal manera que la aplicación reconozca el significado de las tramas enviadas por la periferia y viceversa.
Este protocolo estará una capa por encima del protocolo interno del módulo Bluetooth, de tal forma que éste será transparente para el desarrollador.

Se utilizará un campo de las tramas para cada operación definida en el proceso del cruce.
La aplicación deberá automatizar en la medida de lo posible todas las operaciones, evitando que el usuario necesite interactuar con el dispositivo móvil directamente, haciendo así, una aplicación útil en el sentido práctico.

La activación de la comunicación con la periferia se realizará también de forma automática mediante la localización de cada uno de los dispositivos electrónicos periféricos y el rastreo de la posición del usuario.

4 DISEÑO PRELIMINAR

4.1 Diseño conceptual

El concepto del sistema consiste, como ya se ha comentado, en un asistente de cruce de un paso de cebra sin semáforos para personas con discapacidad visual basado en una plataforma móvil o Smartphone y balizas inalámbricas.

Se utilizará una arquitectura tanto hardware como software que cumpla los requisitos especificados en el punto 3.

El diseño conceptual del sistema se muestra en la siguiente figura.

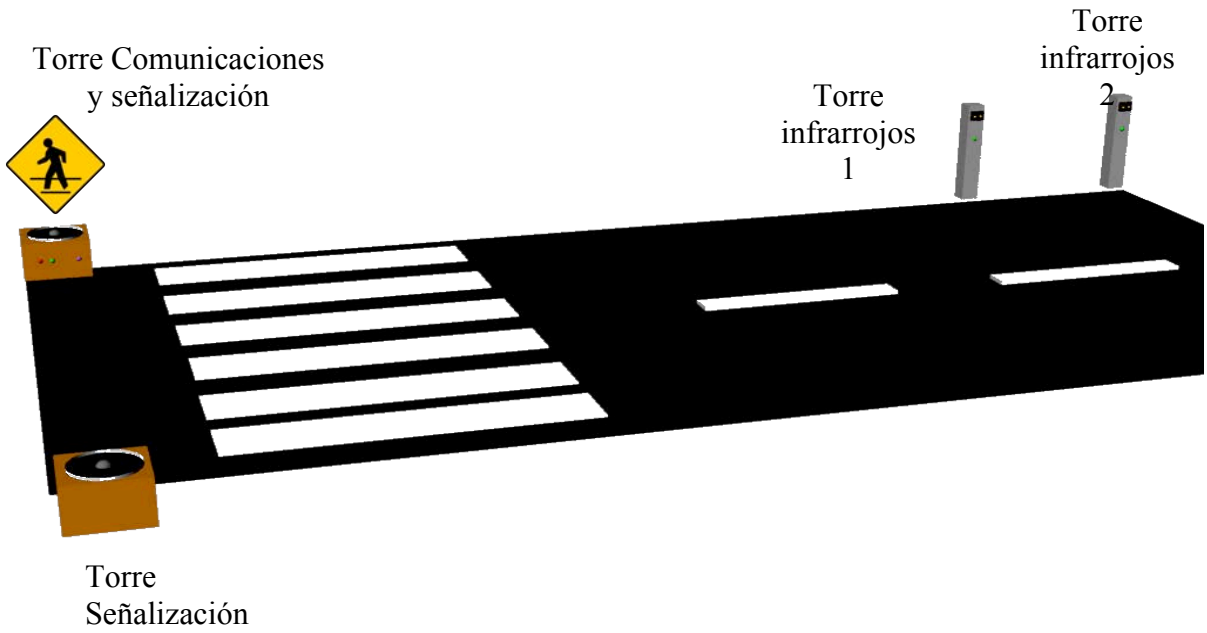


Ilustración 5. Esquema conceptual del sistema

4.1.1 Bloques hardware del sistema

En la ilustración se diferencian 4 bloques principales. Dos torres de señalización, de las cuales, una se corresponde con el encapsulado de la electrónica central periférica y torre de comunicaciones. Ésta cuenta con LEDs indicadores de estado y diagnóstico, y un altavoz que conforma una baliza acústica para marcar el inicio del paso de peatones.

En el otro lado de la acera, se encuentra un dispositivo físicamente de características similares, pero que conforma la baliza acústica que marca el final del paso de cebra, así como otro LED indicador de estado del sistema, empleado únicamente para diagnóstico.

Las dos torretas que se encuentran a la derecha de la imagen, albergan la electrónica periférica relativa a los sensores para la detección de automóviles.

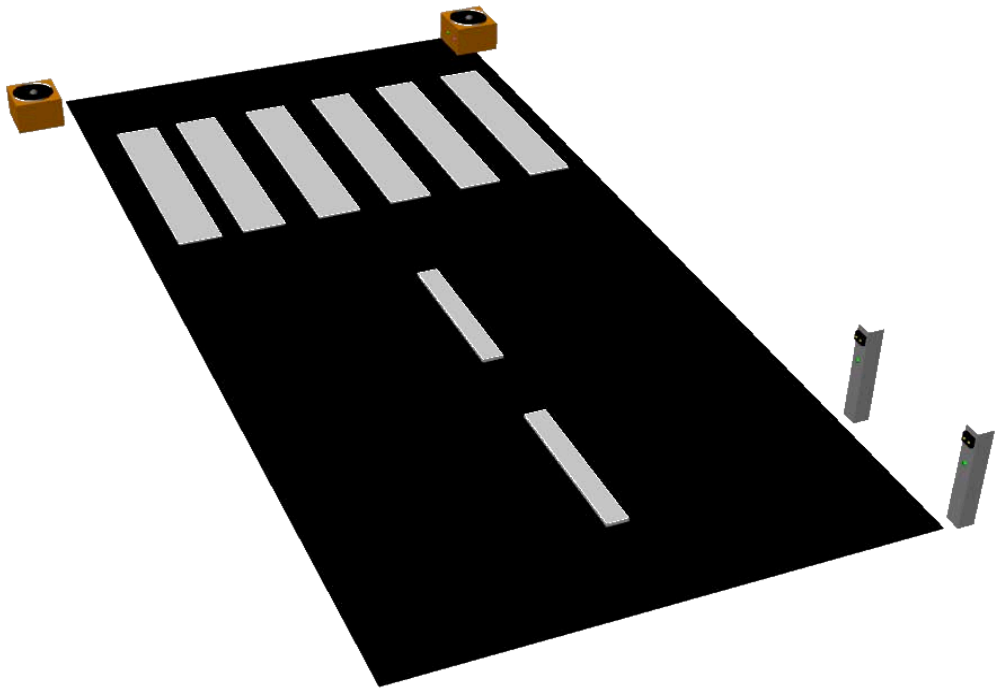


Ilustración 6. Esquema conceptual del sistema. Vista alternativa.

Cada torreta presenta embebidos un sensor y un LED de diagnóstico.

La torre de señalización y comunicaciones se presentará embebida en la electrónica principal, la radio encargada de enviar el resultado del procesado de los datos al terminal móvil que visualizará la aplicación de usuario, la cual permitirá interactuar con el sistema, introduciendo comandos con gestos táctiles sencillos que no requieran precisión, y comunicando resultados del proceso mediante texto hablado.

4.1.2 Implementación de la comunicación "interfaz de usuario-periferia"

La comunicación se implementará mediante tecnología Bluetooth 4.0 tal y como está definido en la especificación de requisitos del sistema.

El terminal tendrá previamente almacenadas en memoria las localizaciones geográficas de todas las balizas de comunicaciones existentes en el sistema.
Por otro lado, irá realizando un rastreo de la posición del usuario mediante el módulo GPS del terminal.

Cuando detecte que la distancia de la posición geográfica del usuario respecto de la posición almacenada de la baliza más próxima, es menor que una distancia definida que se corresponderá con un rango estable de actuación de la comunicación Bluetooth 4.0, se informará mediante un comando de texto hablado al usuario y se permitirá actuar sobre la pantalla táctil para comenzar el enlace entre dispositivo móvil y torre de comunicaciones del cruce en cuestión.

Una vez enlazada la periferia con el dispositivo móvil y por tanto, con la interfaz de usuario, se procederá al reconocimiento inteligente de paso de vehículos.

4.1.3 Algorítmica de reconocimiento de paso de vehículos

La algorítmica implicada en el reconocimiento de paso de vehículos, estará basada en los siguientes parámetros:

- Distancia entre las dos torres de sensores.
- Los pulsos devueltos por los sensores en el instante de paso.
- El registro de los timestamp de cada uno de esos pulsos.
- Distancia entre el último de los sensores y el inicio del paso de peatones.
- Retardo del sensor, y por consiguiente error en la medida de velocidad.
- Tiempo medio aproximado de realización del cruce.

Éstos parámetros se utilizarán de tal manera que la práctica del cruce sea útil y segura, informando en todo momento del mayor número de riesgos en el cruce posibles, tales como tiempo restante de cruce en curso (ya iniciado) menor que tiempo de llegada de un nuevo vehículo, etc.

4.1.4 Funcionalidad de la aplicación de usuario

La información de la que dispondrá el usuario mediante la aplicación, una vez establecido el enlace y confirmada la conexión con la periferia, será la siguiente:

- Información del inicio del reconocimiento inteligente de seguridad en el cruce.
- Información de cruce seguro en el instante oportuno, o por el contrario de espera por paso de vehículos a velocidad que interfiera con el cruce.
- Una vez iniciado el cruce, información de agilización del paso en el cruce por paso de vehículo a una velocidad que puede afectar a la seguridad del mismo.
- Una vez completado el cruce, información de finalización del mismo.

4.1.5 Funcionalidades adicionales

- Cruce seguro en condiciones climatológicas adversas:
  La electrónica incorporará detectores de temperatura y humedad, de tal manera que permitan en conjunto la detección de posibles placas de hielo en el cruce para advertir al usuario de dicha situación.

- Integración con guiado GPS:
  La aplicación estará diseñada de tal manera que permita una integración directa con los trabajos desarrollados en cuestión de guiado GPS de invidentes en el DSLab de la Universidad Autónoma de Madrid.

4.2 Arquitectura del sistema

La arquitectura del sistema presentará la siguiente configuración:

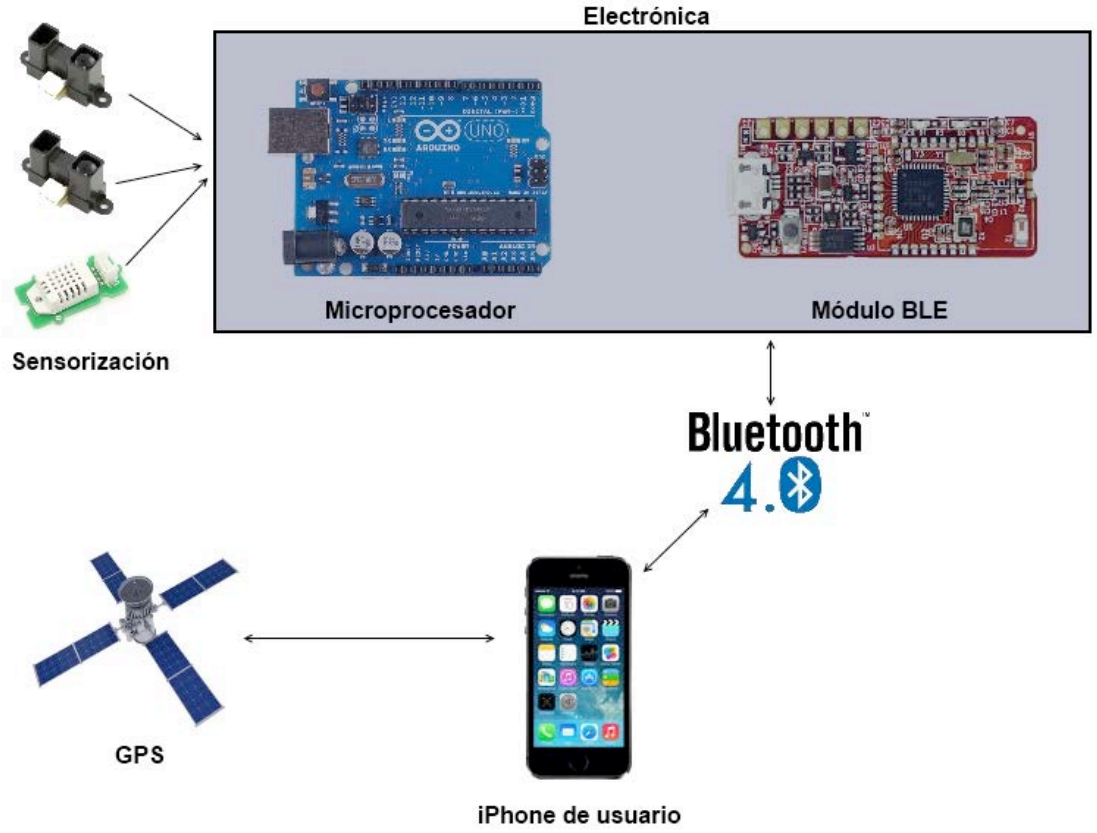


Ilustración 7. Arquitectura completa del sistema

Como se observa en la Ilustración 7, existen dos unidades principales de control de proceso:

- Unidad central de proceso de señal:
  Esta es la unidad principal de la electrónica y del proceso de las señales, cuyos elementos principales son, el microprocesador, encargado de ejecutar los algoritmos involucrados en tratamiento de los datos generados por los sensores, de ejecutar las alarmas pertinentes, y de generar las tramas del protocolo desarrollado para la comunicación inalámbrica, así como de la gestión de la misma.
  Además, la unidad central de proceso de señal, también está compuesta por el módulo de comunicaciones BLE, el cual es responsable de convertir las señales eléctricas correspondientes a las tramas que contienen la información en señales físicas interpretables por módulos BLE.
- Unidad de proceso de interfaz gráfica:
  Constituida por el dispositivo móvil del usuario. En él se ejecuta la aplicación con la interfaz gráfica y presenta los datos gestionados en la unidad central de proceso de señal de una manera útil para el usuario.

Además, esta unidad, gestiona todo lo relativo con las posiciones geográficas tanto de las balizas virtuales como la del usuario.

5 DISEÑO FINAL BASADO EN COMPONENTES COMERCIALES

A continuación se describen los sistemas involucrados en el equipo, las posibles soluciones comerciales al respecto y la solución final elegida.

5.1 Dispositivo móvil

Para la elección del terminal móvil es imprescindible que éste cuente con las siguientes características, además de las definidas en la especificación de requisitos:

- Módulo BLE (Bluetooth 4.0)
- Pantalla táctil
- Posibilidad de actuar mediante comandos de voz y texto hablado
- Configuración de accesibilidad disponible
- GPS integrado
- Conexión 3G
- Implementación de mapas

Inicialmente se han barajado terminales basados en Android e iOS, pero finalmente, la opción elegida ha sido el iPhone 5 o 4S con iOS7 debido a que:

1. Cumple todos los requisitos de la especificación.
2. Cumple los puntos citados anteriormente.
3. Se pretende una integración posterior con la aplicación de guiado GPS del DSLab.
4. Se pretende compatibilidad con iBeacons para trabajos en una posible línea de continuidad del proyecto.
5. Se ha orientado en un inicio el proyecto en el camino del SDK de XCode de Apple, por ser de las primeras compañías en apostar por el Bluetooth 4.0.

Ilustración 8. iPhone 5S

5.2 Microcontrolador

Las características requeridas para el micro-controlador, definidas en la especificación de requisitos, invitan a optar por una línea de micro-controladores compactos, destinados a aplicaciones de bajo consumo y del tipo Open Hardware, para presentar una documentación lo más accesible posible.
Dado que tanto la velocidad de transmisión de datos, como la velocidad de proceso de los mismos no son muy exigentes, se optará por buscar un modelo lo más compacto y energéticamente eficiente posible y de bajo coste.

En el mercado existen varias alternativas posibles:

- Raspberry Pi:

  La tarjeta Raspberry Pi [15]es una tarjeta compacta basada en el micro-controlador ARM1176JZF-S a 700MHz.
  Incorpora un procesador gráfico (GPU) VideoCore IV y 512MB de memoria RAM. No incluye disco duro, pero incorpora un zócalo para tarjetas SD para el almacenamiento permanente.
  La fundación de Raspberry Pi da soporte para distribuciones con arquitectura ARM, Raspbian (derivada de Debian), Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora).

[15] Plataforma Raspberry Pi: http://www.raspberrypi.org/

Además posee 1 entrada de vídeo, salidas de audio y vídeo, 8 GPIOs, puertos SPI, I^2C y 1 UART.
El consumo energético es de unos 700mA y sus dimensiones de 85,60mmx53,98mm.
Su coste en el mercado ronda los 26€.

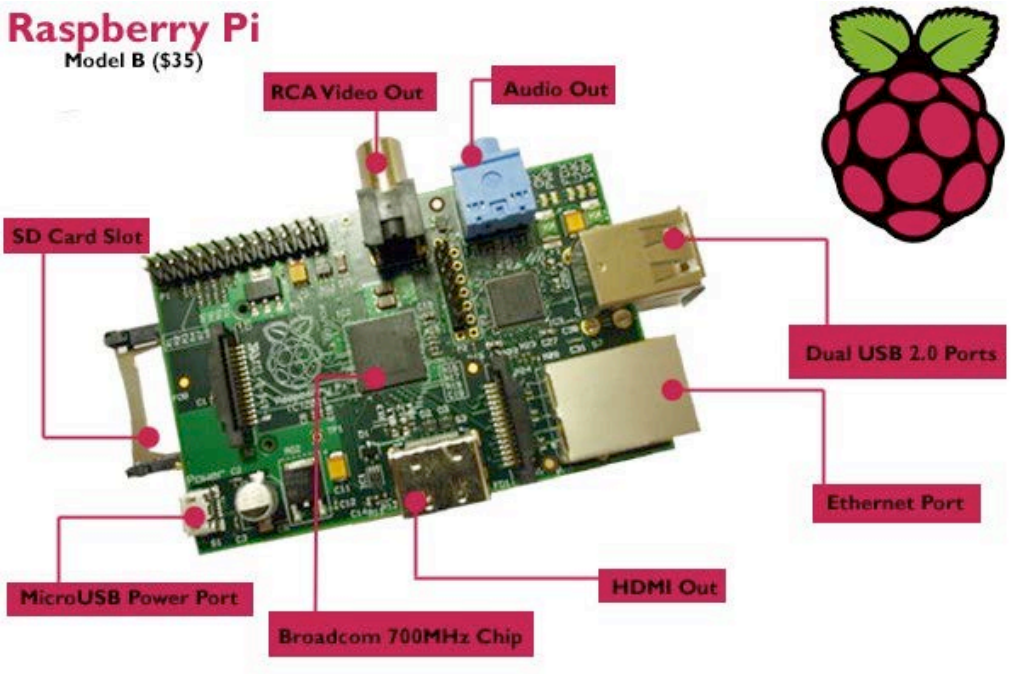


Ilustración 9. Raspberry Pi

- BeagleBoard:

La tarjeta BeagleBoard[16] es una tarjeta basada en el micro-controlador OMAP3530 a 720MHz. Incorpora una GPU PowerVR SGX530 de Imagination Technologies, que soporta OpenGL ES 2.0. Posee 256MB de memoria RAM y 256MB de memoria Flash.
Posee entrada y salida de vídeo, así como salida de Audio y ranura para SD.
Además incorpora 4 UARTs, 8 pines PWM, pines para LCD, 2 puertos SPI, 1 I^2C, Convertidor A/D, 2 CAN Bus y 4 Temporizadores.
El consumo asociado, dependiendo de la actividad y de la velocidad de proceso se estima en unos 460mA.
Las dimensiones de la tarjeta BeagleBoard son de 86,40mmx53,3mm y su precio ronda los 70€.

[16] Plataforma BeagleBoard: http://beagleboard.org/Products/BeagleBoard

Ilustración 10. Beagleboard

- Waspmote

  Waspmote[17] es una plataforma Open Source, para redes inalámbricas de sensores, centralizada especialmente en sistemas con necesidades en cuanto al consumo exigentes.

  La tarjeta está basada en el micro-procesador ATMega1281 de Atmel a 8 o 14MHz según el modelo.

  Incorpora una pequeña memoria RAMde 8KB y Flash de 128KB. Además ofrece un zócalo para tarjetas SD.

  Posee 7 entradas analógicas, 8 pines de E/S digitales 4 UARTs, puerto SPI, I^2C, 15 pines PWM y presenta un consumo máximo de 9mA en ciclos de trabajo elevados y 0.7µA en estado de reposo.

  Además, incorpora un módulo para gestión de baterías de litio con paneles solares y sockets para radios XBee (Wifi, Zigbee, Bluetooth 2.1,… etc).
  Waspmote incorpora su propio SDK basado en C, para programar el micro de la tarjeta.

  Además posee acelerómetro,
  Su precio ronda los 160€.

[17] Plataforma Waspmote: http://www.libelium.com/products/waspmote/

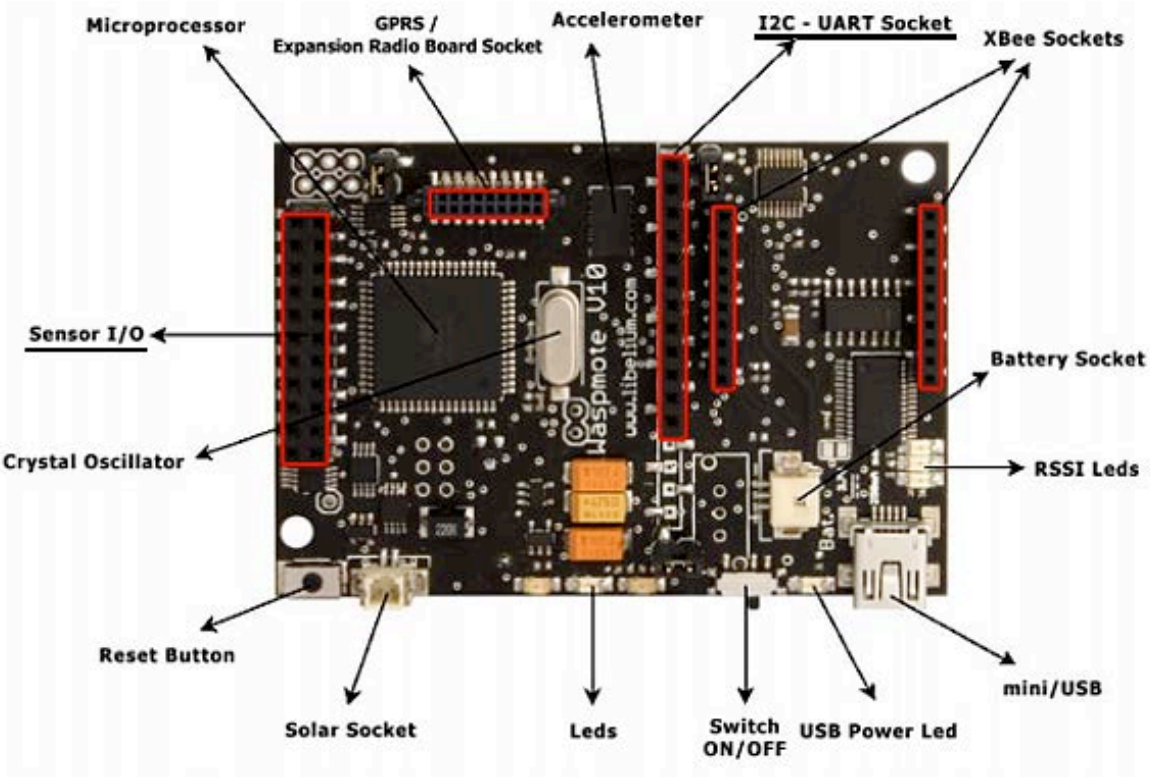


Ilustración 11. Waspmote

- Arduino

Arduino es una plataforma Open Source y Open Hardware. A diferencia de Waspmote gracias al SDK de Arduino para la programación del micro, se puede acceder directamente a cada pin y puerto de la tarjeta, independiente del módulo contectado, permitiendo controlar el hardware desde más bajo nivel, lo que admite un abanico mucho más amplio de posibilidades.

Aunque Arduino ofrece un catálogo muy amplio de tarjetas, se ha seleccionado la más adecuada a la aplicación, Arduino UNO.
Arduino UNO es una tarjeta basada en un micro-controlador ATMega 328 de Atmel a 16MHz.

Posee 14 pines de E/S digitales, 6 PWMs, 6 pines de entrada analógica, 1 UART, 1 SPI y pin de tensión de referencia. Además, su SDK permite de manera sencilla simular UARTs adicionales con pines de E/S digitales.
El consumo, dependiendo del ciclo de trabajo y de otros parámetros es desde decenas de microamperios en reposo, hasta 45mA en ciclos de trabajo mayores.

Su precio es de unos 20€.

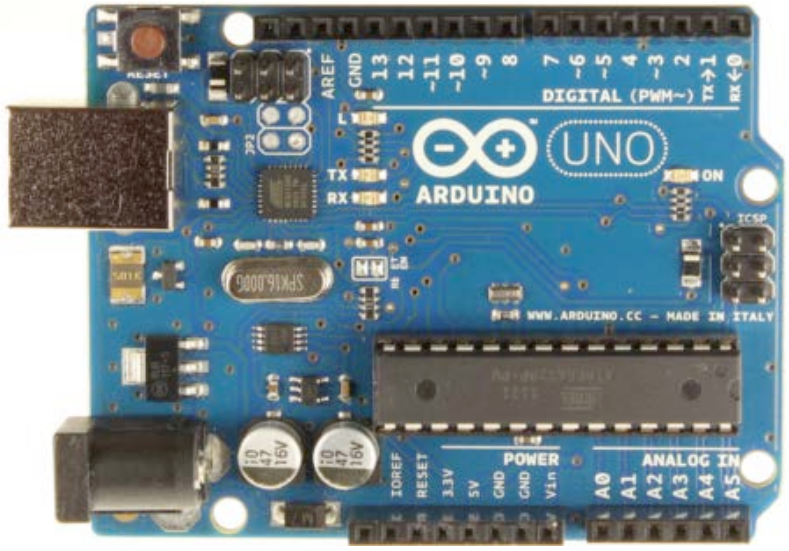


Ilustración 12. Arduino UNO

Tras este estudio de mercado de micro-controladores de bajo coste, se concluye en que, pese a las prestaciones adicionales que ofrecen tanto Raspberry Pi, como BeagleBoard e incluso teniendo en cuenta el coste atractivo de Raspberry Pi, éstas dos tarjetas están sobredimensionadas para los requisitos del sistema.

De las dos opciones restantes, Waspmote queda descartada por dos motivos:
En primer lugar, el precio es más elevado, y las características adicionales que aporta no presentan un aliciente para el proyecto. El zócalo para radios XBee no presenta utilidad, ya que esta marca no posee radios basadas en Bluetooth 4.0, por lo que la comunicación se realizaría igualmente mediante puertos serie.

En segundo lugar al no ser Waspmote Open Hardware, las posibilidades de integración de sensores y módulos de comunicaciones se ven sensiblemente limitadas frente a la plataforma Arduino.

Por ello, visto que tanto las características que mejor se adecúan a las necesidades del proyecto, tanto técnicas como de costes, se materializan en la plataforma Arduino, y en concreto en la tarjeta Arduino UNO, ésta será la elegida para la implementación de la unidad central de proceso.

5.3 Módulo BLE

La intención del proyecto es emplear un módulo BLE que integre tanto el chip de comunicaciones BLE como la adaptación de señales necesaria.

Pese a que esta tecnología es relativamente reciente, existen multitud de alternativas en el mercado, con diferentes configuraciones de chipset, memoria y pinouts.

Dado que las características de las comunicaciones no son muy exigentes, se busca un módulo compacto y económico dentro de que las prestaciones sean suficientes.

- BlueGiga BLE112 Bluetooth Smart Module[18]:

  Este módulo incorcopora el chipset BlueGiga BLE 112.

  Permite conexiones de hasta 8 clientes y en rangos de hasta 30 y 150 metros, en función del entorno, con +3dBm de potencia en transmisión y -93dBm de sensibilidad en recepción.

  La antena está integrada dentro del módulo y el consumo es de unos 36mA en el pico máximo de transmisión.

  Permite comunicación con el micro mediante I2C, SPI o UART e incorpora 23 GPIOs de propósito general.

  Sus dimensiones son de 18.15mm x 12.05mm x 2.3mm.
  El precio es de unos 50€.

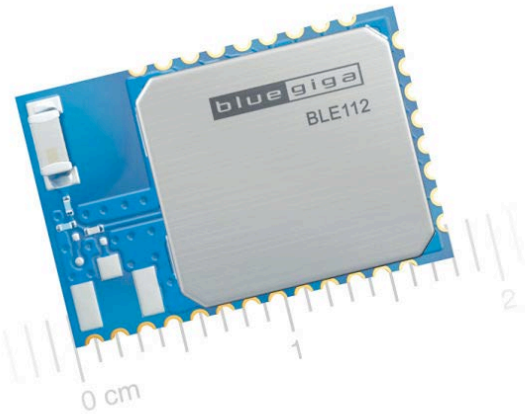


Ilustración 13 Módulo BLE112 de BlueGiga

[18] http://www.bluegiga.com/en-US/products/bluetooth-4.0-modules/ble112-bluetooth--smart-module/

- Bluetooth 4.0 BLE Module de Smart Prototyping:

  El módulo Bluetooth 4.0 BLE Module de Smart Prototyping [19]está basado en el chipset Bluetooth 4.0 Clase 2 CC2540 de Texas Instruments.

  Éste transmite a 4dBm y tiene una sensibilidad en recepción de -84dBm, con antena integrada en el propio módulo.

  La interfaz de comunicación con el micro se realiza a través de la UART con comandos AT.

  Sus dimensiones son de 26.9mm x 13mm x 2.2mm.

  El precio es de unos 10€.

Ilustración 14. Bluetooth 4.0 BLE Module de Smart Prototyping

- BLE Shield de RedBearLab

  El módulo BLE Shield de RedBearLab[20] optimizado para el entorno de Arduino está basado en el chipset Nordic nRF8001 Low Energy.[21]

  La interfaz de comunicación con el micro está implementada en puerto SPI. Y su diseño está pensado para ser pinchado en tarjetas Arduino UNO, MEGA, Leonardo y DUE.

  El chipset presenta un pico de consumo en transmisión de 12,5 mA y tasas de consumo medias de 9μA (para tiempos de conexión de 1s.)

  Ofrece una potencia en transmisión configurable a 0, -6, -12 o -18 dBm.

[19] http://smart-prototyping.com/Bluetooth-4.0-BLE-Module.html

[20] http://redbearlab.com/bleshield/

[21] http://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF8001#Download

La interfaz de comunicación con el micro permite conexión a través de SPI o UART de dos hilos (TX-RX).

Su tamaño es aproximadamente el mismo que el de la tarjeta ARDUINO UNO.

El precio ronda los 80€.

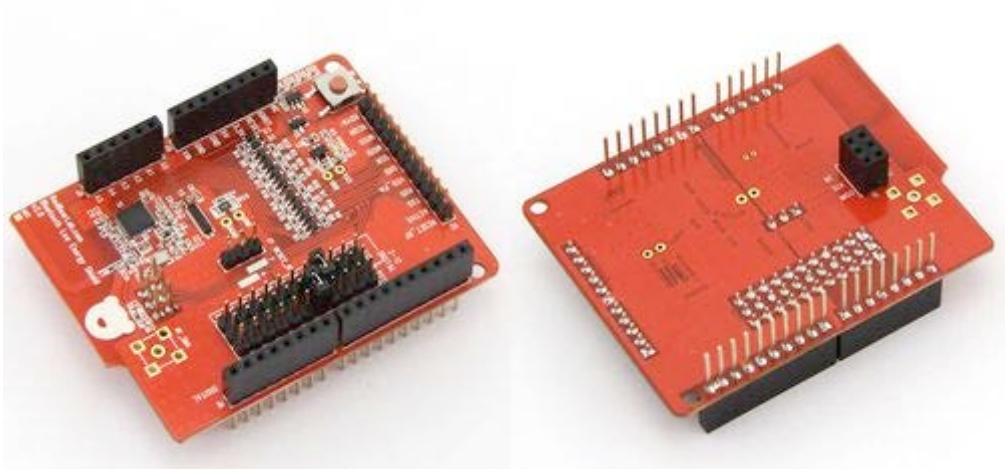


Ilustración 15. BLE Shield de RedBearLab

- BLE Mini de RedBearLab

  El módulo BLE Mini [22]de RedBearLab es una versión mejorada del BLE Shield de la misma compañía.

  Éste incorpora el chipset CC2540 de Texas Instruments, y las dimensiones son considerablemente más reducidas que su versión anterior.

  El módulo incorpora ciertos elementos programables, (ya que éste en si, puede trabajar sin microprocesador adicional) EEPROM de 512kB, LEDs y pulsador.

  Permite conexiones de hasta 8 clientes y en rangos de hasta 30 y 150 metros, en función del entorno, con +3dBm de potencia en transmisión y -93dBm de sensibilidad en recepción.

  Su precio es de unos 25€.

[22] http://redbearlab.com/blemini

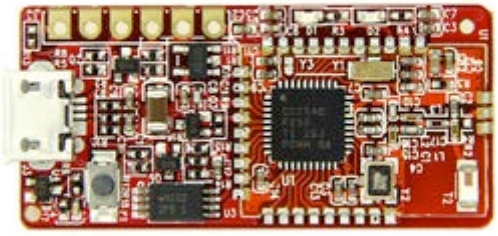


Ilustración 16. BLE mini

5.4 Sensores

5.4.1 Sensores para detección de vehículos

Para la detección de velocidad en vehículos, se pueden utilizar sensores de distinta naturaleza, en función de la magnitud física a emplear en el algoritmo de detección de velocidad:

- Detección con radares Doppler[23]:

  Los radares Doppler pueden estar basados en tecnología infrarroja o luz pulsada.
  Éste tipo de radares, emplean el efecto Doppler en los ecos de retorno de objetivos para medir su velocidad radial.

  En el mercado existen versiones de radares Doppler o pistolas Doppler económicas [24] (por poco más de 30€), sin embargo, no son productos pensados para utilizar en proyectos de desarrollo, si no que conforman productos finales con fines en los que la precisión en la medida es poco relevante.
  En la ilustración 17, se puede observar el despiece de la electrónica de una de estas pistolas Doppler de bajo coste.

  Las opciones de pistolas o radares doppler de gran precisión suponen desembolsos elevados y requerimientos de espacio que no encajan con el requisito de dimensiones reducidas descrito en la especificación del proyecto.

[23] http://science.howstuffworks.com/science-vs-myth/everyday-myths/doppler-effect3.htm

[24] Pistolas Doppler económicas: http://www.eetimes.com/document.asp?doc_id=1272375

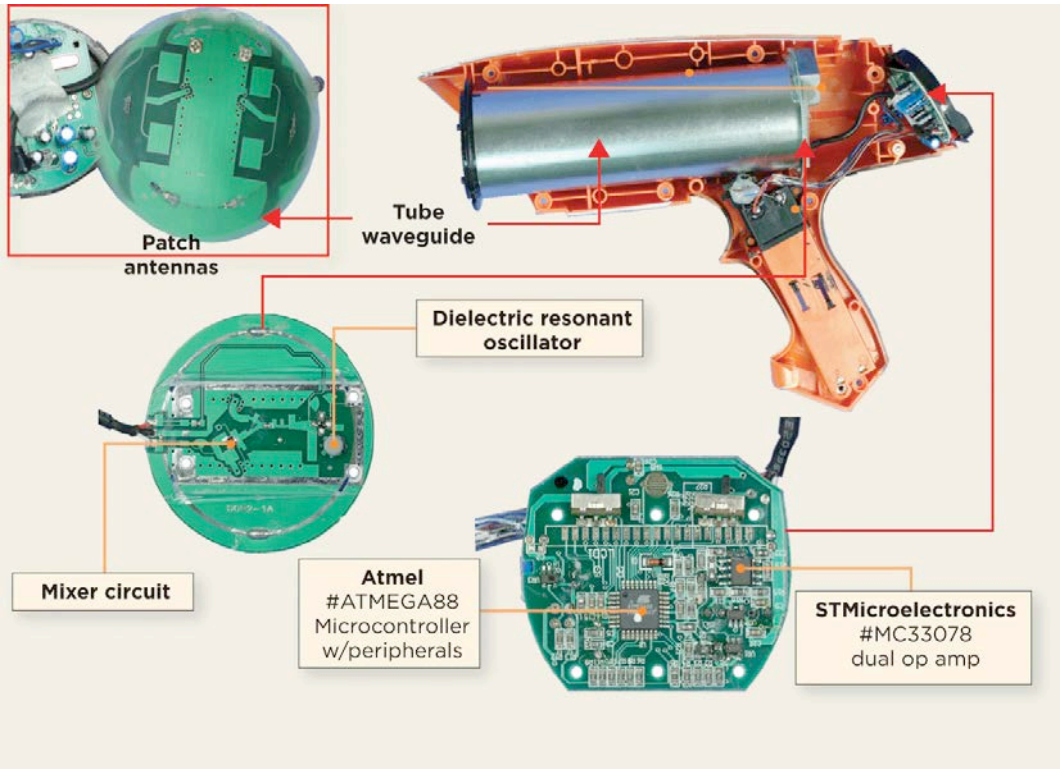


Ilustración 17. Despiece de Pistola Doppler de bajo coste.

- Detección con sensores posicionales

  Otra alternativa a los radares Doppler para medición de la velocidad de un blanco, son los sensores posicionales.

  Éstos requieren, normalmente, dos unidades sensoras separadas por una distancia conocida. Además, el blanco del cual se requiere la información de velocidad, debe realizar su paso por una distancia relativamente próxima a ambos sensores.

  El inconveniente de esta técnica respecto a la de los radares Doppler reside en la necesidad de más de un sensor para la medición, además de no poder realizar medidas a distancia y no ser aplicable a carriles con posibilidad de adelantamiento o más de un carril por sentido.

  Sin embargo, este tipo de sensores son bastante más económicos y están más extendidos.

  Para su implementación en la detección de velocidad, se utilizan algoritmos de cálculo basados en parámetros de distancias y capturas temporales.

  A continuación se muestran algunos de estos sensores:

  - Sensores infrarrojos:
    Los sensores infrarrojos se basan en la interacción entre un diodo infrarrojo emisor y un foto transistor.

En función de la intensidad lumínica, el flujo de electrones circulantes por el fototransistor, es mayor o menor, permitiendo detectar no sólo presencia de objetos en sus proximidades, si no además cómo de próximos están éstos (dentro de un rango definido).

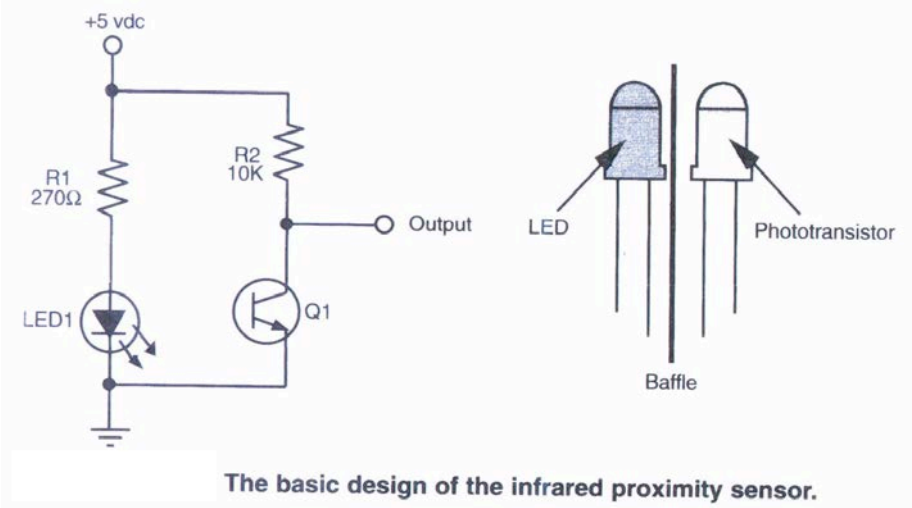


Ilustración 18. Sensor infrarrojo de proximidad

Son los detectores de presencia más sencillos y habituales. Se denominan de tipo pasivo, ya que no son capaces de "ver" a través de obstáculos, y poseen una sensibilidad baja.

Este tipo de sensores se caracterizan por detectar mejor los movimientos que cruzan el campo de visión del propio detector que los que circulan hacia o desde el detector.

Tipos de sensores infrarrojos por sus características funcionales:

- Sensores IR de corto alcance:

  Estos sensores permiten detección de objetos desde 3mm hasta unos 40cm

- Sensores IR de largo alcance:

  Permiten detección de objetos desde 15cm hasta 150cm.

- Sensores IR analógicos:

  Los sensores IR analógicos permiten detectar profundidad en la detección de objetos, es decir, cómo de próximos se encuentran éstos respecto del sensor.

  La salida de los IR analógicos suele ser en tensión.

Conociendo la resolución de la medida de tensión a la salida, se mapea ésta con la resolución adecuada permitida por el micro que gestione dicho sensor.

Ilustración 19. Sensor Sharp GP2D120 analógico de corto alcance

- Sensores IR digitales:

  Los sensores IR digitales actúan como interruptores electrónicos con la presencia de objetos.
  La distancia de detección está definida por el fabricante y no permite el análisis de profundidad de proximidad.

Ilustración 20. Sensor digital de corto alcance QRE1113 de Sparkfun

- Sensores de ultrasonidos

  Los sensores de ultrasonidos[25] se basan en la emisión de un pulso de ultrasonido con lóbulo de forma cónica.

  Dicho pulso rebota en el objetivo de medición y éste es interceptado por el receptor ultrasónico.

  El esquema de funcionamiento se muestra en la ilustración 21.

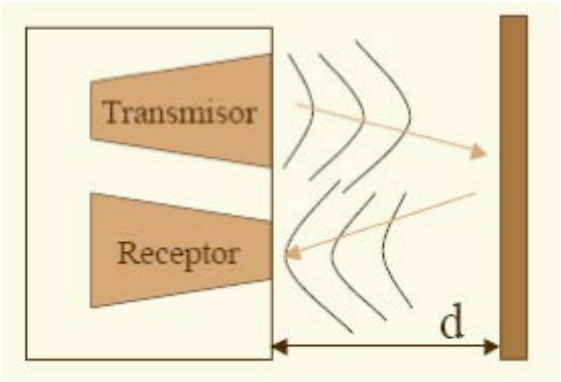


Ilustración 21. Funcionamiento básico de los ultrasonidos

Este tipo de sensores se emplean sobre todo en aplicaciones en las que se necesita conocer la distancia a la que se encuentra el objetivo.

Aún así, también son válidos para detección de presencia de objetos, utilizando dichos sensores en "modo switch", interpretando la diferencia entre la frecuencia de la onda emitida y la recibida como presencia de objetos.

Estos detectores son de tipo activo, es decir, pueden "ver" a través de objetos, por lo que permiten su integración en detección de objetos pequeños y suelen cubrir superficies mayores que los sensores IR.

El precio además, es algo más elevado que el de los sensores IR.

[25] Sensores de distancia por ultrasonidos: http://picmania.garciacuervo.net/recursos/redpictutorials/sensores/sensores_de_distancias_con_ultrasonidos.pdf

Ilustración 22. Sensor de ultra sonidos HC-SR04 de bajo coste

- Sensores capacitivos

  Este otro tipo de sensores[26] se basan en el aprovechamiento del efecto de ciertos materiales de aumentar la capacidad del sensor cuando se encuentran dentro del campo eléctrico generado.

  Se emplean como switches electrónicos en el paso de objetos de estos determinados materiales, tales como papel, plástico, aceite, metales, … etc.

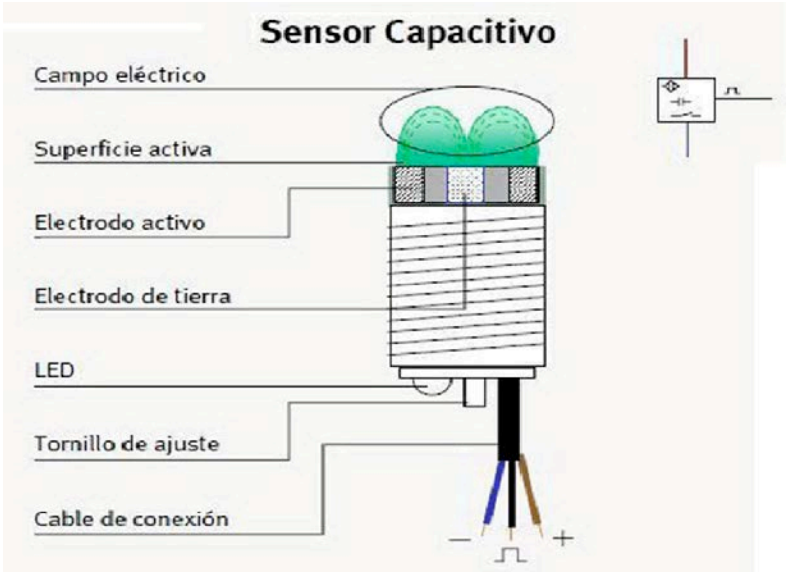


Ilustración 23. Estructura del sensor capacitivo

[26] Sensores capacitivos: http://www.capacitive-sensing.com

Los sensores capacitivos se utilizan sobretodo en aplicaciones de automatización industrial y domótica.

La gran desventaja para su implementación en este proyecto es la necesidad de contacto directo o gran proximidad del objeto con respecto al sensor.

Además su coste es el más elevado de los sensores a estudio.

Ilustración 24. Sensor capacitivo Banner CR

- Detección por visión artificial[27]

  Las técnicas de visión artificial son ya una realidad en gran cantidad de sistemas de reconocimiento y metrología en el mercado actual.

  Éstas se pueden implementar de múltiples formas para el ejercicio que nos ocupa.

  Una de las opciones posibles, se basa en el cálculo de trayectorias entre peatón y vehículo o detección de conflicto.

  Con ésta técnica, aplicando algoritmos de seguimiento y procesado de imagen, es posible detectar si existe posibilidad de accidente en un paso de peatones.

[27] Técnicas de visión artificial para la detección de conflicto peatón-vehículo: https://www.uclm.es/profesorado/licesio/Docencia/MM/NuevasTecnologias.pdf

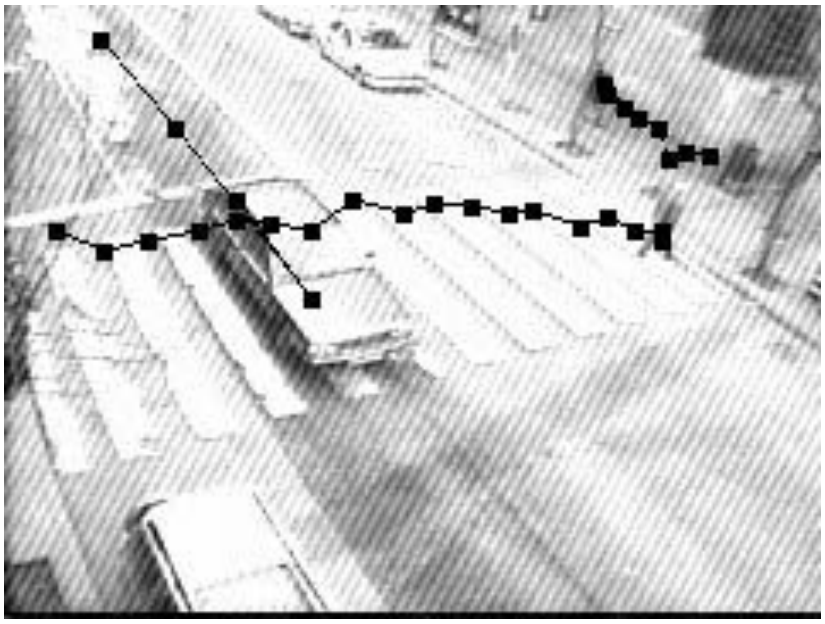


Ilustración 25. Procesado de imagen para el cálculo de trayectorias en detección de conflicto peatón-vehículo

La ventaja de este tipo de técnicas es que permite realizar un análisis multi-vehículo en varios carriles simultáneamente.

Sin embargo, estos sistemas se ven afectados por las condiciones del entorno, tanto horarias como climatológicas. De noche o con niebla o intensa lluvia los resultados son poco fiables.

También cabe destacar como consecuencia negativa que el hardware dedicado se encarece, debido principalmente a que el tratamiento de datos tanto de captura como de procesado de imagen a tiempo real necesita procesadores potentes.

Además, aunque se pueden encontrar en el mercado micro-cámaras adaptables a proyectos de bajo coste, éstas siguen siendo sustancialmente menos económicas que los sensores descritos en los puntos anteriores.

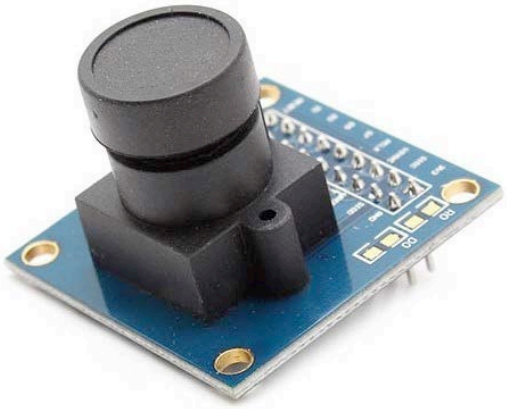


Ilustración 26. Cámara VGA OV7670

5.4.2 Sensores de temperatura y humedad

Se pretende monitorizar la temperatura y humedad ambiente con el objetivo de detectar posibles placas de hielo en la calzada que dificulten el cruce en el paso de peatones.

Dado que esta medida no requiere una precisión exigente se recurrirá a sensores de temperatura y humedad de bajo coste y tamaño reducido que sean fácilmente a la electrónica del sistema y que permita una adaptación sencilla al desarrollo del mismo.

Dado que las posibles electrónicas que manejamos como opción para el microprocesador poseen entradas analógicas en tensión así como entradas digitales barajaremos la posibilidad de utilizar sensores analógicos con salida en tensión o sensores digitales.

Sensores analógicos:
Estos sensores

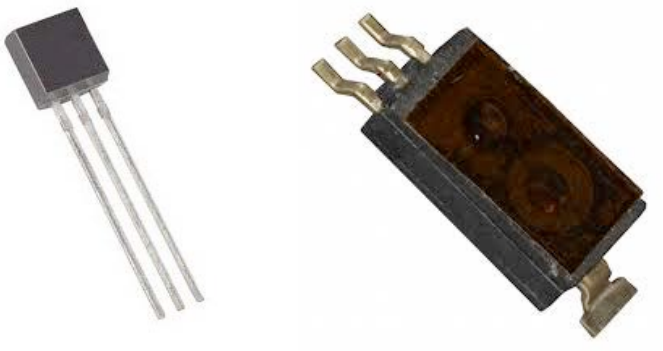


Ilustración 27. Sensores de temperatura y humedad analógicos

Sensores digitales:

Estos sensores incorporan un pequeño IC que realiza internamente una adaptación analógica de los parámetros físicos ambientales, y en función del resultado, devuelve una trama de datos digitales, mediante I2C, SPI, … etc.

Ilustración 28. Sensor de humedad SHT15 digital

La gran diferencia de estos sensores es que los datos volcados vienen pre-procesados. Los sensores analógicos a veces necesitan una tensión de referencia externa al microprocesador si ésta se ve afectada por alguna variable, sin embargo los sensores digitales no tienen esta problemática.

Además, en algunos sensores digitales, se encuentran integrados sensor de temperatura y humedad en el mismo encapsulado.

Ilustración 29. Sensor DHT22 digital de temperatura y humedad

5.5 Mecánica

La electrónica diseñada, debe acoplarse a elementos mecánicos diseñados acorde a la función que vayan a desempeñar, y a las condiciones del entorno en las que se encuentren dichos elementos.

Como se ha descrito anteriormente, contaremos en total con tres elementos mecánicos (cuatro en realidad, uno por duplicado):

- 1 elemento mecánico que configure la mecánica de la caja principal de comunicaciones
- 2 elementos mecánicos que configuren la mecánica de las torretas de detección
- 1 elemento mecánico que configure la caja auxiliar de señalización.

Conceptualmente estos diseños deben cumplir las siguientes características:

5.5.1 Caja principal de comunicaciones

La caja principal de comunicaciones debe tener un tamaño tal, que albergue de forma holgada la electrónica relativa a la dicha caja, con sus elementos periféricos.

Por tanto, será una caja de plástico (poliamida sinterizada) cúbica con IP65[28] para exteriores, y mecanizados y 3 conectores IP65 para las conexiones con las dos torretas de detección y la caja auxiliar de señalización, así como el altavoz y los 3 LEDs definidos.

El altavoz contará con su propia protección IP65.

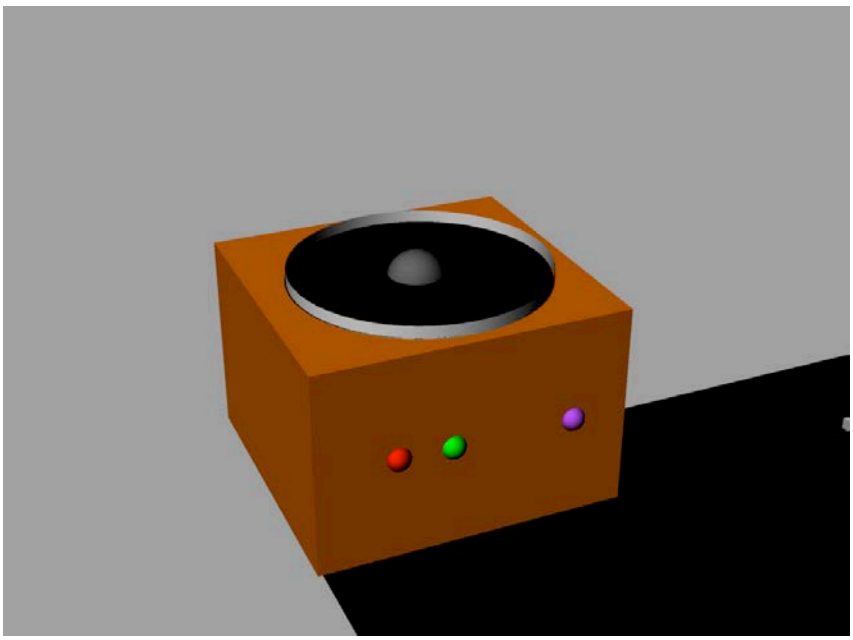


Ilustración 30. Concepto mecánico caja principal comunicaciones.

[28] Protección IP: http://www.mpl.ch/info/IPratings.html

5.5.2 Torretas de detección

El diseño mecánico de las dos torretas de detección debe tener en cuenta la funcionalidad que van a desempeñar.

Se les proporcionará una altura de 30cm respecto del suelo y se colocarán al borde de la calzada, para evitar detectar cuerpos u objetos indeseados.

Incorporarán conectores y mecanizados que respeten el IP65 definido, tanto para el LED de cada torreta, como para los sensores y los conectores asociados.

El mecanizado destinado a los sensores IR irá en la parte superior de las torretas, en la superficie que se encuentra de cara al asfalto.

Los LEDs se colocarán en la misma cara, aunque su localización no es crítica, ya que se utilizará únicamente para diagnóstico.

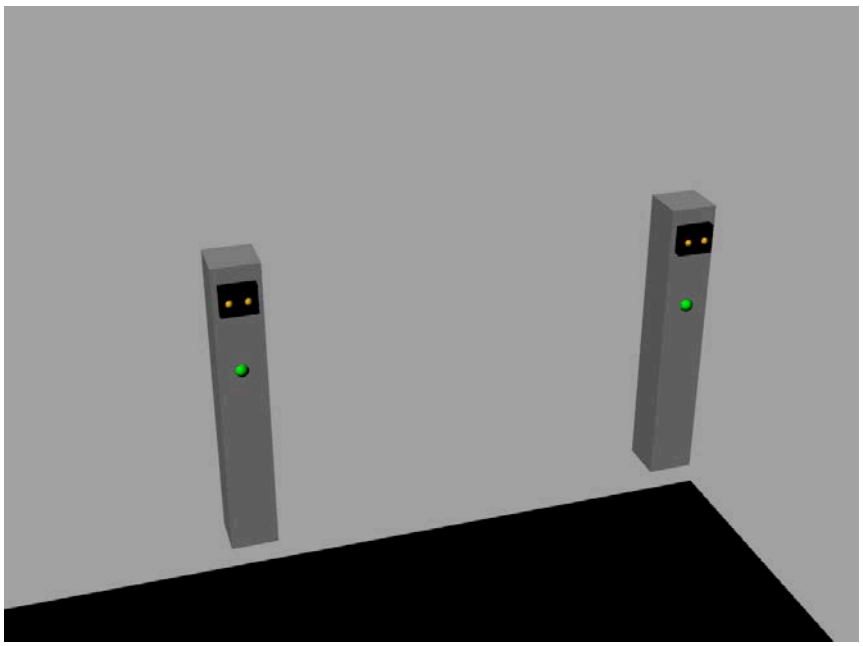


Ilustración 31. Concepto mecánico torretas detección.

5.5.3 Caja auxiliar de señalización

El diseño mecánico de la caja auxiliar de señalización, será muy similar al de la caja principal de comunicaciones.

Ésta será de forma cúbica y albergará la electrónica relativa a los elementos auxiliares de señalización.

Igualmente poseerá mecanizados para altavoz y LED y conector IP.

La instalación se realizará en el extremo opuesto de la calzada en el que se haya instalado la caja principal de comunicaciones, y la colocación del LED se ubicará con la superficie en la que se encuentra de cara al asfalto, igualmente su localización no es crítica por emplearse únicamente para diagnóstico.

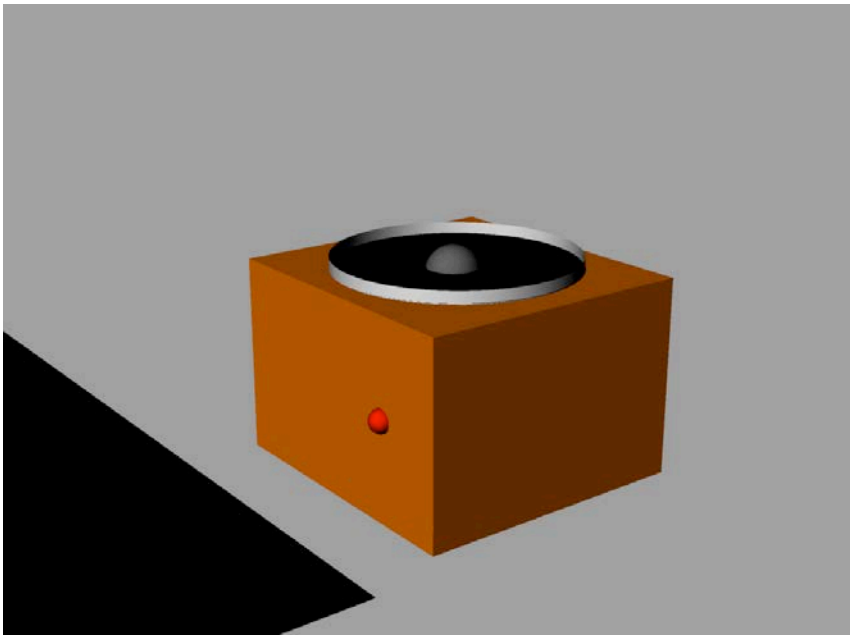


Ilustración 32. Concepto mecánico caja auxiliar señalización.

5.6 Elección final e integración de componentes

A continuación se describren los componentes elegidos para el sistema con una pequeña justificación de la elección con respecto a los requisitos de la especificación.

- Microprocesador:

  Como microprocesador se ha elegido la plataforma Arduino UNO. Los aspectos determinantes han sido:
  - el precio, conformando la opción más económica que cumplimenta la especificación con los recursos suficientes a tal efecto.
  - El tamaño, que supone una de las opciones más compactas estudiadas.
  - El consumo, que supone una de las opciones de más ahorro energético de las estudiadas.
  - El SDK de Arduino, que supone una plataforma sencilla y abierta.

- Módulo BLE:

  Como módulo BLE se ha escogido el BLE mini de RedBearLab. Éste módulo incorpora el chipset CC2540 de Texas Instruments, que es el más extendido, probado y el que posee mejor relación calidad/precio de los tres.

  El módulo Módulo Bluetooth 4.0 BLE Module de Smart Prototyping, que es algo más económico e incorpora este mismo chipset, no ofrece características interesantes como la EEPROM, el puerto USB para carga de actualizaciones de firmware que sí ofrece el módulo de RedBearLab por un poco más.

  El módulo de BlueGiga presenta grandes prestaciones, pero el precio se dispara como inconveniente.

- Sensores:

  Como sensores de detección de aproximación de vehículos, se ha decidido emplear dos torretas con sensores IR de proximidad de largo alcance analógicos, colocadas a distancias conocidas, y emplear algoritmos de cálculo de velocidad y de conflicto peatón-vehículo por parámetro de velocidad de paso.

  La razón principal es la necesidad de emplear sensores económicos, de fácil implementación, y que requieran tasas de transmisión de datos considerablemente bajas, con lo que los radares Doppler y las cámaras para visión artificial quedan fuera del alcance del proyecto.

  Los sensores capacitivos requieren distancias sensor-objeto irrealizables o dificultosas en el marco de trabajo de manera práctica, por lo que igualmente se descartan.

Los sensores de ultrasonidos presentan el inconveniente de detectar objetos detrás de un posible vehículo en tránsito detectado, y por consiguiente falsear la medida.

Por tanto, la solución más coherente en términos de compromiso entre el precio, la realizabilidad práctica de una forma sencilla, y la eficacia funcional reside en los detectores de proximidad IR de largo alcance, en concreto el modelo Sharp GP2Y0A21YK (ver datasheet).

Respecto a los sensores de temperatura y humedad, se ha decidido incorporar un sólo sensor para ambas mediciones, por cuestiones de ahorro de espacio y energético y simplicidad de integración.

Esta decisión nos lleva a la elección de un sensor digital, que volcará los datos de las mediciones al micro en tramas que transportarán las mediciones realizadas, en concreto el sensor escogido es el DHT11 (ver datasheet).

6 ESTIMACIÓN DE COSTES Y PLANIFICACIÓN

6.1 Costes del proyecto

La estimación de costes del proyecto se puede consultar el apartado Presupuesto del final del documento.

6.2 Planificación del proyecto

La planificación del proyecto se muestra en la siguiente página.

Dicha planificación, se ha dividido en 5 grandes hitos generales, que han quedado enmarcados de la siguiente manera, en función de la duración estimada de los mismos y su fecha estimada de finalización.

Plazos HITOS		
HITO	**Duración**	**Finalización**
1. Especificación de requisitos definida	4 semanas	Semana4ª
2. Diseño del sistema completo finalizado.	8 semanas	Semana 12ª
3. Electrónica del sistema fabricada y disponible.	12 semanas	Semana 24ª
4. Firmware de la electrónica y aplicación de usuario finalizados.	17 semanas	Semana 41ª
5. Producto final terminado y validado.	12 semanas	Semana 48ª

	Cronograma ejecución											
Tarea	**Mes 1**	**Mes 2**	**Mes 3**	**Mes 4**	**Mes 5**	**Mes 6**	**Mes 7**	**Mes 8**	**Mes 9**	**Mes 10**	**Mes 11**	**Mes 12**
1. Análisis de mercado y especificación de requisitos	1											
1.1. Análisis mercado aplicaciones similares												
1.2 Análisis de mercado de componentes												
1.3 Definición de objetivo y especificación de requisitos												
2. Diseño del sistema		2										
2.1 Diseño arquitectura del sistema												
2.2 Diseño preliminar HW con componentes comerciales												
2.3 Diseño interfaz de comunicaciones												
2.4. Diseño estructura SW												
3. Desarrollo HW				3								
3.1 Desarrollo de los esquemas electrónicos												
3.2 Compra y recepción de componentes HW												
3.3 Fabricación de los elementos del prototipo electrónico												
3.4 Puesta a punto												
4. Desarrollo SW										4		
4.1 Desarrollo del Firmware de la electrónica												
4.2 Desarrollo de la interfaz de comunicaciones												
4.3 Desarrollo de la aplicación iOS												
4.4 Integración con funcionalidad de reconocimiento de balizas												
5. Fase pruebas generales												5
5.1 Pruebas interfaz comunicaciones												
5.2 Pruebas funcionalidad reconocimiento de balizas												
5.3 Pruebas sistema completo												

7 DESARROLLO

A continuación se procede a la descripción de la fase de desarrollo del proyecto.
Esta parte estará dividida en dos bloques fundamentales: Las balizas y la aplicación de usuario.
A su vez, éstos bloques se dividen en sub-bloques específicos de trabajo.
Estos sub-bloques son:

- Fabricación de prototipos electrónicos de las balizas.
- Desarrollo del firmware de la electrónica.
- Desarrollo software de la aplicación de usuario.

Además como introducción, se explicará la preparación de las herramientas necesarias para el desarrollo.

7.1 Preparación de las herramientas de desarrollo

Las herramientas básicas de desarrollo que emplearán en esta fase del proyecto, son fundamentalmente 3:

- Fritzing, para la realización de esquemáticos de módulos electrónicos basados en Arduino
- SDK de Arduino, para la programación del firmware de la electrónica
- XCode, que utilizaremos para el desarrollo de la aplicación de usuario.

7.1.1 Fritzing

Fritzing es una herrmamienta para el desarrollo de esquemáticos electrónicos.

La gran ventaja de Fritzing frente a otros entornos de esquematización y modelado de circuitos, es que este está pensado para proyectos basados en la plataforma Arduino y otras plataformas similares.

De esta manera, incorpora modelos prediseñados con placas comerciales y su pinout correspondiente, facilitando la tarea de modelado de dichos módulos y haciendo extremadamente sencillo el proceso de routing e interconexión.

Además, se pueden añadir modelos adicionales gracias a la gran cantidad de librerías de Fritzing que los fabricantes ofrecen de sus catálogos de productos.

También posee librerías con los componentes necesarios para adaptación de señales, los cuales también incluyen una referencia comercial.

Sin embargo, hay que destacar, que esta herramienta sólo permite realizar esquemáticos a modo de plano de interconexiones para el montaje físico. No posee herramientas de compilación hardware ni simulación.

Por este motivo, es muy importante poseer la documentación de los módulos que se van a utilizar y asegurarse de que se hace un routing y una adaptación electrónica (en caso de aplicar) correctos.

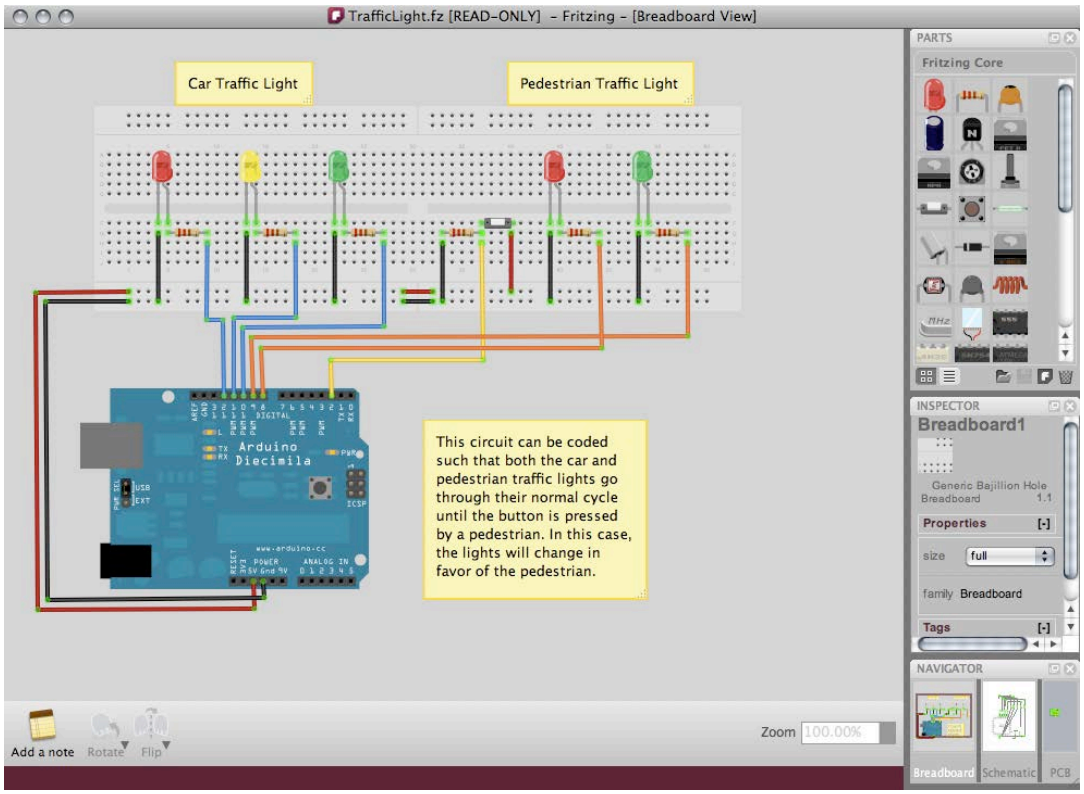


Ilustración 33. Captura del entorno de Fritzing

Fritzing se puede descargar de manera totalmente gratuita en el siguiente enlace: http://fritzing.org/download/

Éste está disponible para los sistemas operativos Windows XP y posteriores, Mac OS X 10.5 y posteriores y Linux de 32 y 64 bits.

Además, por ser de código abierto, éste se puede descargar en el siguiente enlace: http://fritzing.org/download/0.8.7b/source-tarball/fritzing-0.8.7b.source.tar.bz2

7.1.2 SDK de Arduino y librerías asociadas al HW

Se utilizará el propio SDK de Arduino para la programación del firmware del microprocesador.

Es un entorno de código abierto, el cual está escrito en java y basado en processing, avr-gcc y otra serie de programas también de código abierto.

El software de Arduino se puede descargar de manera gratuita en la siguiente url: http://arduino.cc/es/main/software#.Uv5PQBbl17s.

En concreto, la versión utilizada para este proyecto, ha sido la 0019, versión hospedada en Google Code.

Dicha versión es compatible con Windows, Mac OS X y Linux.

El código fuente del software Arduino puede ser navegado online o descargado de forma gratuita.

Se podrían haber utilizado otros entornos más potentes y con más funcionalidades como Atmel Studio, sin embargo, por simplicidad se ha optado por trabajar en el SDK de Arduino.

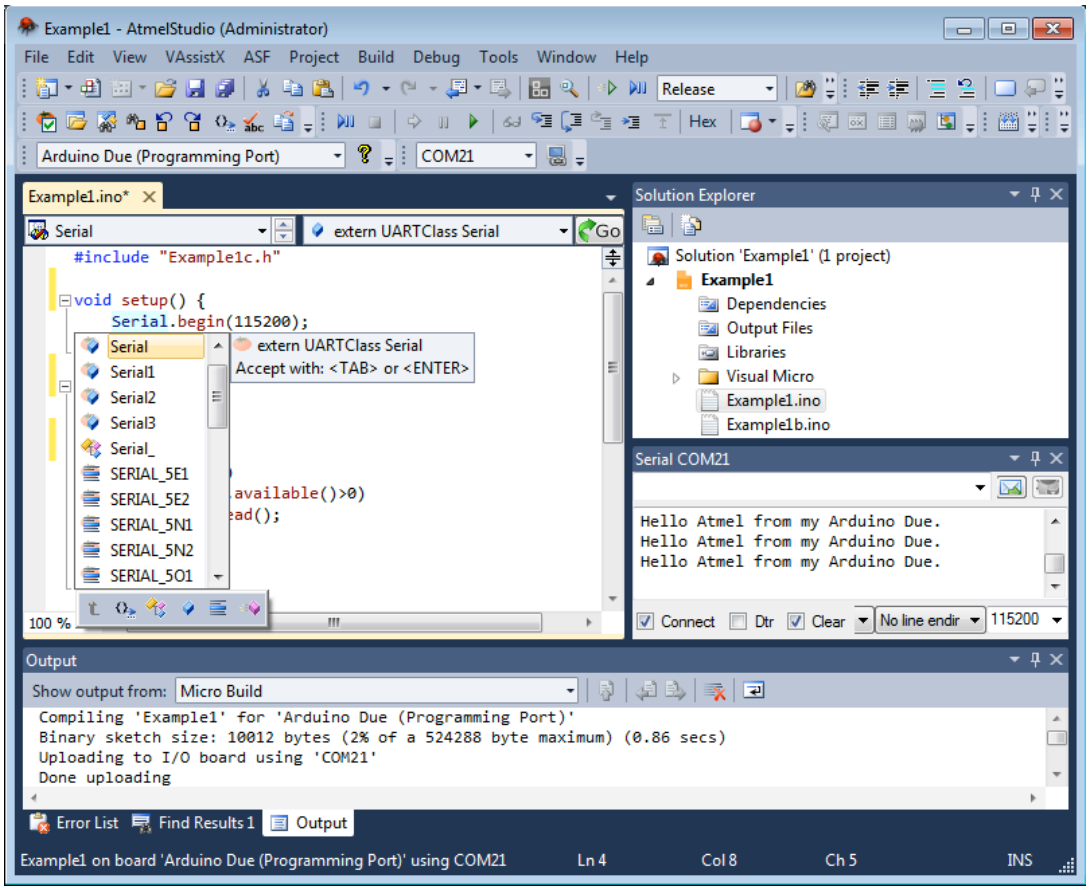


Ilustración 34. Captura de pantalla de Atmel Studio 6.1

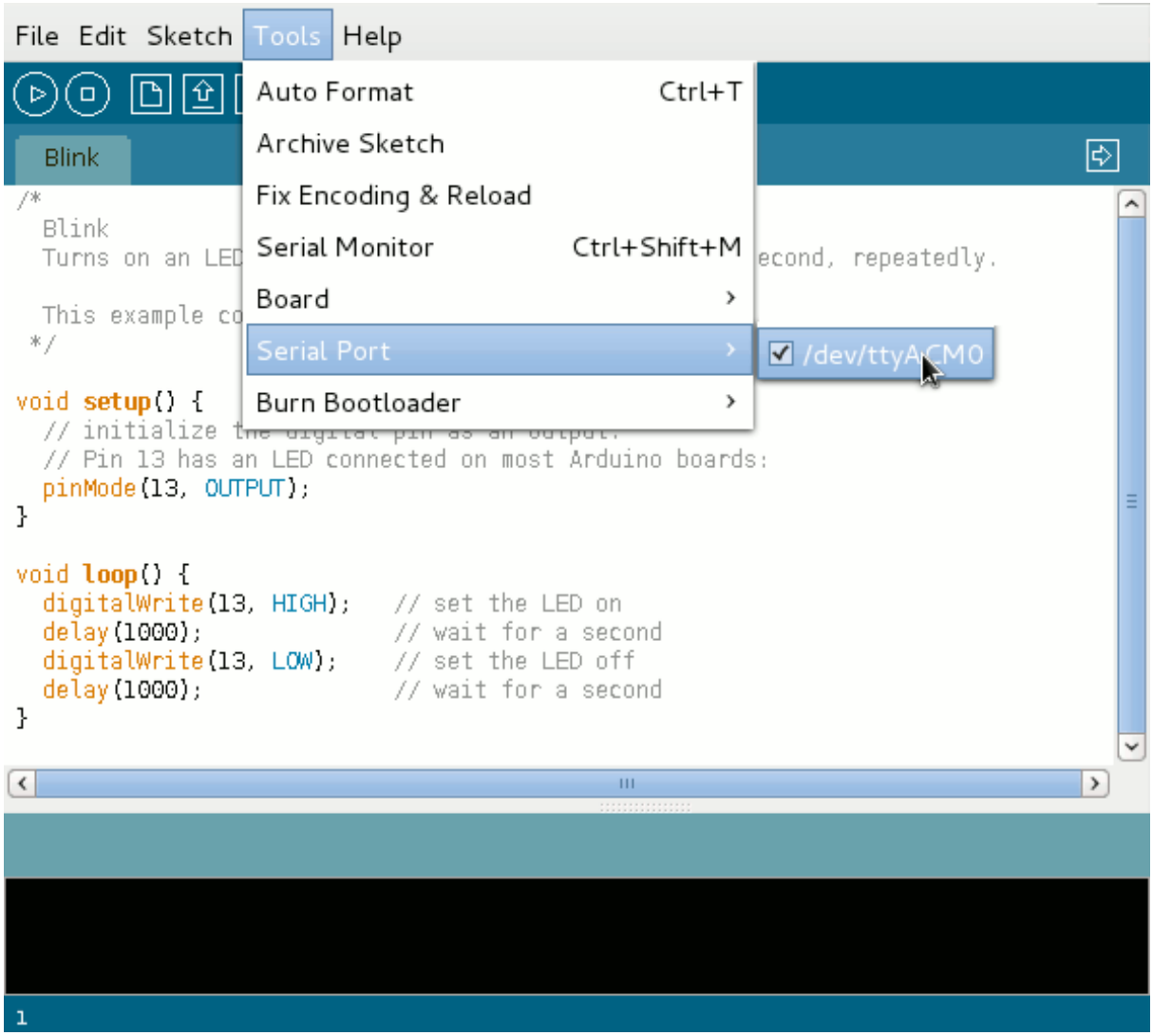


Ilustración 35. SDK de Arduino

El SDK de Arduino permite importar librerías externas específicas para ciertas acciones de comunicaciones, o manejo de ciertos sensores comerciales.

Para este proyecto se han empleado las siguientes librerías basadas en el SDK de Arduino:

- ble_mini.h:
  Esta librería es la que posibilita la comunicación entre el micro de Arduino y el módulo BLE (Para consultar el código de la misma, ver ANEXO A).

  Gracias a ella podemos utilizar las instrucciones:

  - BLEMini_begin(): Esta instrucción habilita el interfaz de comunicación serie del micro para la comunicación con el módulo módulo BLE Mini.

  - BLEMini_available(): Esta función realiza una consulta al módulo BLE, que contesta con un valor entero. En función del valor devuelto por esta instrucción, se obtiene el estado del módulo BLE.

- BLEMini_write(): Mediante esta instrucción, escribimos 1byte en la línea serie que llega al módulo BLE y éste se encarga de reproducir a través de Bluetooth.

- BLEMini_read(): Esta instrucción permite recibir mediante la línea serie del microprocesador el byte recibido a través de bluetooth en el módulo BLE Mini.

- DHT.h:
  Esta librería permite la comunicación digital entre el sensor de temperatura y humedad DHT y el micro (Para ver el código fuente de la misma ver ANEXO B).

  Es una librería muy sencilla que nos permite utilizar las siguientes funciones:

  - Dht.begin(): Inicializa el protocolo de lectura de tramas digitales.

  - Dht.readTemperature(): Esta función hace transparente al desarrollador el formato de tramas de comunicación y vuelca directamente la temperatura en punto flotante en grados Celsius.

  - Dht.readHumidity(): Al igual que la lectura de temperatura, esta función hace transparente al desarrollador el formato de tramas de comunicación y devuelve directamente la humedad en entero en % de humedad relativa.

- pitches.h:
  Esta librería permite reproducir sonidos identificables a través de la conexión digital de los speakers. (Para ver el código fuente de la librería pitches.h ver ANEXO C)

  Esto se realiza a través de las siguientes instrucciones:

  - tone(a,b,c):
    Esta instrucción recibe los parámetros a: número de pin en el que se encuentra el speaker a través del cual se quiere reproducir el sonido, b: nota de la escala musical que se desea reproducir en notación americana (A: La, B: Si, C: Do, D: Re, E: Mi, F: Fa, G: Sol; y el número de la octava) con el formato NOTE_XY, donde X es la nota e Y la octava (Por ejemplo NOTE_C3 equivaldría al Do 3), y c: duración de la nota en fracciones de negra, donde 4 equivale a un cuarto de negra, 8 a un octavo, … etc.

  - noTone(a):
    Mediante esta instrucción se detiene la reproducción del sonido en curso en el speaker del pin a.

7.1.3 XCode y librerías asociadas al SW

XCode es un IDE (*Integrated Developmnet Environment*, Entorno de Desarrollo Integrado en inlgés) de la compañía Apple.

Éste se puede adquirir de manera gratuita en el website de Apple, y se suministra (también de manera gratuita) con el sistema operativo Mac OS X.

Además, en las versiones recientes de XCode, éste trabaja conjuntamente con Interface Builder. Interface Builder es una herramienta heredada de la compañía Next, que se emplea en la creación de interfaces gráficas de usuario.

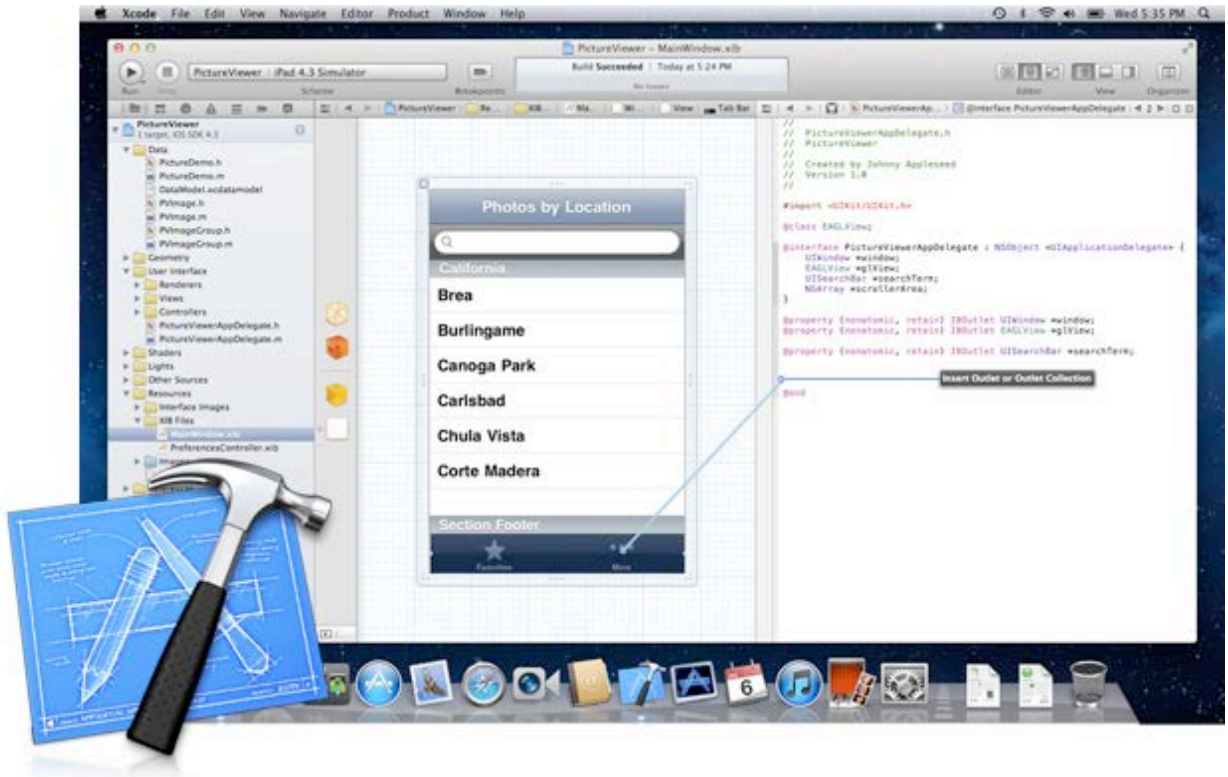


Ilustración 36. Captura de pantalla de la herramienta XCode en su versión 5.0

XCode incluye la colección de compiladores del proyecto GNU (GCC) y es capaz de compilar código escrito en los lenguajes C, C++, Objective-C, Objective-C++ y Java, y además posee soporte para GNU Pascal[29], Free Pascal[30], Ada y Perl[31].

Se pueden realizar desarrollos en XCode de manera gratuita, con la limitación de poder probar las aplicaciones creadas únicamente en el simulador y no poder subir dichas aplicaciones a la tienda online AppStore.

[29] http://www.microbizz.nl/gpcxcode.html

[30] http://pascal-central.com/fp-xcode/

[31] http://camelbones.sourceforge.net/index.html

Para poder realizar el debugging del código de la aplicación corriendo el propio dispositivo es necesario adquirir una cuenta de desarrollador iOS en la web de Apple, por un precio de 99$ al año desde esta url: https://developer.apple.com/programs/ios/

Con esta cuenta de desarrollador iOS, además se pueden subir las aplicaciones desarrolladas a la AppStore.

Además de la adquisición de la cuenta de desarrollador iOS es necesario descargar un certificado de desarrollador en cada máquina que utilicemos para desarrollo desde el Member Center de la página de desarrolladores de Apple desde esta web: https://developer.apple.com/account/ios/certificate/certificateList.action

Para poder descargar dicho certificado es necesario poseer un identificador de organización de desarrollo. En el caso concreto de este proyecto se utiliza el identificador fuam de la fundación UAM para el DSLab de la Universidad Autónoma de Madrid

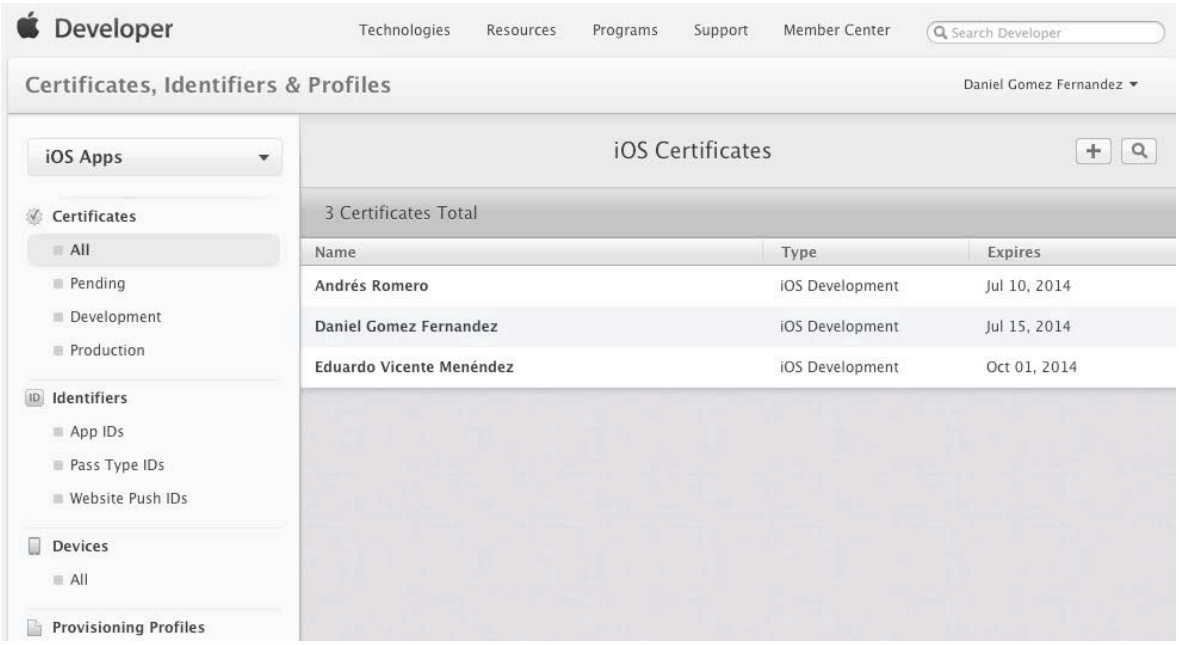


Ilustración 37. Detalle de la lista de certificados de desarrolladores iOS del DSLab en la web del desarrollador de Apple

Es importante tener en cuenta que los certificados mencionados caducan en la fecha de expiración de la licencia de desarrollador, por lo que, al renovarla, será necesario generar y descargar una nueva.

Además es necesario registrar el dispositivo destinado a desarrollo a utilizar y emparejarlo a uno de los *Provisioning Profiles* o cuentas de desarrollador.

Se pueden vincular 99 dispositivos diferentes al identificador de organización de desarrollo de App creado.

Librerias SW y frameworks de XCode utilizados:

Para el desarrollo de la aplicación, se han utilizado las siguientes librerías y frameworks:

- *CoreLocation.framework*

  El framework CoreLocation[32] permite identificar la localización geográfica actual del dispositivo a través del hardware disponible para determinación de posiciones geográficas (módulo GPS) y realizar un rastreo en el tiempo de la misma.

  Se utilizan las clases y protocolos del CoreLocation framework para configurar y programar la petición de localización y eventos de rastreo.

  De igual manera, se puede emplear para definir regiones geográficas y monitorizar el instante de tiempo en el que el usuario cruza los límites de estas regiones.

  Como funcionalidad añadida e interesante en el marco de este proyecto, el framework CoreLocation permite definir regiones alrededor de balizas Bluetooth [33]o iBeacons.

- *AudioToolbox.framework*

  El framework AudioToolbox[34] proporciona una interfaz para grabación, reproducción y análisis de streaming de audio.

  Adicionalmente, este framework proporciona herramientas para administrar sesiones musicales.

- *CoreBluetooth.framework*

  El framework CoreBluetooth[35] proporciona las clases necesarias para la comunicación con dispositivos que están equipados con tecnología BLE.

[32] CoreLocation.framework: https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocation_Framework/_index.html

[33] Clase CLBeaconRegion : https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CLBeaconRegion_class/Reference/Reference.html#//apple_ref/doc/uid/TP40013054

[34] AudioToolBox.framework: https://developer.apple.com/library/mac/documentation/musicaudio/reference/CAAudioTooboxRef/_index.html

[35] CoreBluetooth framework: https://developer.apple.com/library/ios/documentation/CoreBluetooth/Reference/CoreBluetooth_Framework/_index.html

- *Foundation.framework*

  El framework Foundation[36], presente por defecto en todos los proyectos de XCode, define una capa base de clases Objective-C.

  Adicionalmente a la provisión de una serie de clases objeto primitivas, añade numerosos modelos que definen funcionalidades no cubiertas por lenguaje Objective-C.

  Según la web de referencia para desarrolladores de Apple, Foundation framework se ha diseñado con los siguientes objetivos:

  - Proporcionar un pequeño set de clases y utilidades básicas.
  - Hacer el desarrollo software más sencillo, introduciendo convenciones consistentes para temas como liberación de memoria o registros.
  - Soportar strings Unicode, persistencia de objetos y distribución de objetos.
  - Proporcionar un nivel de independencia de sistema operativo, para mejorar la portabilidad.

  El Foundation framework incluye las clases objeto raíz, clases que representan tipos de datos básicos como strings o byte arrays, colecciones de clases para almacenamiento de objetos, clases que representan información de sistema como fecha, hora, …; y clases que representan puertos de comunicación.

- *CoreGraphics.framework*

  El framework CoreGraphics[37] es una API basada en C que se fundamenta en el motor gráfico avanzado Quartz.

  Proporciona rendering 2D ligero de bajo nivel, de gran fidelidad.

  Se utiliza este framework para manejar modelado *path-based*, transformaciones, administración de color, renderizado *offscreen*, patrones, gradientes y sombreados, administración de datos de imagen, creación de imagen, enmascarado, y creación y visualización de documentación en PDF.

- *UIKit.framework*

[36] Foundation.framework: https://developer.apple.com/library/mac/DOCUMENTATION/Cocoa/Reference/Foundation/ObjC_classic/_index.html

[37] CoreGraphics.framework: https://developer.apple.com/library/ios/documentation/CoreGraphics/Reference/CoreGraphics_Framework/_index.html

El framework UIKit,[38] proporciona las clases necesarias para construir y administrar interfaces de aplicación de usuario para iOS.

Suministra objetos de aplicación, manejo de eventos, modelado gráfico, ventanas, vistas, y controles específicos para interfaces de pantalla táctil.

- *XCTest.framework*

El framework XCTest [39] permite editar tests de aplicación y ejecutarlos de uno en uno o como grupos de test, dentro del mismo entorno de XCode o a través de la línea de comandos.

- *BLE.Framework*

La librería BLE framework es una librería desarrollada por RedBearLab, que transparentiza ciertas instrucciones y clases del framework CoreBluetooth y está pensado específicamente para comunicación con el módulo BLE mini.

Esta librería es Open Source, se proporcionan manuales al respecto para poder adaptarla a las necesidades de cada proyecto y se puede descargar de forma gratuita en el perfil de RedBearLab de GitHub.[40]

Para ver el código fuente de esta librería ver ANEXO D.

[38] UIKit.framework: https://developer.apple.com/library/ios/documentation/uikit/reference/UIKit_Framework/_index.html

[39] XCTest framework y otras herramientas de desarrollo: https://developer.apple.com/technologies/tools/

[40] Librería BLE framework en el repositorio de RedBearLab de GitHub: https://github.com/RedBearLab/iOS

7.2 Fabricación de los prototipos HW de las balizas

A continuación se comenzará con el desarrollo Hardware relativo a la electrónica de las balizas.

7.2.1 Descripción de componentes HW

Se recuerdan los componentes principales involucrados en dicho desarrollo, comentados en el punto 5.6:

1. Arduino UNO como unidad de control central.
2. Módulo BLE Mini como unidad de comunicaciones BLE.
3. Dos sensores IR GP2Y0A21YK de detección de presencia de largo alcance.
4. Un sensor DHT11 de Temperatura y Humedad digital.

Como se ha descrito en el punto 5.5, la mecánica consta de una caja principal de comunicaciones, una caja auxiliar de señalización acústica y visual, y dos torretas de detección.

Además, se incluirán elementos de señalización adicionales a los principales de la descripción:

1. En la caja principal de comunicaciones se instalarán:
    a. Un altavoz de señalización acústica.
    b. Un LED de indicación de encendido.
    c. Un LED de indicación de paso libre.
    d. Un LED de indicación de alerta en el cruce.

Dichos LEDs únicamente se utilizan para diagnóstico del equipo.

2. En la caja auxiliar irán instalados los siguientes elementos:
    a. Un altavoz de señalización acústica.
    b. Un LED de indicación de alerta en el cruce.

3. Las torretas de detección incorporan la siguiente electrónica:
    a. Sensor de detección IR de largo alcance.
    b. Un LED indicador de detección.

Algunos de estos elementos, como se muestra en sus datasheets incorporan la adaptación electrónica correspondiente, pero otros, como los LEDs, los altavoces y el sensor de temperatura y humedad necesitan adaptaciones adicionales, que se muestran en el esquema de la ilustración 41.

Una vez clasificados todos los elementos, es necesario asignar un pinout del micro a cada uno de los pines de los diferentes elementos.

Para ello, es necesario identificar primero cuántos pines necesitamos para cada componente y de qué tipo son estos.

7.2.2 Adaptaciones y pinout

- Módulo BLE:

  El micro se comunica con el módulo BLE a través de la UART. En el datasheet del módulo BLE se puede observar que no se necesita niguna adaptación electrónica para la conexión de los pines de la UART (pines digitales TX y RX), por lo que dichos pines irán conectados directamente a los pines de la UART del micro.

  Es importante tener en cuenta que el pin TX del módulo es el pin emisor de datos del mismo, por lo que debe conectarse al pin RX de la UART del micro, que es el receptor.

- Sensores IR:

  Según el datasheet de los sensores GP2Y0A21YK, éstos poseen 3 pines correspondientes a +5V, GND y datos.

  Por tanto, los pines de alimentación +5V y GND se conectan a los pines homónimos de Arduino.

  El pin de datos es un pin analógico, que se conecta directamente al pin de la placa del microprocesador. Por tanto, necesitamos reservar dos pines analógicos del micro para ambos sensores IR.

  Reservaremos los pines analógicos 1 y 2 (A1 y A2).

- Sensor DHT11:

  El datasheet de este sensor, indica que de los cuatro pines dos son de alimentación (+5V y GND) que conectaremos directamente a la placa.

  El pin 3 del sensor no se utiliza, por lo que lo conectaremos al GND de la placa directamente.

  El pin 2 se corresponde con el pin digital de datos, el cual, según el datasheet, necesita un pull-up.

  Por tanto, el pin 2 lo conectaremos a un pin digital de la placa de Arduino, a través de una resistencia de pull-up. Por diseño, se ha elegido el pin digital 9, así que reservaremos el pin digital número 9 para el sensor de temperatura y humedad.

- Altavoces:

  Se utilizan dos altavoces de 8Ω de 2W y 3” de dos hilos. Uno de los dos hilos se conecta al pin GND del micro y el otro, utilizando una resistencia de protección de sobrealimentación a un pin digital del micro.

Ilustración 38. Altavoz de 8Ohm y 3" de 2W

Por tanto se reservarán dos pines digitales del micro a cada uno de los altavoces. Por criterio de diseño se han reservado los pines digitales 2 y 3.

- LEDs:

Haciendo recuento de los LEDs que se han descrito anteriormente, se van a utilizar un total de 6 LEDs de señalización visual para diagnóstico.

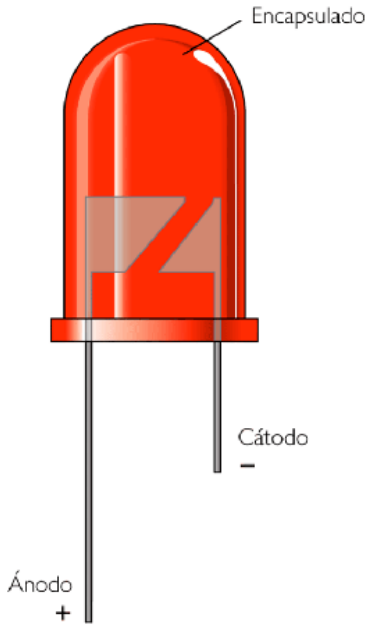


Ilustración 39. Configuración física de LEDs

Dichos LEDs requieren una conexión directa del ánodo al pin GND del micro, y una conexión del cátodo a un pin digital del micro a través de una resistencia de protección de sobrealimentación.

Por tanto, reservaremos 6 pines digitales para LEDs. Por criterios de diseño se han elegido los pines digitales 4,5,6,7,10 y 11.

Una vez identificados todos los pines reservados, el diseño del pinout queda configurado de la siguiente manera:

Pinout Sistema de balizas		
Nº Pin	Tipo Pin	Asignación
RX	Digital- UART	TX - Módulo BLE
TX	Digital-UART	RX - Módulo BLE
2	Digital	Speaker Caja auxiliar señalización
3	Digital	Speaker Caja principal comunicaciones
4	Digital	Led Torre detección 1
5	Digital	Led Torre detección 2
6	Digital	Led Paso Permitido Caja principal comunicaciones
7	Digital	Led Encendido Caja principal comunicaciones
8	Digital	Sin uso
9	Digital	Pin 2 Sensor DHT11 de temperatura y humedad
10	Digital	Led Alerta en cruce caja principal de comunicaciones
11	Digital	Led Alerta en cruce caja auxiliar señalización
12	Digital	Sin uso
13	Digital	Sin uso
A0	Analógico	Sin uso
A1	Analógico	Pin 1 Sensor IR GP2Y0A21YK torre 1
A2	Analógico	Pin 1 Sensor IR GP2Y0A21YK torre 2
A3	Analógico	Sin uso
A4	Analógico	Sin uso
A5	Analógico	Sin uso

Ilustración 40. Pinout sistema de balizas

7.2.3 Otras consideraciones de diseño

Para las conexiones de los hilos de los diferentes componentes al micro, se utilizarán conectores Amp macho de 10 pines.

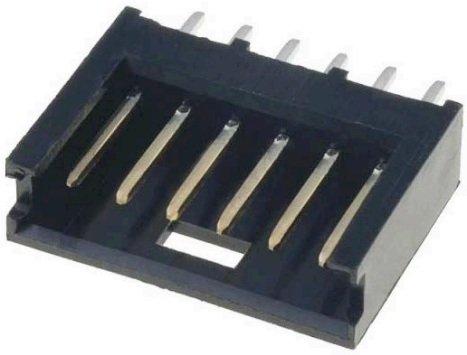


Ilustración 41. Conector AMP-MOD Macho

Para las conexiones con las cajas de la periferia se adaptarán conectores molex.

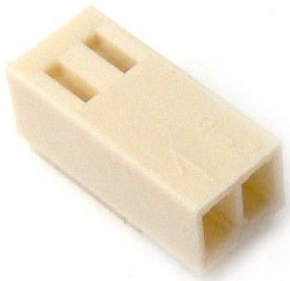


Ilustración 42. Conector Molex hembra

7.2.4 Esquema y documentación de los prototipos fabricados

En la ilustración 40 se muestra el esquema de conexiones del prototipo en el PCB diseñado con la herramienta Fritzing.

El esquema incluye todos los componentes y adaptaciones con el pinout correspondiente de la placa de Arduino.

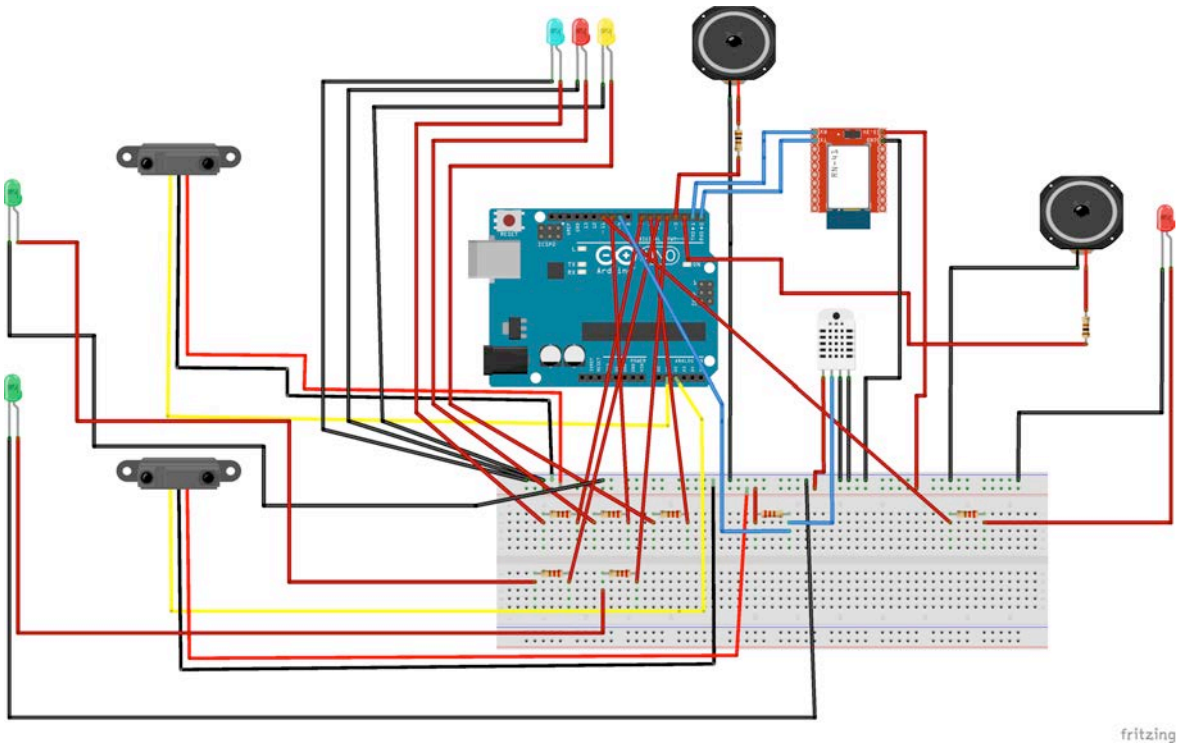


Ilustración 43. Esquema de conexiones en PCB del prototipo

La ilustración 41 muestra el esquema electrónico con el ruteado correspondiente diseñado en Fritzing.

En el ANEXO E se muestra dicho esquemático ampliado, para mayor resolución de detalles.

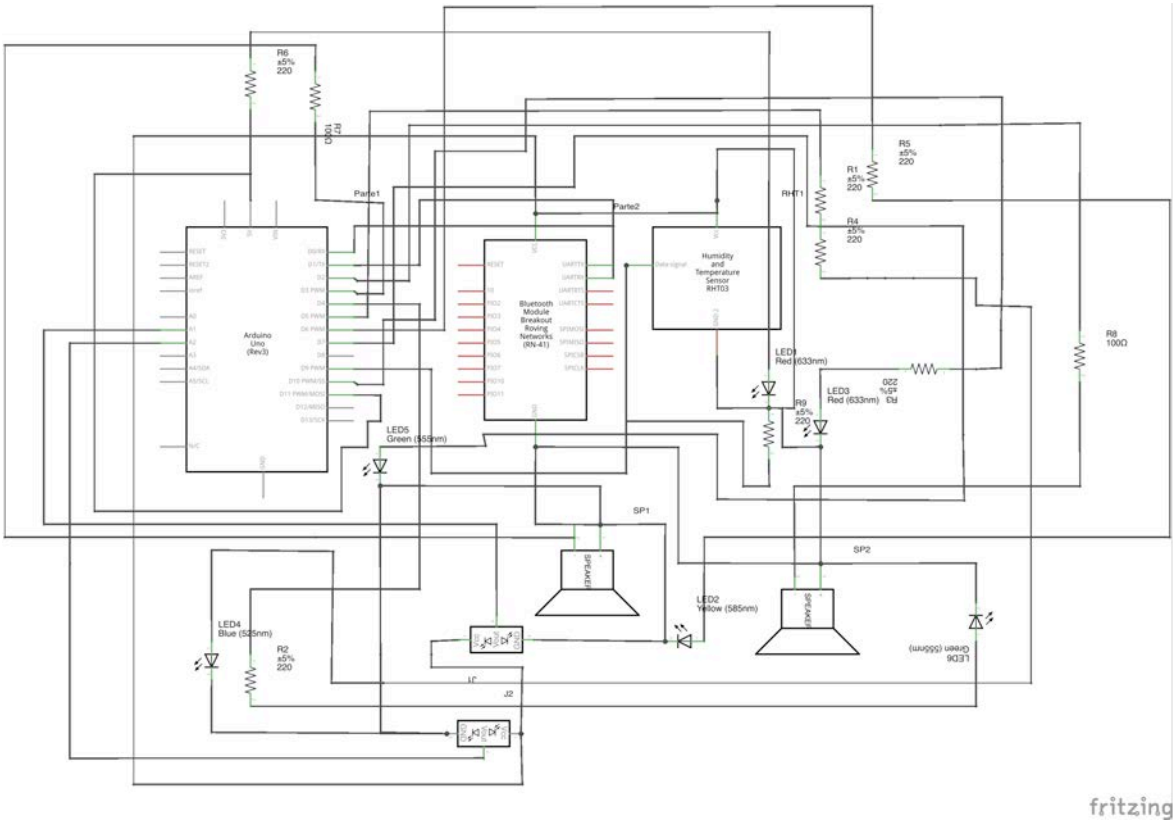


Ilustración 44. Esquema electrónico del proyecto

De la figura 42 a la 46 se muestran las fotografías correspondientes a los prototipos físicos, una vez terminada la fase de fabricación.

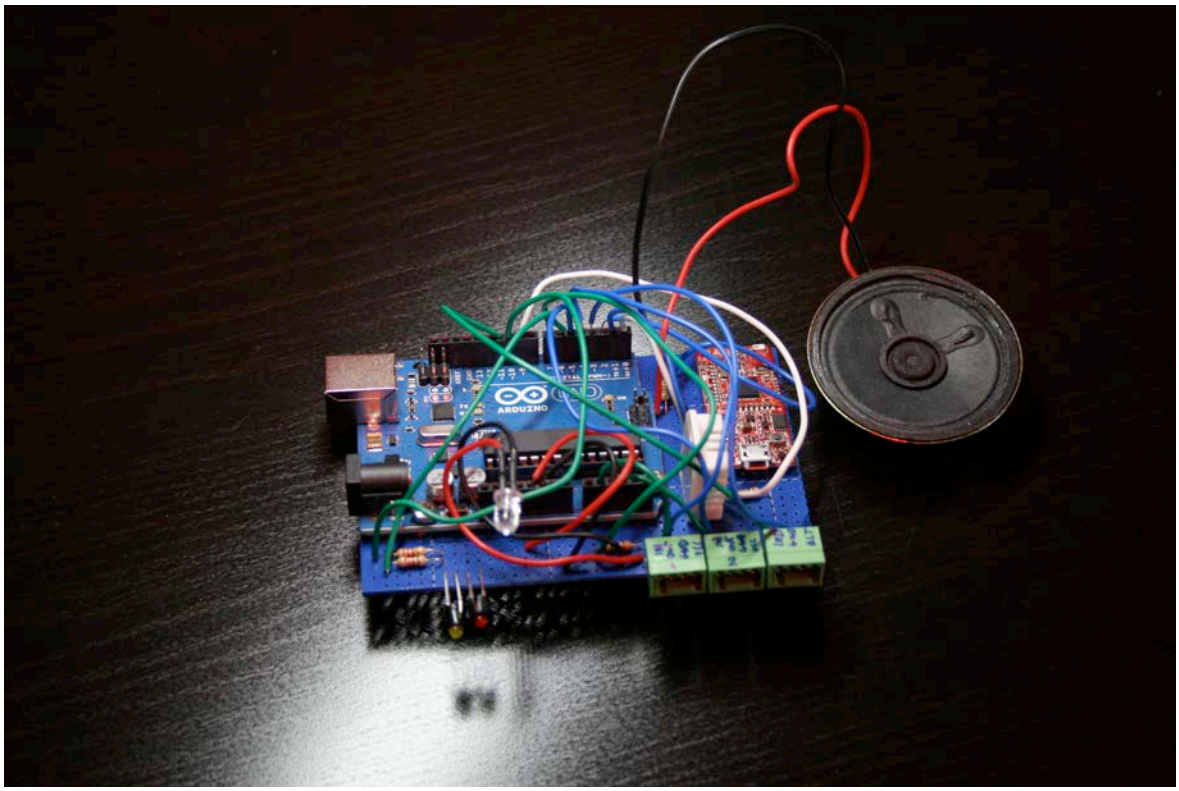


Ilustración 45. Electrónica caja principal comunicaciones

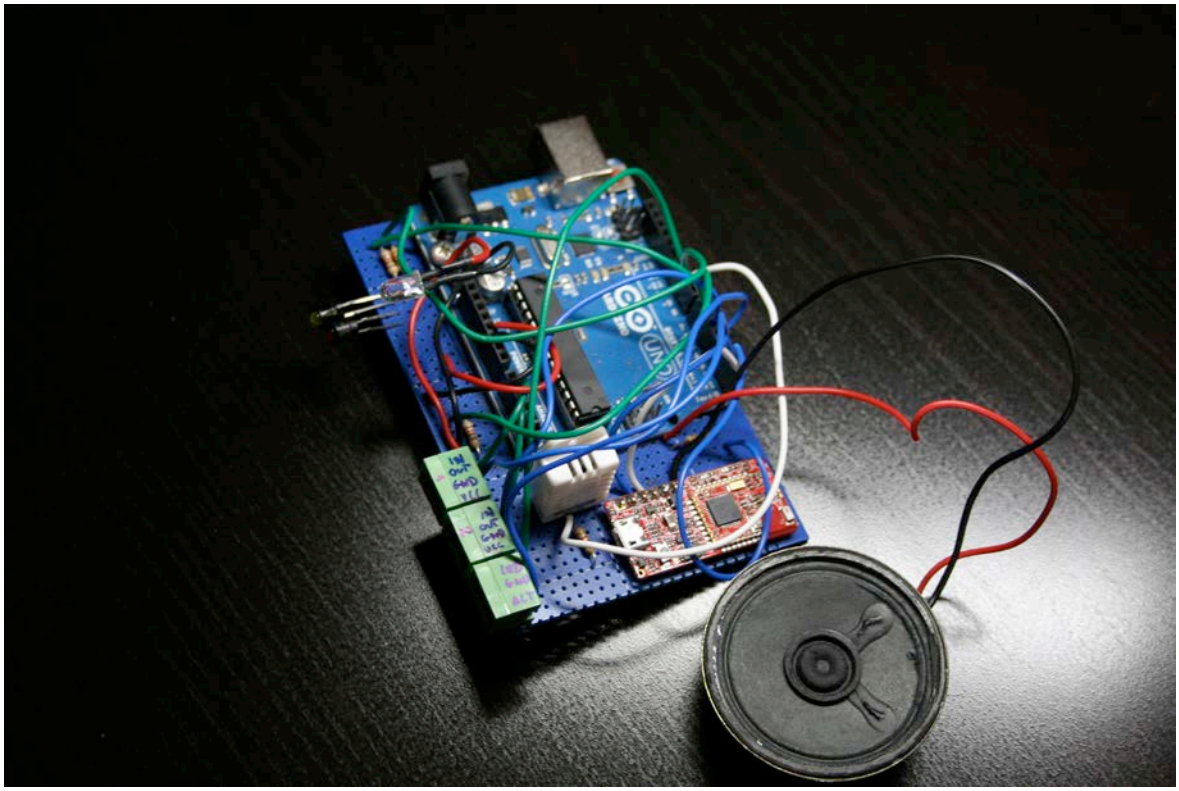


Ilustración 46. Electrónica caja principal comunicaciones. Vista alternativa.

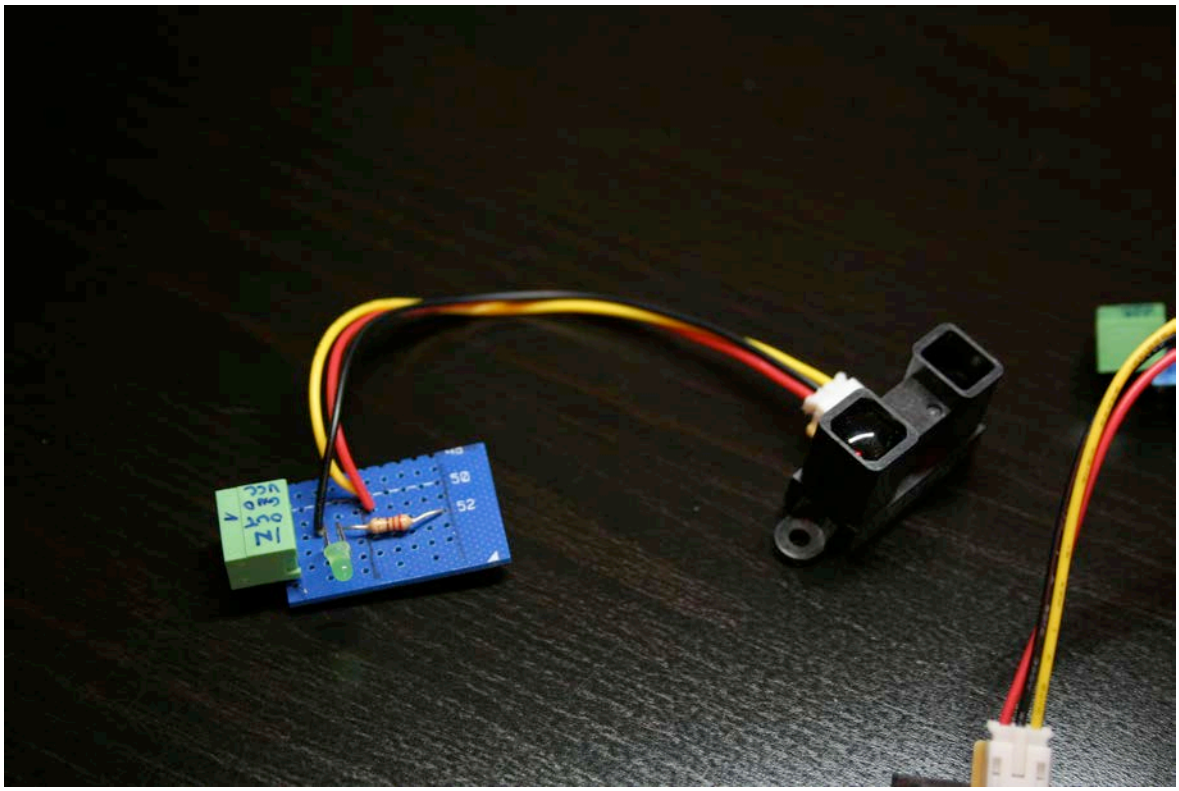


Ilustración 47. Electrónica de una de las torres de detección.

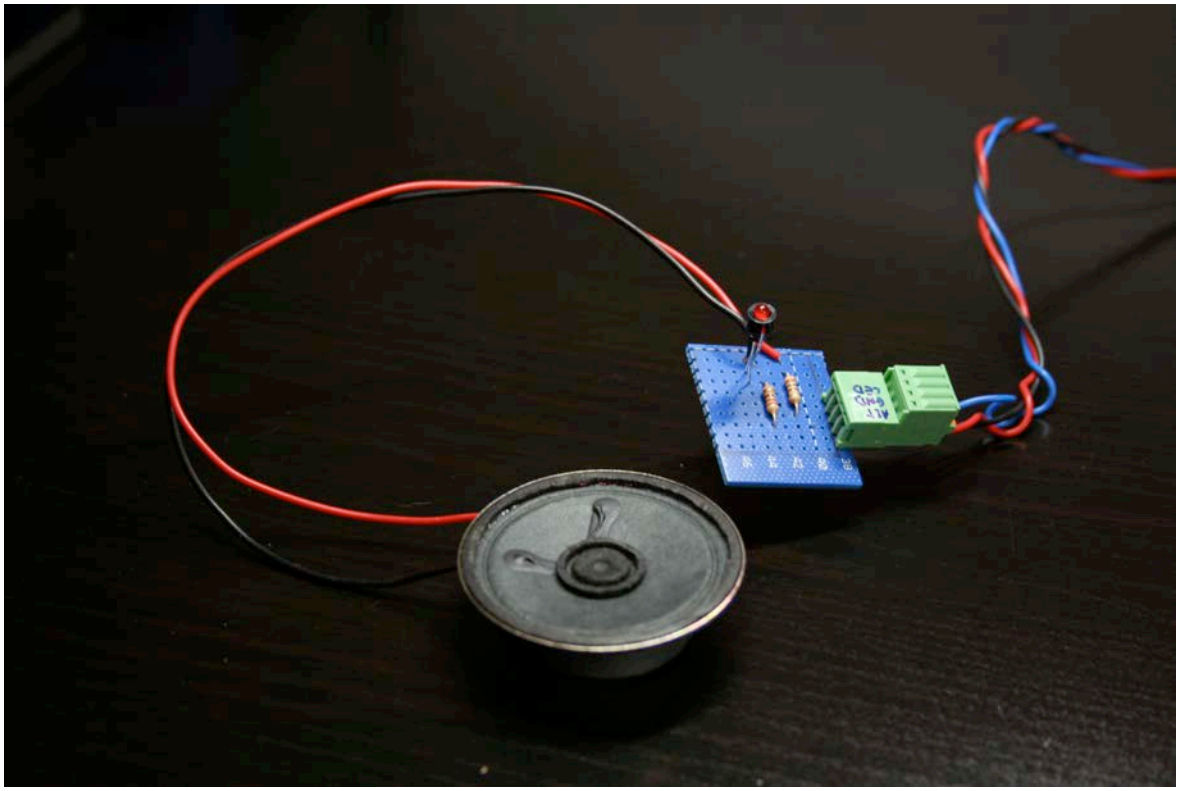


Ilustración 48. Electrónica de la caja auxiliar de señalización.

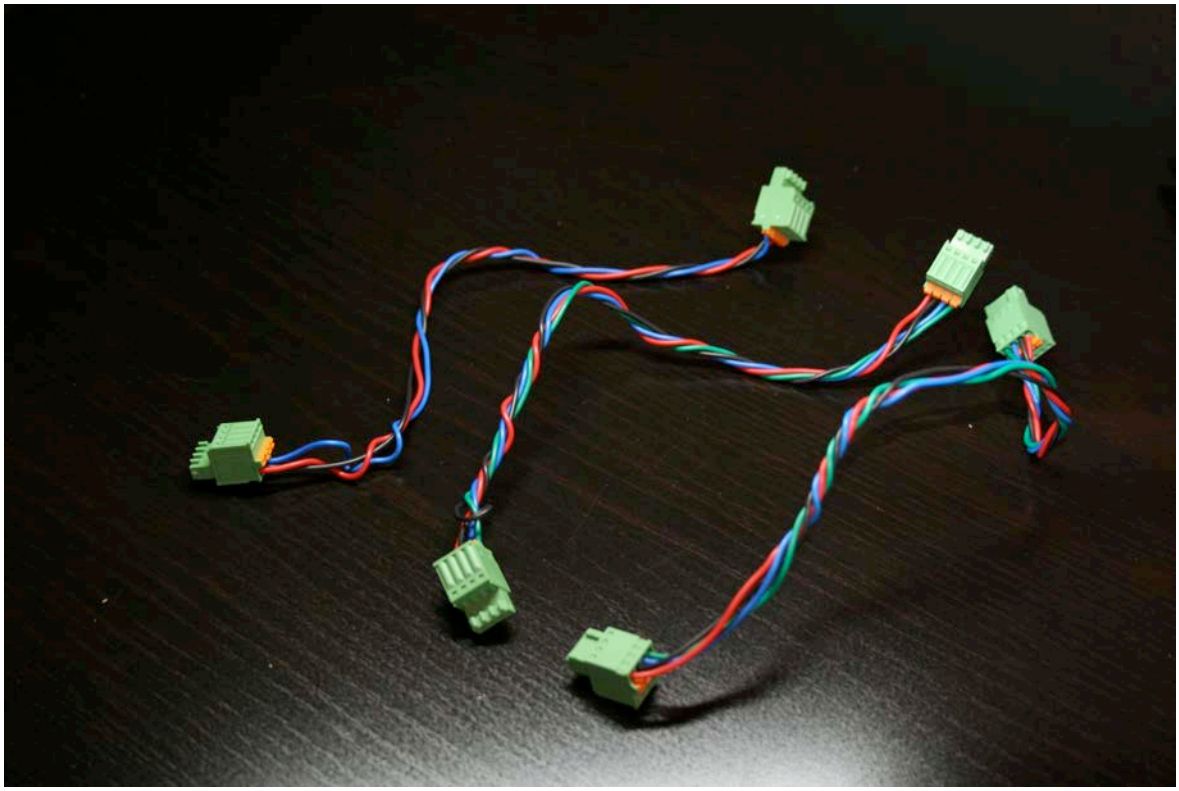


Ilustración 49. Cableado de interconexión de cajas y torres.

7.3 Algorítmica relativa al firmware del micro-controlador

A continuación, y una vez fabricados los prototipos electrónicos, se procederá al desarrollo del firmware de dicha electrónica.

Se realizará un desarrollo basado en los siguientes algoritmos:

- Testeo inicial de componentes
- Detección de vehículos
- Cálculo de velocidad de vehículos
- Cálculo de conflicto peatón-vehículo
- Detección de placas de hielo
- Monitorización de alarmas acústico-visuales
- Protocolo de comunicaciones.

A continuación se procede a la descripción y explicación de cada uno de estos algoritmos implementados.

7.3.1 Testeo de componentes

En la fase de setup del micro se realizará un proceso de testeo de cada uno de los componentes embebidos. Dicho testeo se utilizará como diagnosis, para identificar si hay un fallo en el hardware del sistema.

Este proceso activará durante un tiempo determinado cada uno de los elementos, de tal manera que en caso de malfunción del sistema en esta primera fase de setup, se identifique cualquier fallo en el hardware.

Dicho testeo se realizará con los componentes visuales y acústicos.

A continuación se muestra la parte de código correspondiente a este algoritmo de testeo en la fase de setup del micro.

```
  //En la fase de setup del micro se realiza un
encendido/apagado de los
  //Perifericos con finalidad de test
  digitalWrite(ledTorreIr1,HIGH);
  digitalWrite(ledTorreIr2,HIGH);
  digitalWrite(ledAmarilloCom,HIGH);
  digitalWrite(ledRojoCom,HIGH);
  digitalWrite(ledRojoAux,HIGH);
  digitalWrite(ledOn,HIGH);
  tone(3, NOTE_C4,1000/2);
  delay(1000);
  tone(2, NOTE_C4,1000/2);
  delay(2000);
```

```
digitalWrite(ledTorreIr1,LOW);
digitalWrite(ledTorreIr2,LOW);
digitalWrite(ledAmarilloCom,LOW);
digitalWrite(ledRojoCom,LOW);
digitalWrite(ledRojoAux,LOW);
```

Como se puede observar, se escriben valores HIGH (1s lógicos en primera instancia) en los LEDs del sistema y se emiten sonidos en ambos altavoces (empleando la instrucción tone de la librería pitches.h comentada en el punto 7.1.2), durante 2000ms (instrucción delay) los LEDs y 500ms los altavoces.

Una vez transcurrido ese tiempo, se deja de emitir sonido en los altavoces sin necesidad de emplear ninguna instrucción, y se desactivan los LEDs mediante la escritura de valores LOW (0s lógicos) en los pines correspondientes.

7.3.2 Detección de vehículos

Para la detección de vehículos se utiliza un algoritmo basado en la lectura analógica de los sensores IR.

Por criterios de diseño, se ha establecido una distancia máxima de las balizas respecto al paso de vehículos de 40cm.

En primera instancia se realiza una conversión del valor analógico devuelto por el micro de la medición de cada uno de los sensores. Dicha conversión devuelve un valor en cm.

Una vez realizada la conversión se considera detección de paso de un vehículo cuando el retorno de dicha conversión es menor o igual que esos 40cm.

El código asociado a este algoritmo se muestra en las siguientes líneas:

```
time1=analogRead(Ir1);
time2=analogRead(Ir2);
cm = 10650.08 * pow(time1,-0.935) - 10;
cm2 = 10650.08 * pow(time2,-0.935) - 10;

if(cm<40)
{
  digitalWrite(ledTorreIr1,HIGH);

  tone(2, NOTE_C6,1000/2);
  flag1=HIGH;
  tiempo=tiempoTranscurrido();
}

else{
  digitalWrite(ledTorreIr1,LOW);
  digitalWrite(ledTorreIr2,LOW);
```

```
}
```

El cálculo de dicha distancia se realiza en base a la gráfica de variación de tensión del sensor con la distancia al objeto proporcionada por el fabricante

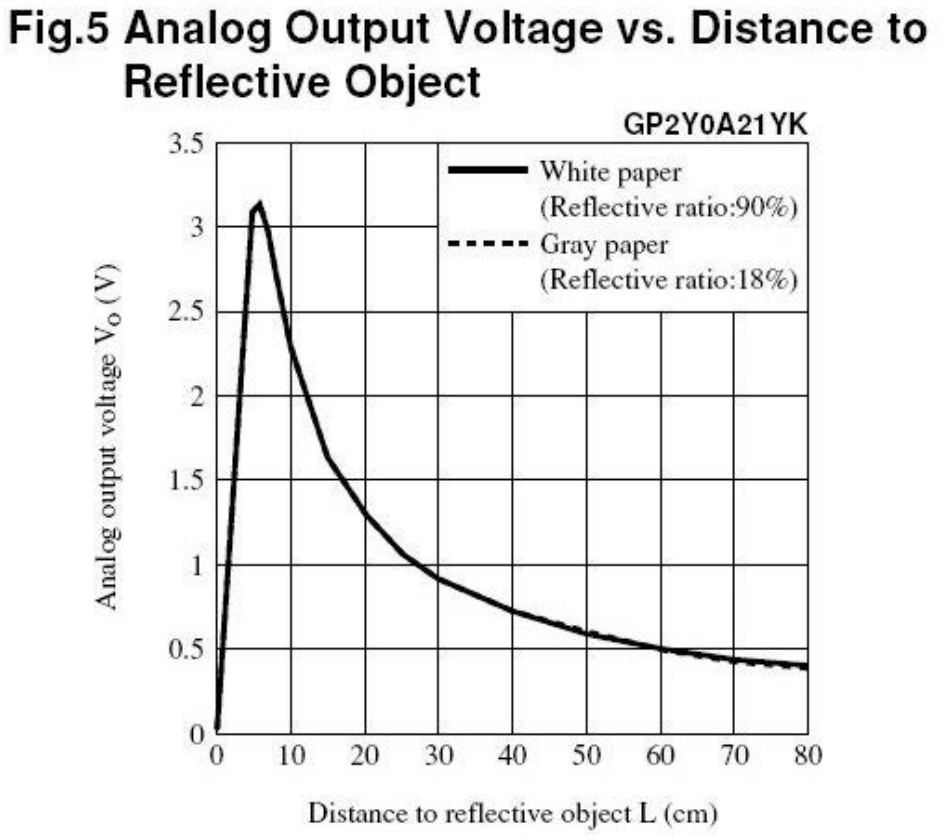


Ilustración 50. Variación de tensión con respecto a la distancia de detección del sensor GP2Y0A21YK

Como se observa en esta gráfica, la detección es efectiva a partir de aproximadamente 6cm de distancia, donde la curva de tensión comienza a caer a razón de la función $V=L^{-0,935}$

El resto de operaciones en esa misma línea, realizan la conversión de valores analógicos de 8bits a un valor en coma flotante devuelto en cm.

En las primeras 4 líneas se realiza el cálculo de la distancia de detección. A continuación, si esta distancia es menor que la especificada, se considera detección y se procede al cálculo de la velocidad del vehículo.

Además, como se observa en el código, la detección de un vehículo implica el encendido del LED de la torre desde la que se realiza la detección y la emisión de un tono del altavoz.

7.3.3 Cálculo de la velocidad de vehículos

Para calcular la velocidad, partimos de la detección de vehículo realizada.

El algoritmo se basa en una primera detección realizada en la primera torre de detección, en el conocimiento de la distancia existente entre ambas torres, y en la estampación de los instantes de tiempo en la que se realizan ambas detecciones.

A continuación se muestra la parte de código del firmware en la que se implementa el algoritmo de cálculo de velocidad:

```
time1=analogRead(Ir1);
time2=analogRead(Ir2);
cm = 10650.08 * pow(time1,-0.935) - 10;
cm2 = 10650.08 * pow(time2,-0.935) - 10;

if(cm<40)
{
  digitalWrite(ledTorreIr1,HIGH);

  tone(2, NOTE_C6,1000/2);
  flag1=HIGH;
  tiempo=tiempoTranscurrido();
}

else{
  digitalWrite(ledTorreIr1,LOW);
  digitalWrite(ledTorreIr2,LOW);
}

if (flag1==HIGH){
  if (flag2==HIGH){

//Suponemos que hay unos 10m entre los dos sensores
velocidad=10/(tiempoCronometrado/1000.00);
velocidadKmh=velocidad/1000*3600;
```

Como se puede observar, con la detección de un vehículo en la primera torre, se activa una bandera denominada flag1.

Esta bandera se emplea para marcar el inicio de paso de un vehículo del cual se va a realizar el cálculo de velocidad.

Una vez levantada esa bandera se ejecuta la función tiempoTranscurrido:

```
//*************************************************************
************************//
// TiempoTranscurrido()
// Variables de entrada: ninguna
// Salida: La funcion devuelve el tiempo transcurrido entre d
os eventos en cada
//         Uno de los sensores IR
//*************************************************************
************************//
unsigned long tiempoTranscurrido()
{
  t_anterior=millis();
  while (10650.08 * pow(analogRead(Ir2),-0.935) - 10>40);
//Espero hasta que cambie a HIGH
  tiempoCronometrado=millis()-t_anterior;
  flag2=HIGH;
  digitalWrite(ledTorreIr2,HIGH);
}
```

Al ejecutarse esta instrucción, se almacena el tiempo en una primera instancia, que corresponderá con la captura de tiempo de la primera detección.

Tras esta captura se ejecuta un bucle en espera de obtener detección en la siguiente torre, y se rastrea el tiempo transcurrido desde el primer timestamp hasta el instante actual.

Una vez se ha realizado la detección también la segunda torre, se sale del bucle y se activa una segunda bandera (flag2) a la vez que se enciende el LED de la segunda torre indicando la detección.

Al salirse la ejecución del programa del bucle se queda almacenado en la variable tiempoCronometrado el timeStamp correspondiente a la segunda detección.

Continuando en la primera parte del código (al salir de la función tras la detección final) el programa se encuentra con ambos flags (flag1 y flag2 levantados, por lo que se ejecuta el contenido integrado en estas dos condiciones, que se registra en la variable velocidad, la velocidad del vehículo analizado.

Para este cálculo es necesario conocer la distancia entre las torres de detección. En este caso particular se ha establecido una distancia de 10m.

A continuación se bajan las dos banderas para preparar el sistema para una nueva detección.

7.3.4 Cálculo de conflicto peatón-vehículo

Para el cálculo de conflicto peatón-vehículo es necesario conocer el resultado previo de los algoritmos de detección de velocidad, la distancia de las torres de detección respecto del paso de peatones en cuestión y una estimación del tiempo necesario para realizar el cruce holgadamente sin posibilidad de conflicto.

```
  //Suponemos que entre el poste IR2 y el paso de peatones
hay 150m y que el peatón tarda 5 segundos en cruzar, pero
dejamos otros 5 segundos de margen
  //Por tanto suponemos un intervalo de 10segundos de margen
para poder dar vía libre al peatón para cruzar
  tiempoRestante=150.00/velocidad;

  delay(100);
  flag1=LOW;
  flag2=LOW;

  //Si al pasar un coche, el tiempo calculado de llegada al
paso de peatones es inferior
  //al calculado de tiempo de paso, se activara una alarma en
funcion del estado del peaton (en espera/Circulando)
  if (tiempoRestante<10.00){
    digitalWrite(ledRojoCom, HIGH);
```

En la variable tiempoRestante se calcula el tiempo que queda tras el paso de un vehículo por las torres de detección a una velocidad conocida, hasta que alcance el paso de peatones.

Es necesario, como se apuntaba anteriormente conocer la distancia entre la última torre y el paso de peatones, que en este caso en concreto es de 150m.

Si el tiempo restante es menor de la estimación temporal de paso del cruce en cuestión (en este caso concreto se ha estimado un tiempo de cruce medio de 10 segundos), se realizarán las acciones pertinentes definidas del proceso de alarmas por posible conflicto vehículo-peatón.

7.3.5 Monitorización de temperatura y humedad

Gracias a la librería DHT.h el procedimiento para la monitorización y seguimiento de temperatura y humedad es extremadamente sencillo.

Este procedimiento se realiza en las siguientes líneas:

```
humedad=dht.readHumidity();
temperatura=dht.readTemperature();
  if (isnan(temperatura) || isnan(humedad)) {
```

```
        Serial.println("Failed to read from DHT");
    }
```

En las variables temperatura y humedad se almacenan los parámetros leídos del sensor mediante los comandos readHumidity() y readTemperature.

Si los datos devueltos por alguna de las dos instrucciones no corresponde con un formato de datos numérico, el firmware devolverá un error.

7.3.6 Detección de placas de hielo

Para la detección de placas de hielo el algoritmo es extremadamente sencillo, y se basa en el cálculo de los parámetros de temperatura y humedad descritas con anterioridad.

Ésta detección se realiza en las siguientes líneas:

```
if (temperatura < 2.0){
 if (humedad > 60){
```

Es decir, si la temperatura detectada es menor de dos grados y la humedad relativa es mayor del 60%, se considera que empieza a haber riesgo de placas de hielo y se activará el procedimiento de alarma definido a tal efecto.

7.3.7 Protocolo de comunicaciones

Para la comunicación entre la baliza y el dispositivo se ha diseñado un protocolo específico para este sistema, basado en tramas de 24 bits.

Estas tramas poseen una cabecera de 8 bits y una división de 2 campos, uno de instrucción y otro de parámetros de 8 bits cada uno.

Nº de bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Cabecera								Campo de instrucción								Campo de parámetros							

Ilustración 51. Formato de trama de protocolo de comunicación

La cabecera del protocolo puede tomar dos valores en hexadecimal:

- 0x0A:
  Esta cabecera indica que los datos del campo de instrucción se corresponden con instrucciones de alerta o información prioritaria que el dispositivo debe procesar, interpretar y comunicar de manera inmediata.

  Estas instrucciones son unívocas y no llevan parámetros asociados, por lo que el campo de parámetros estará siempre a 0.

  Están codificadas de la siguiente manera:

  - 0x01: Señal o alarma de paso permitido en el cruce. El micro ha realizado todos los procesos de detección y no existe conflicto peatón-vehículo.
  - 0x02: Señal o alarma de posibilidad de pavimento con placas de hielo. El micro ha detectado parámetros ambientales que posibilitan la existencia de placas de hielo en el asfalto.
  - 0x03: Señal o alarma de peligro, el micro ha realizado la detección de vehículo circulando a velocidad propicia para conflicto peatón-vehículo en la fase de reconocimiento previa al cruce.
  - 0x04: Señal de inicio de fase de reconocimiento. El micro está realizando un reconocimiento previo para detectar posibles vehículos en la zona de detección.
  - 0x05: Señal de final de cruce y desconexión del sistema. Ha transcurrido el tiempo de cruce o *timeout.* Se realiza la desconexión del módulo con el dispositivo.

- 0x0B:
  Esta nueva cabecera indica una instrucción con paso de parámetros. En este caso, el campo de parámetros se utiliza para pasar valores de variables del micro al software del dispositivo.

  En este proyecto se utiliza sólo una instrucción de este tipo, codificada de la siguiente manera:

  - Campo de instrucción 0x00: Indica que, una vez se ha emitido la señal de paso libre y por tanto la orden de cruce al usuario, se ha detectado un vehículo que propicia el conflicto peatón-vehículo.

    El campo de parámetro se emplea para pasar el número de vehículos detectados en esta circunstancia.

En el firmware, el protocolo se implementa de la siguiente manera:

```
if(flagEscritura==HIGH)
{
```

```
 //PASO PERMITIDO
if (flagsBLE== 1)
{
  BLEMini_write(0x0A);
  BLEMini_write(0x01);
  BLEMini_write(0x00);
  flagEscritura=LOW;
  //Serial.println("Escritura ");
}
//PASO PERMITIDO CON PRECAUCION POR PLACAS DE HIELO
else if (flagsBLE==2)
{
  BLEMini_write(0x0A);
  BLEMini_write(0x02);
  BLEMini_write(0x00);
  flagEscritura=LOW;
  //Serial.println("Escritura ");
}

//PASO PERMITIDO AGILIZADO POR COCHE APROXIMANDOSE
else if(flagsBLE==3)
{
  BLEMini_write(0x0B);
  BLEMini_write(0x00);
  BLEMini_write(value);
  flagEscritura=LOW;
  //Serial.println("Escritura ");
}

//COCHE APROXIMANDOSE, PERMANEZCA A LA ESPERA
else if (flagsBLE==4)
{
   BLEMini_write(0x0A);
   BLEMini_write(0x03);
   BLEMini_write(0x00);
   flagEscritura=LOW;
   //Serial.println("Escritura ");
}

//DESCONEXION
else if (flagsBLE==5)
{
  BLEMini_write(0x0A);
  BLEMini_write(0x05);
  BLEMini_write(0x00);
  flagEscritura=LOW;
  //Serial.println("Escritura ");
}

else if (flagsBLE==6)
{
  BLEMini_write(0x0A);
  BLEMini_write(0x04);
```

```
    BLEMini_write(0x00);
    flagEscritura=LOW;
  }

}
```

7.3.8 Monitorización de alarmas acústico-visuales

Los algoritmos de monitorización de alarmas acústico-visuales conforman los más complejos del sistema.

Esto es debido a que es la parte más crítica, ya que, por el perfil de usuario al que va dirigida la aplicación, es fundamental señalizar correctamente mediante el audio disponible cada información procesada en el micro.

Todos los sub-sistemas de detección tienen que estar perfectamente sincronizados y los tiempos de ejecución perfectamente definidos y parametrizados.

El sistema se ha diseñado sobre una base de tiempos y eventos perfectamente coordinados para una funcionalidad práctica de la aplicación, sobre la que se ha configurado una máquina de estados basada en estos parámetros tiempo-evento.

En función de el estado actual de la máquina de estados se procesan unas alarmas determinadas a través de las balizas, el dispositivo o ambos.

A continuación se muestra la máquina de estados utilizada, para simplificar el gráfico, se codificarán las variables dependientes de la entrada de parámetros a cada estado de la máquina:

La entrada será del tipo:

nºbit	0	1	2	3	4
Nombre	Temporizador	ConexiónBLE	PasoPermitido	PasoPermitido Agilizado	AlarmaClima
Tipo de dato	Entero	Booleano	Booleano	Booleano	Booleano

El dato Temporizador tomará en la máquina 3 estados posibles en función de su valor:

- 0: El contador está entre 0 y 5s.
- 1: El contador está entre 5 y 15s.
- 2: El contador es mayor de 15s.

El parámetro Conexión BLE toma el valor 1 cuando se ha establecido conexión con un dispositivo BLE.

El resto son banderas que se activan/desactivan en función de diferentes procesos del firmware.

De esta manera, por ejemplo, una entrada con valor 11100 significa:

1- Temporizador entre 5 y 15s.
1- Conexión BLE activa
1- Bandera PasoPermitido levantada
0- Bandera PasoPermitido agilizado bajada
0- Bandera Alarma clima bajada.

Las salidas corresponden con valores de la variable flagsBLE que indican comandos del protocolo de comunicación, y en algún caso de modificación del temporizador:

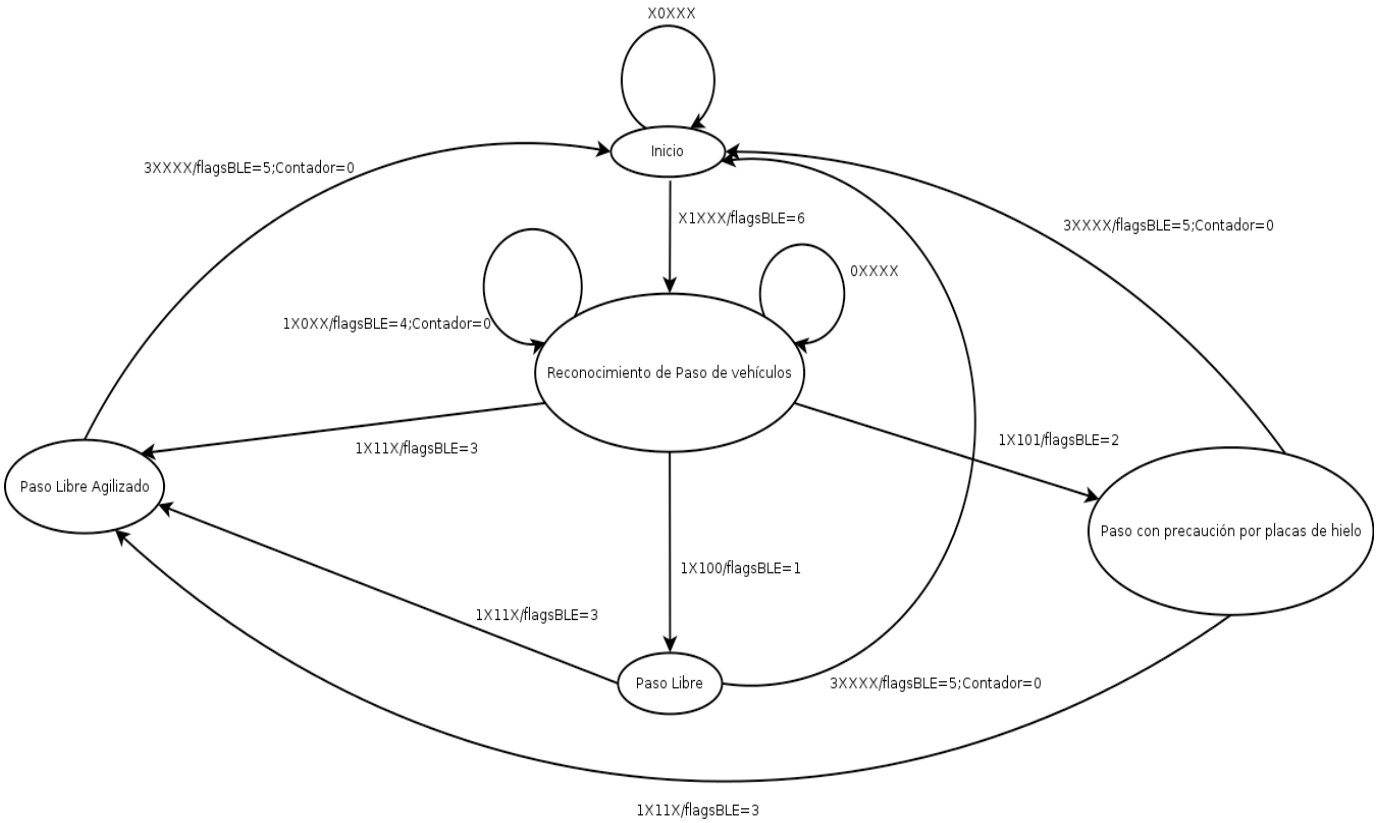


Ilustración 52. Diagrama máquina de estados del sistema

Se puede observar que ciertas entradas no se contemplan en muchos estados.

En la gran mayoría de los casos, es por cuestiones de seguridad.

Por ejemplo, en el estado de paso libre, no se contempla que se desactive la conexión Bluetooth, esto es debido a que puede haber habido algún fallo en el dispositivo del usuario, y por seguridad, se termina el proceso del cruce sin cortar la conexión, hasta prever la finalización del mismo.

Otro ejemplo es que una vez se activa el estado de paso con precaución por placas de hielo, si en el transcurso del cruce se desactiva la bandera de alarma climática, el sistema no desactiva dicha alarma, ya que si existían placas de hielo, estas seguirán permaneciendo, o puede deberse a un fallo del sensor.

Además de la señalización acústica relativa a nuevos vehículos detectados comentada en puntos anteriores, con la bandera PasoLibre levantada se ejecuta una serie de tonos predeterminados a través de los altavoces.

Con la salida de flagsBLE=4 perteneciente a la desconexión del sistema se ejecuta una secuencia de tonos característica para indicar el fin del cruce, mediante el siguiente código:

```
void SpeakersFin()
{

  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // Para calcular la duración de cada nota, se toma un
//segundo dividido entre la fracción de negra
    //p.ej.: cuarto de negra = 1000 / 4, octavo de negra =
1000/8, etc.
    int noteDuration = 1000/noteDurations[thisNote];
    tone(2, melody[thisNote],noteDuration);

    // Para distinguir las notas, se establece un minimo
entre ellas.
    // Una duración de + 30% funciona correctamente:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(3);
  }
}
```

7.4 Software de la aplicación de usuario

Una vez disponibles todos los elementos relativos al desarrollo de las balizas, se procederá, según lo establecido en la planificación, al desarrollo de la aplicación de usuario, a la cual se ha llamado “Crosswalk Asistant” (Asistente de cruce).

Como se apuntaba en apartados anteriores, es necesario tener en cuenta que el proceso de sincronización con las rutinas cronológicas de la aplicación, son una parte crítica para la eficacia funcional del proyecto.

Ilustración 53. Icono de la aplicación de usuario.

Además, los procesos de monitorización e interacción con el usuario, en general soportan la mayor carga de responsabilidad en cuanto a requisitos de calidad para construir un proyecto no sólo viable, sino útil y práctico.

Por tanto, la gestión de la interacción usuario-dispositivo implementada en la aplicación de usuario jugará un papel esencial en el sistema.

A continuación se describirán los detalles de las diferentes fases de desarrollo llevadas a cabo, desde el diseño de la estructura de la aplicación, hasta el código fuente final.

7.4.1 Estructura de la aplicación

La aplicación estará estructurada en dos grandes bloques que desempeñarán tareas distintas, pero encadenadas.

Estas dos tareas fundamentales son:

- Identificación posicional de la baliza en un mapa geográfico y rastreo de la localización de usuario.

- Comunicación con la baliza y monitorización de instrucciones recibidas.

En la primera tarea, se visualizará un mapa geográfico a través del framework CoreLocation, el cual tendrá almacenadas las coordenadas de la posición de las balizas situadas en los pasos de cebra sin semáforos determinados.

Paralelamente se irá realizando el rastreo a tiempo real de la ubicación del dispositivo.

La aplicación, asignará una región circular determinada a la posición de las balizas, de tal manera que, una vez la localización registrada del dispositivo mediante la tarea de rastreo se encuentre dentro de esa región, la aplicación salte a la siguiente tarea.

De esta manera, cuando el usuario se encuentre en las proximidades físicas de la baliza con su dispositivo móvil, automáticamente aparecerán en la pantalla del mismo los elementos que se definen a continuación para la interacción con dicha baliza.

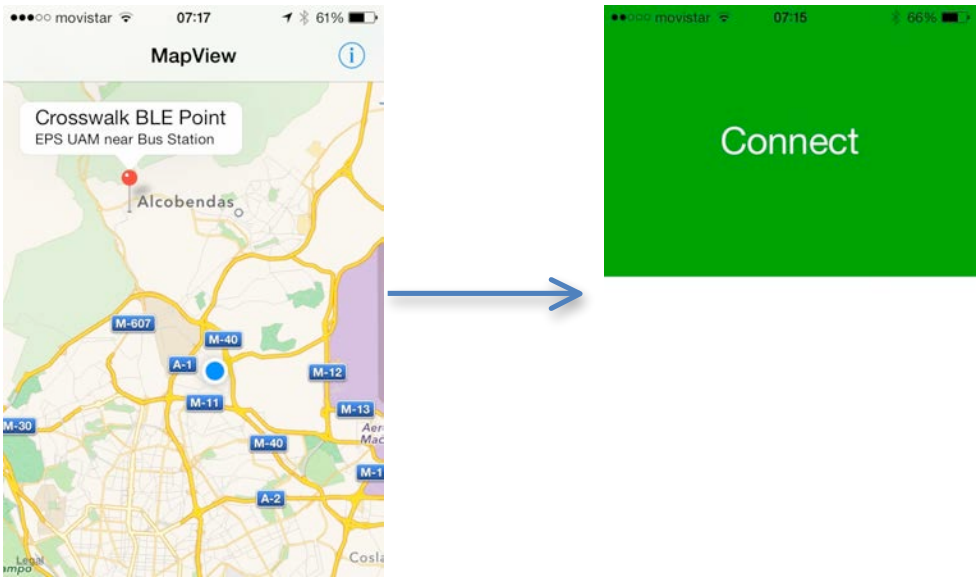


Ilustración 54. Conexión de las dos funciones principales de la aplicación.

La segunda tarea principal, consiste en la interacción con la baliza.

En este marco de tarea, se supone en primera instancia que la baliza se encontrará en una región geográfica dentro de los límites del rango de acción de la comunicación, establecida por el fabricante del módulo BLE.

Por tanto, la primera acción significativa es la confirmación de la conexión dispositivo-módulo BLE.

Una vez se ha realizado la conexión, la aplicación se limita a recibir comandos de la baliza y transmitirlos al usuario de manera eficiente, teniendo en cuenta el perfil de usuario al que va dirigido la aplicación.

Una vez se realiza la desconexión con la baliza, se retorna a la vista de la tarea número 1, y se procede al rastreo de posibles nuevas balizas cercanas.

7.4.2 Diseño de interacción con el usuario

Sin duda, la parte de interacción con el usuario, como se comentaba en puntos anteriores presenta un punto crítico en el desarrollo de la aplicación.

Se necesita la interacción directa de la manera más sencilla posible del usuario con la pantalla táctil, para la introducción de comandos que permitan al dispositivo conocer cuál es la voluntad del usuario en cada momento.

Esta tarea, en un Smartphone que posee una pantalla de 3,5 pulgadas, puede resultar tediosa para usuarios con dificultades visuales.

Por este motivo, se ha querido reducir al máximo el número de interacciones necesarias entre el usuario y el dispositivo a través de gestos táctiles.

En las acciones en las que no queda otra posibilidad se ha pretendido optimizar dicha interacción, con gestos extremadamente sencillos, información al usuario mediante comandos acústicos la región de la pantalla donde debe realizar la interacción y aumentando la proporción de la región donde debe interactuar en la medida de lo posible.

Se ha procurado utilizar altos contrastes y tamaños de letra grandes en las pantallas que necesitan interacción con el usuario o que proporcionan información adicional.

Después de este análisis de diseño, se ha desarrollado la interfaz de interacción bajo las siguientes premisas:

- Durante el transcurso de la interacción con las balizas, el usuario es informado en todo momento de lo que sucede en su entorno, y cómo debe interactuar con el dispositivo para que sucedan cosas en el sistema.

- Se ha reducido el número de interacciones gestuales con el dispositivo a una, consistente en la pulsación de un botón para la confirmación de la conexión con las balizas.

- El elemento botón, constituye el 50% de la pantalla táctil, que supone toda la mitad superior de la misma. Además presenta un verde oscuro intenso con letras en un formato de texto suficientemente grande en color blanco.

- La mitad inferior, presenta un fondo blanco, con letras de igual tamaño que las mencionadas del botón. Esta mitad inferior se utiliza para monitorizar con texto escrito los comandos recibidos de la baliza. Se utiliza tanto para usuarios con capacidades visuales menos limitadas, como para diagnóstico.

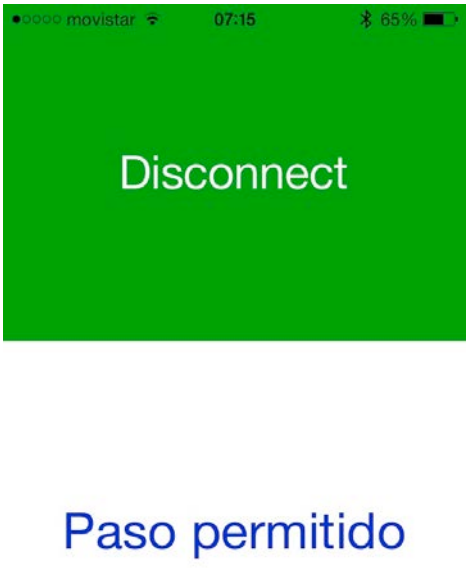


Ilustración 55. Captura de una de las pantallas del proceso de interconexión con la baliza

7.4.3 Rastreo posicional y geolocalización de usuario y balizas

En primera instancia, se ha desarrollado una mini-aplicación para el rastreo posicional del dispositivo del usuario, el almacenamiento de las coordenadas geográficas de las balizas y la identificación de el traspaso de los límites del rango de acción de cada baliza.

Esta mini-aplicación está basada en los frameworks MapKit y CoreLocation de XCode.

En primera instancia, se realiza la visualización de la parte de mapas geográficos que contiene la ubicación de usuario con la función showUserLocation.

Posteriormente, se almacenan los datos de dicha localización de usuario, en dos variables de longitud y latitud, mediante userLocation.coordinate.latitude y userLocation.coordinate.longitude.

Se almacenan en un tipo estructurado de datos tipo coordenada 2D (CLLocationCoordinate 2D).

Se calcula la proximidad de estas coordenadas con respecto a las coordenadas de las balizas.

Éstas han sido previamente almacenadas en variables estáticas o en memoria del dispositivo.

A continuación, se define una región de alcance de la comunicación de la baliza. En el caso de este proyecto, se ha definido un rango de acción de 200m.

Si en una de las actualizaciones posicionales de la ubicación de usuario, se detecta que las coordenadas se encuentran dentro de los límites establecidos, se ejecuta el segue definido, que en el caso del sistema que se describe en este documento, corresponde con el inicio de la conexión y comunicación con las balizas.

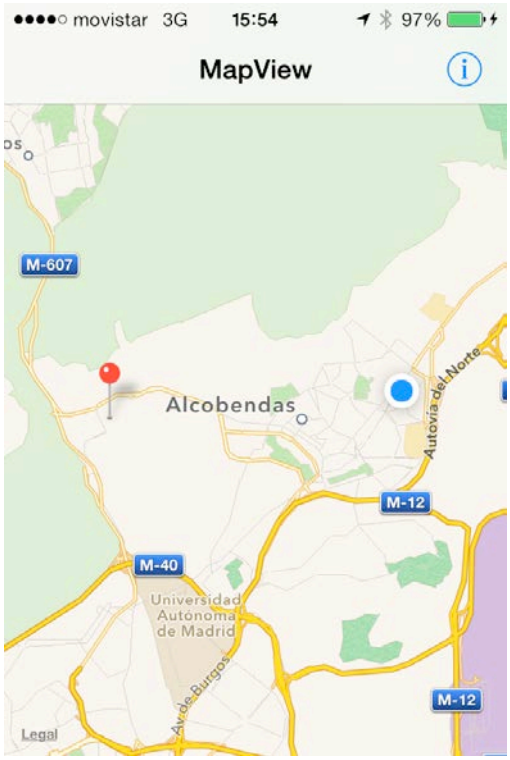
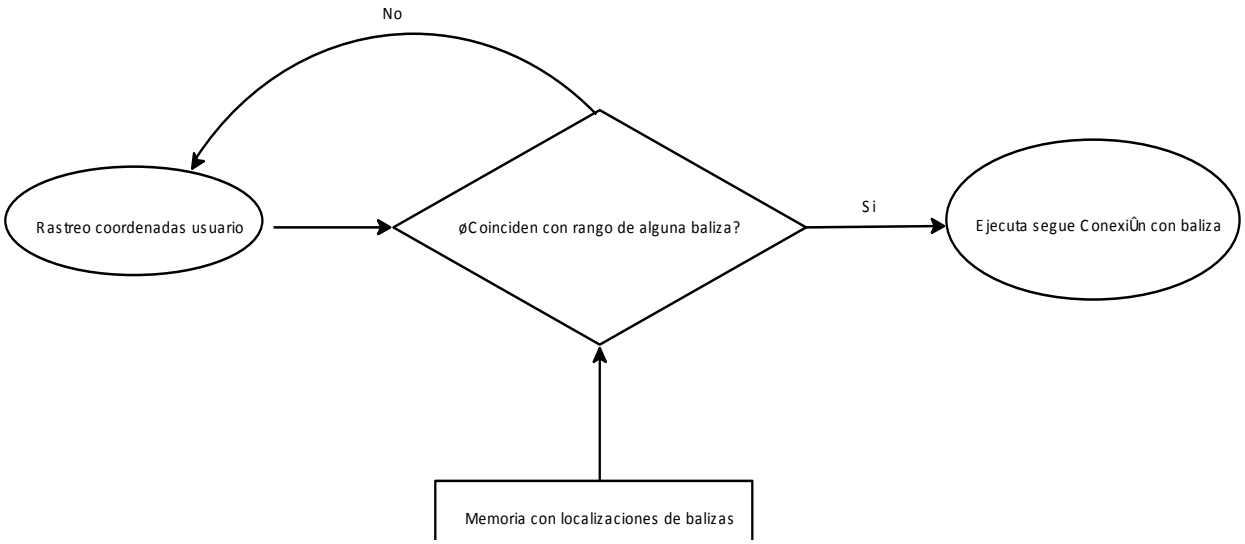


Ilustración 56. Detalle de la mini-aplicación de localización

De manera adicional, en el mapa se muestran de forma permanente las ubicaciones de todas las balizas.

El diagrama de flujo de esta mini-aplicación sería el siguiente:



Una vez realizada esta mini-aplicación, se debe integrar con la aplicación completa, ejecutando el segue correctamente con el resto de la funcionalidad de la aplicación

7.4.4 Conexión/Desconexión e intercomunicación dispositivo móvil – baliza

Una vez el sistema detecta que el dispositivo se encuentra en el radio de acción de una de las balizas, éste ofrecerá la posibilidad al usuario de intentar establecer la conexión con dicha baliza.

Por lo que mostrará la siguiente pantalla:

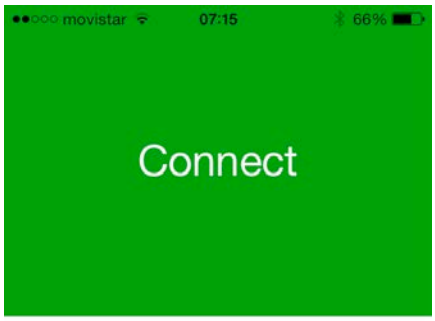


Ilustración 57. Pantalla inicio comunicación con baliza.

En la que además se informará con un comando de voz pregrabado en el dispositivo y la función AudioServicesPlaySystemSound(); del framework AudioToolBox de la posición del botón y la acción que se realiza al pulsarlo.

Al presionar dicho botón, se procede al intento de conexión, en el que se mostrará la siguiente pantalla:

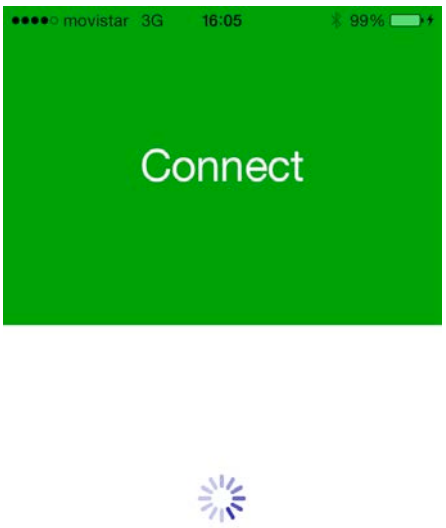


Ilustración 58. Pantalla de intento de conexión de la aplicación.

Si la conexión no se realiza, vuelve a aparecer el título connect en el botón, hasta que se cumple un tiempo de timeout.

Si por el contrario, se realiza la conexión con la baliza, ésta se activa y comienza el procedimiento de cruce en la misma.

En este procedimiento, la aplicación se limita a recibir instrucciones de la baliza, mediante el protocolo definido en el punto 7.3.7 y el framework BLE (basado en CoreBluetooth) y a informar al usuario mediante los comandos de voz pregrabrados

El protocolo comienza con un reconocimiento previo y la siguiente pantalla:

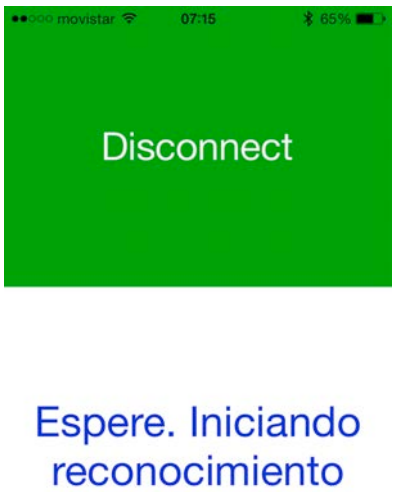


Ilustración 59. Pantalla inicio reconocimiento.

Tras iniciar el reconocimiento, se ejecutan los algoritmos de detección de vehículos a través del firmware la baliza, y en función del resultado de la detección, pueden aparecer las siguientes pantallas:

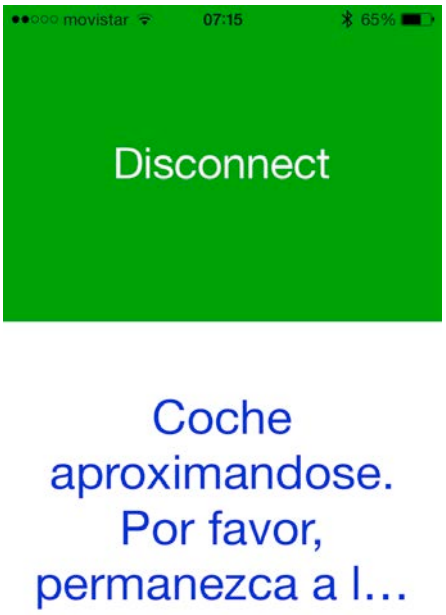


Ilustración 60. Pantalla de coche aproximándose.

Esta pantalla indica que el resultado de la detección ha informado de un coche aproximándose a velocidad suficiente como para posibilitar el conflicto peatón-vehículo y que se debe permanecer a la espera.

O puede aparecer la siguiente otra pantalla:

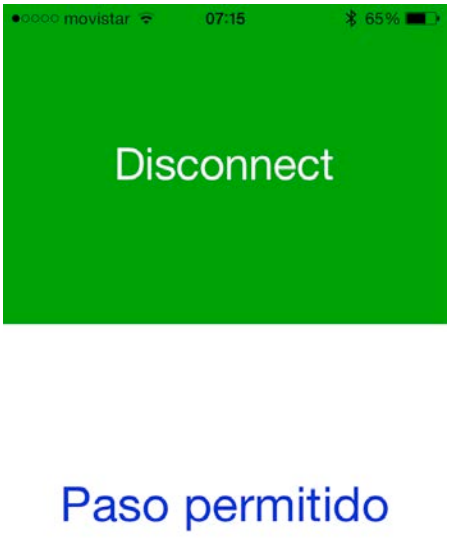


Ilustración 61. Pantalla de paso permitido.

Que indica que por el contrario no se han detectado coches aproximándose al cruce, o los coches que se acercan no ponen en peligro la condición de conflicto peatón-vehículo.
El cruce se puede realizar de forma segura.

Una vez se ha dado la señal de paso permitido, durante el transcurso del cruce, pueden aproximarse vehículos que afecten a la condición de conflicto peatón-vehículo, por lo que se informará al usuario y se pedirá mediante comandos acústicos la agilización del paso en el cruce.

Además, la aplicación informa del número de vehículos que se encuentran aproximándose en estas condiciones.

También, se informará al usuario mediante comandos acústicos de la posibilidad de placas de hielo en el pavimento, una vez se ha informado el paso libre.

Tras un tiempo predeterminado, la aplicación entiende que el cruce se ha realizado con éxito y se procede a la desconexión con la baliza, informando de ello al usuario mediante la siguiente pantalla:

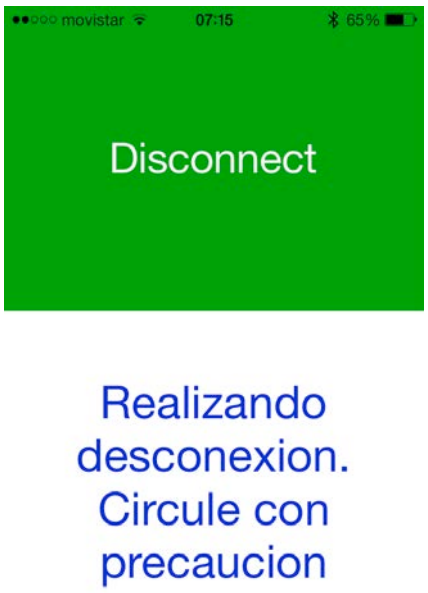


Ilustración 62. Pantalla de desconexión con la baliza

Acto seguido, se retorna al punto de inicio, mostrando el mapa con la ubicación de usuario, a la espera de la entrada en una nueva región balizada.

7.5 Desarrollo de la parte mecánica.

Se ha realizado el desarrollo de los planos mecánicos basados en el diseño conceptual del punto 5.5. para cada módulo electrónico, utilizando el software de diseño industrial Autodesk Inventor.

Dichos planos se pueden consultar en el ANEXO F.

Posteriormente se han fabricado las piezas diseñadas con PLA mediante una impresora 3D.

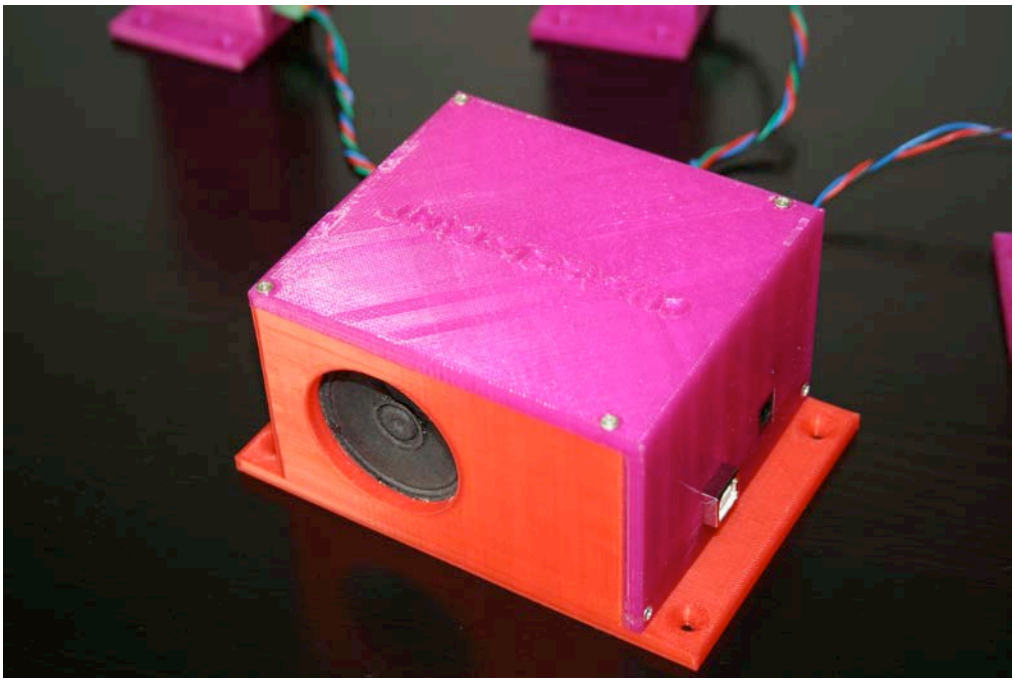


Ilustración 63. Mecánica de la caja principal de comunicaciones

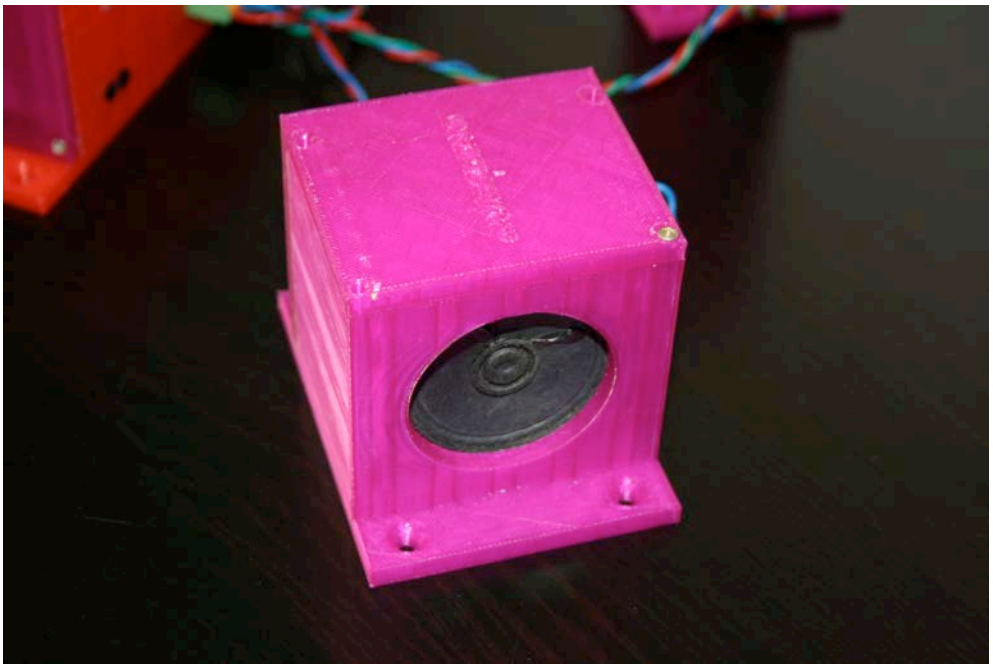


Ilustración 64. Mecánica de la caja auxiliar de señalización.

Ilustración 65. Mecánica de las torres de detección.

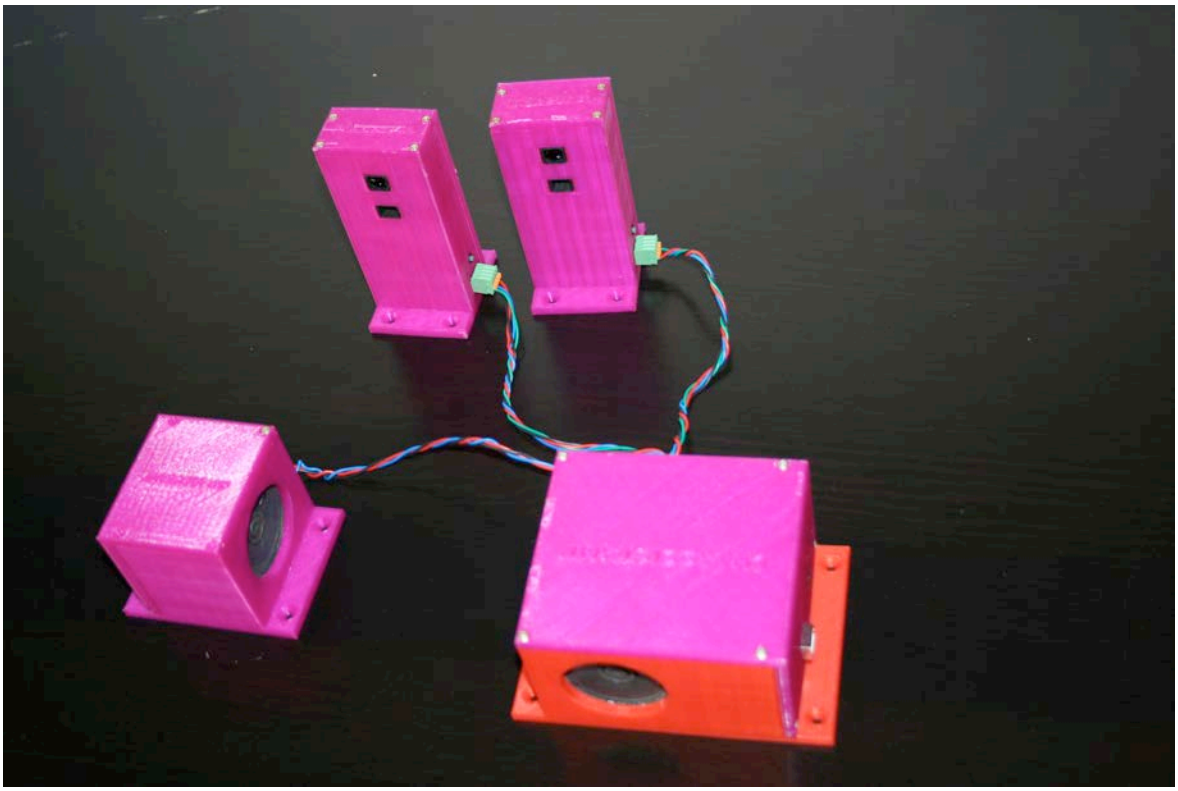


Ilustración 66. Sistema completo

8 PRUEBAS Y RESULTADOS

Para las pruebas, se ha empleado un sistema basado en Arduino de un proyecto personal desarrollado de forma paralela. Este sistema, se basa en el control de 4 motores paso a paso para mover barras metálicas.

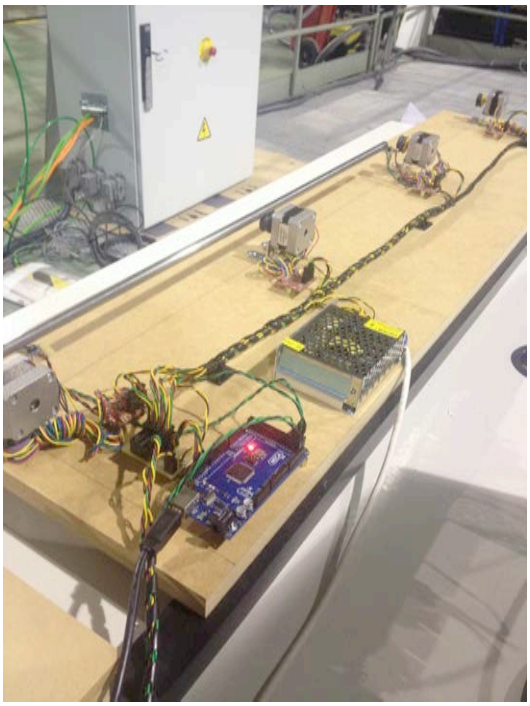


Ilustración 67. Sistema de medición de velocidad

A través de una sencilla aplicación desarrollada en .NET se puede establecer una consigna de velocidad para los motores, y activar y desactivar el movimiento de los mismos.

Se utilizará este sistema para simular el paso de un vehículo y calcular la precisión del prototipo de la baliza en medición de velocidad.

Se han colocado los detectores debajo de la línea de paso de barra, para simular la detección de paso de un objeto.

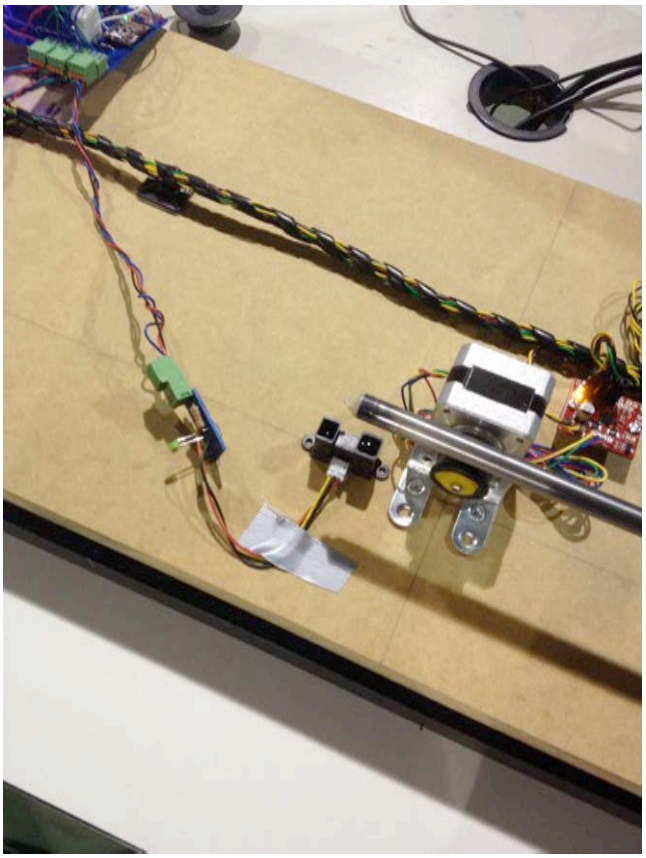


Ilustración 68. Posición de uno de los detectores con respecto a la barra

Las pruebas se han desarrollado en el interior de una nave industrial.

En un primer instante, se inicia la aplicación. En el momento de abrir la puerta de la nave en la que se encuentra instalado el dispositivo, el rango de activación programado en la aplicación activa el segue que inicia el reconocimiento.

Se ha ajustado la distancia entre detectores a la distancia real de la instalación, que es de 40cm.

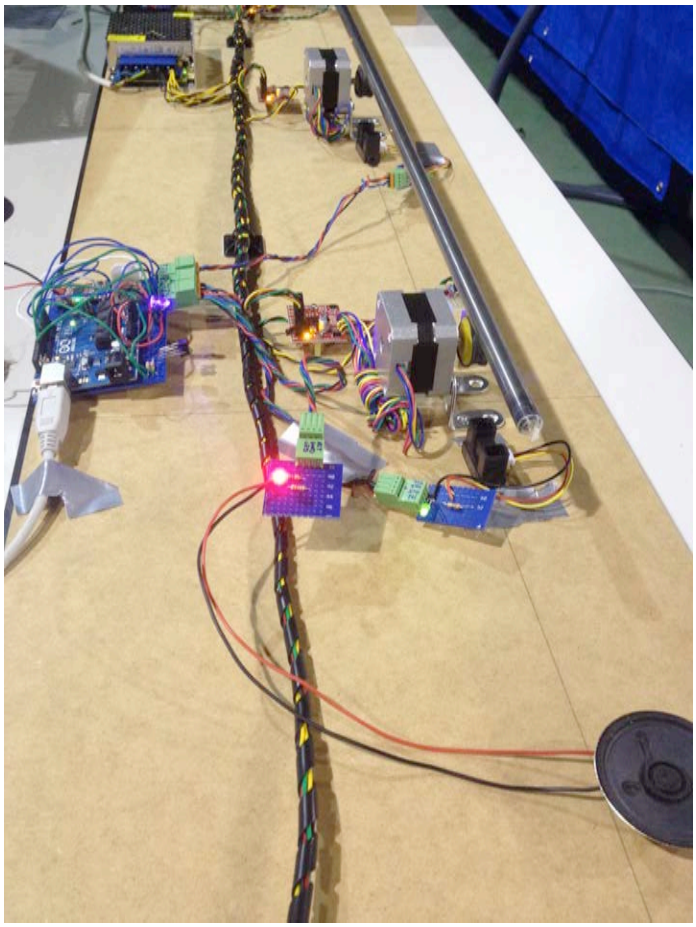


Ilustración 69. Detalle de la instalación de los detectores

Se realiza el ajuste de velocidad, mediante los controles de la aplicación en .NET

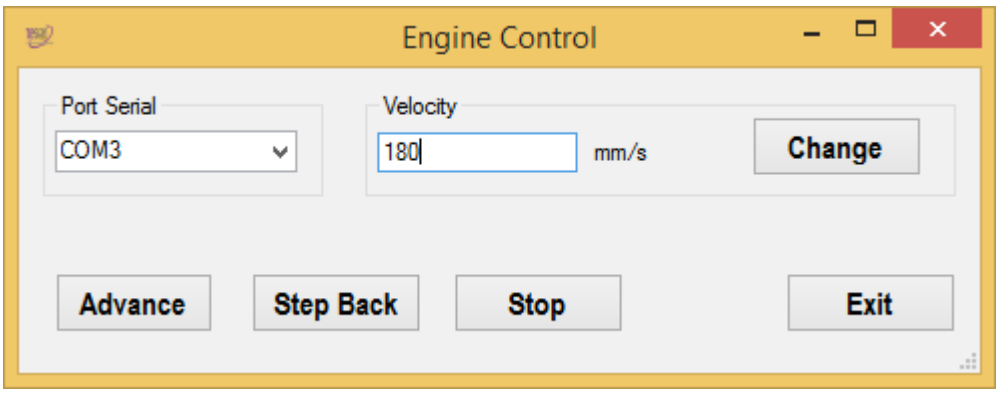


Ilustración 70. Detalle de la ventana de la aplicación en .NET para el cotrol de velocidad

Y se ejecuta el avance de la barra.

Se ha realizado una modificación en la aplicación, para que muestre por la consola serie la velocidad detectada en el objeto.

El resultado de la prueba es el siguiente:

Ilustración 71. Captura de los resultados de la prueba de medida de velocidad.

Se puede comprobar que el resultado obtenido todas las veces que se ha pasado la barra, es de 0,2m/s.

La diferencia entre la velocidad real de la barra y la velocidad detectada por el sistema es de 0,02m/s, en 40cm.

Por tanto los resultados obtenidos, suponen un error de 0,02m/s cada 0,04m. Es decir el error es de 0,5m/s por metro.

Los resultados, por tanto, en este aspecto son muy satisfactorios, ya que están muy por encima de los 1,38m/s (5km/h) de la especificación.

Respecto al consumo, se puede concretar que ninguno de los dispositivos trabaja con intensidades mayores de las definidas en la especificación, por lo que, por diseño hardware, se cumple la especificación también en este sentido.

Por tanto, se puede concluir por los resultados de las pruebas descritas, que tanto las funcionalidades prácticas como las técnicas, cumplen los requisitos por encima de la especificación.

9 CONCLUSIONES TÉCNICAS Y TRABAJO FUTURO.

9.1 Conclusiones

Las conclusiones del proyecto son positivas, partiendo de la base de que se han cumplido los requisitos por encima de las expectativas.

Es cierto que el desarrollo final ha dado como fruto un sistema que constituye únicamente un prototipo, que para una implementación final debería implementar ciertas modificaciones.

Principalmente la parte mecánica es la más susceptible de sufrir mejoras en una implementación real, así como el cableado, que al necesitar distancias largas en su implementación, quizá debiera utilizarse un sistema inalámbrico, o con mayor potencia.

Otro aspecto a tener en cuenta, es que la aplicación es dependiente de varios factores críticos como son la distancia entre balizas, distancia entre baliza y paso de peatones y estimación temporal del cruce del peatón.

Sin embargo, los resultados sobre los aspectos funcionales del sistema son muy satisfactorios.

Se trata de una aplicación de fácil manejo, que no necesita mirar la pantalla para poder utilizarla, y que informa, tanto desde el dispositivo como desde los actuadores acústicos de la baliza de una forma intuitiva de lo que sucede en el entorno.

La aplicación es funcional también en cuanto al desarrollo de las diferentes tareas.

El rango de actuación de la baliza es efectivo así como el reconocimiento geográfico del mismo, y es sencillo identificar y realizar dicha conexión.

Los resultados de las pruebas de campo han determinado que la precisión de la medición de velocidad es aceptable. Y que la detección del conflicto peatón-vehículo también entra de los márgenes de aceptabilidad.

Por tanto, la conclusión final, visto que se cumplen todos los requisitos definidos en la especificación del punto 3, es que los resultados son satisfactorios y se ha diseñado y desarrollado con éxito el prototipo que se pretendía.

9.2 Trabajo futuro

Como trabajo futuro se propone la realización de las siguientes mejoras sobre el sistema:

- Automatización del reconocimiento de posición del peatón respecto al paso de cebra.

  Esta automatización eliminaría la necesidad de estimar un tiempo de cruce del peatón en el paso, y por tanto haría más efectiva la detección del conflicto peatón-vehículo.

- Eliminación del cableado correspondiente a las torres de detección y de la caja auxiliar de señalización.

  Gracias a esta eliminación, el sistema sería más y mejor realizable como producto final.

- Empleo de diferentes tipos de sensores según el tipo de vía. Uno de los problemas fundamentales del sistema en funcionalidad práctica, reside en el paso de dos vehículos simultáneamente, ya que se necesita un tiempo de detención del bucle del proceso para realizar el reconocimiento una vez se ha producido una detección en el primer sensor.

  Además en vías con más de un carril por sentido, sólo se realizaría la detección sobre el vehículo más cercano en caso de paso de dos vehículos de forma paralela.

  Por estos motivos, sería necesario realizar pruebas con diferentes tipos de detectores y algoritmos de detección.

- Integración del sistema de guiado del DSLab con el sistema de este proyecto.

- Desarrollo de un sistema de posicionamiento basado en balizas BLE para interiores de edificios y apertura de puertas automática. Integración del mismo con los dos sistemas anteriores.

  Con un desarrollo de este tipo, se consolidarían prácticamente la gran mayoría de necesidades de guiado y se obtendría un producto con gran potencial en el mercado.

  No sólo para personas con discapacidad visual, sino además para otros perfiles de usuarios y otro tipo de aplicaciones.

Referencias

[1] M.H. Chodhury, D.Aguerrevere, and A.B. Barreto, "A Pocket-PC Based Navigational Aid for Blind Individuals", In Proc. Of IEEE Int.Conf. on Virtual Environments, Human Computer Interfaces, and Measurement Systems, pp. 43-48,2004.

[2] S.Chumkamon, P.Tuvaphanthaphiphat, and P.Keeratiwintakorn, "A blind navigation system using RFID for indoor environments," In Proc. of Int. Conf. on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, pp. 765-768, 2008.

[3] B. Ding, H. Yuan, X. Zang, and L. Jiang, "The Research on Blind Navigation System Based on RFID," In Proc. of Wireless Communications, Networking and Mobile Computing, pp. 2058-2061, 2007.

Bibliografía

- V. Nahavandipoor, “iOS 5 Programming Cookbook”, Oreilly 2012.
- A.Allan, “Geolocation in iOS”, Oreilly 2012.
- A.Mishra, G.Backlin, “iPhone ans iPad App 24-Hour Trainer”, Wrox 2012.
- E.Sadun “The iOS Developer’s Cookbook”, Addison Wesley 2012.
- M.Neuberg, “Programming iOS 5”, Oreilly 2012
- A.Allan “Learning iOS Programming”, Oreilly 2012.
- C.Adamson and B.Dudney, “iOS SDK Development”, The Pragmatic Programmers 2012.
- R.Lewis “iPhone and iPad Apps for Absolute Beginners iOS5 Edition”, Apress 2012.
- Michael Margolis “Arduino Cookbook”, Oreilly 2011
- John Iovine “Microcontroller Project Book” TAB Robotics 2013
- Robin Heydon “Bluetooth Low Energy: The Developer’s Handbook”, Prentice Hall 2010
- Naresh Gupta “Inside Bluetooth Low Energy”, Mobile Communications Series 2012

Glosario

API	Application Programming Interface
BLE	Bluetooth Low Energy
iOS	iPhone Operating System
ONCE	Organicación Nacional de Ciegos Españoles
GPS	Global Positioning System
SDK	Software Development Kit
IOT	Internet Of Things
IR	Infra Red
WSN	Wireless Sensor Networks
RSI	Redes de Sensores Inalámbricas
PCB	Printed Circuit Board
RFID	Radio Frequency Identification
PIC	Peripheral Interface Controller
RAM	Random Access Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
IC	Integrated Circuit
CPU	Central Processing Unit
GPU	Graphics Processing Unit
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
I^2C	Inter-Integrated Circuit
GPIO	General Purpose Input/Output
IP	Ingress Protection rating

Anexos

A Librería Blemini.h

Blemini.h

```
#ifndef  _BLE_MINI_H
#define _BLE_MINI_H

#include <Arduino.h>

void BLEMini_begin(unsigned long bound);
int BLEMini_available();
void BLEMini_write(unsigned char dat);
void BLEMini_write_bytes(unsigned char *dat, unsigned char len);
int BLEMini_read();

#endif
```

Blemini.cpp

```
#include "ble_Mini.h"

// UNO
// TX: pin 1
// RX: pin 0
#if defined (__AVR_ATmega168__) || defined (__AVR_ATmega328P__)
    #define BLEMINI Serial

// other board
// board                        TX              RX
// Leonardo                1                    0
// MEGA                         18              19
// DUE                                  18              19
#else
    #define BLEMINI Serial1
#endif

void BLEMini_begin(unsigned long bound)
{
     BLEMINI.begin(bound);
}

int BLEMini_available()
{
     return BLEMINI.available();
}
```

```
void BLEMini_write(unsigned char dat)
{
      BLEMINI.write(dat);
}

void BLEMini_write_bytes(unsigned char *dat, unsigned char len)
{
      delay(10);
      if(len > 0)
            BLEMINI.write(dat, len);
}

int BLEMini_read()
{
    delay(10);
    return BLEMINI.read();
}
```

B Librería DHT.h

```
#ifndef DHT_H
#define DHT_H
#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

/* DHT library

MIT license
written by Adafruit Industries
*/

// how many timing transitions we need to keep track of. 2 *
number bits + extra
#define MAXTIMINGS 85

#define DHT11 11
#define DHT22 22
#define DHT21 21
#define AM2301 21

class DHT {
 private:
  uint8_t data[6];
  uint8_t _pin, _type, _count;
  boolean read(void);
  unsigned long _lastreadtime;
  boolean firstreading;

 public:
  DHT(uint8_t pin, uint8_t type, uint8_t count=6);
  void begin(void);
  float readTemperature(bool S=false);
  float convertCtoF(float);
  float readHumidity(void);

};
#endif
```

DHT.cpp

```
/* DHT library

MIT license
written by Adafruit Industries
*/

#include "DHT.h"

DHT::DHT(uint8_t pin, uint8_t type, uint8_t count) {
  _pin = pin;
  _type = type;
  _count = count;
  firstreading = true;
}

void DHT::begin(void) {
  // set up the pins!
  pinMode(_pin, INPUT);
  digitalWrite(_pin, HIGH);
  _lastreadtime = 0;
}

//boolean S == Scale.  True == Farenheit; False == Celcius
float DHT::readTemperature(bool S) {
  float f;

  if (read()) {
    switch (_type) {
    case DHT11:
      f = data[2];
      if(S)
           f = convertCtoF(f);

      return f;
    case DHT22:
    case DHT21:
      f = data[2] & 0x7F;
      f *= 256;
      f += data[3];
      f /= 10;
      if (data[2] & 0x80)
    f *= -1;
      if(S)
    f = convertCtoF(f);

      return f;
    }
  }
  Serial.print("Read fail");
  return NAN;
}
```

```
float DHT::convertCtoF(float c) {
    return c * 9 / 5 + 32;
}

float DHT::readHumidity(void) {
  float f;
  if (read()) {
    switch (_type) {
    case DHT11:
      f = data[0];
      return f;
    case DHT22:
    case DHT21:
      f = data[0];
      f *= 256;
      f += data[1];
      f /= 10;
      return f;
    }
  }
  Serial.print("Read fail");
  return NAN;
}


boolean DHT::read(void) {
  uint8_t laststate = HIGH;
  uint8_t counter = 0;
  uint8_t j = 0, i;
  unsigned long currenttime;

  // pull the pin high and wait 250 milliseconds
  digitalWrite(_pin, HIGH);
  delay(250);

  currenttime = millis();
  if (currenttime < _lastreadtime) {
    // ie there was a rollover
    _lastreadtime = 0;
  }
  if (!firstreading && ((currenttime - _lastreadtime) < 2000)) {
    return true; // return last correct measurement
    //delay(2000 - (currenttime - _lastreadtime));
  }
  firstreading = false;
  /*
    Serial.print("Currtime: "); Serial.print(currenttime);
    Serial.print(" Lasttime: "); Serial.print(_lastreadtime);
  */
  _lastreadtime = millis();

  data[0] = data[1] = data[2] = data[3] = data[4] = 0;

  // now pull it low for ~20 milliseconds
```

```
  pinMode(_pin, OUTPUT);
  digitalWrite(_pin, LOW);
  delay(20);
  cli();
  digitalWrite(_pin, HIGH);
  delayMicroseconds(40);
  pinMode(_pin, INPUT);

  // read in timings
  for ( i=0; i< MAXTIMINGS; i++) {
    counter = 0;
    while (digitalRead(_pin) == laststate) {
      counter++;
      delayMicroseconds(1);
      if (counter == 255) {
        break;
      }
    }
    laststate = digitalRead(_pin);

    if (counter == 255) break;

    // ignore first 3 transitions
    if ((i >= 4) && (i%2 == 0)) {
      // shove each bit into the storage bytes
      data[j/8] <<= 1;
      if (counter > _count)
        data[j/8] |= 1;
      j++;
    }

  }

  sei();

  /*
  Serial.println(j, DEC);
  Serial.print(data[0], HEX); Serial.print(", ");
  Serial.print(data[1], HEX); Serial.print(", ");
  Serial.print(data[2], HEX); Serial.print(", ");
  Serial.print(data[3], HEX); Serial.print(", ");
  Serial.print(data[4], HEX); Serial.print(" =? ");
  Serial.println(data[0] + data[1] + data[2] + data[3], HEX);
  */

  // check we read 40 bits and that the checksum matches
  if ((j >= 40) &&
      (data[4] == ((data[0] + data[1] + data[2] + data[3]) &
0xFF)) ) {
    return true;
  }


  return false;}
```

C Librería pitches.h

```
/*************************************************
 * Public Constants
 *************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
```

```
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

D BLE framework

BLE.h

```
/*

 Copyright (c) 2013 RedBearLab

 Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy,
modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

 The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.

*/

#import <Foundation/Foundation.h>
#if TARGET_OS_IPHONE
    #import <CoreBluetooth/CoreBluetooth.h>
#else
    #import <IOBluetooth/IOBluetooth.h>
#endif

@protocol BLEDelegate
@optional
-(void) bleDidConnect;
-(void) bleDidDisconnect;
-(void) bleDidUpdateRSSI:(NSNumber *) rssi;
-(void) bleDidReceiveData:(unsigned char *) data length:(int)
length;
@required
@end

@interface BLE : NSObject <CBCentralManagerDelegate,
CBPeripheralDelegate> {

}
```

```
@property (nonatomic,assign) id <BLEDelegate> delegate;
@property (strong, nonatomic) NSMutableArray *peripherals;
@property (strong, nonatomic) CBCentralManager *CM;
@property (strong, nonatomic) CBPeripheral *activePeripheral;

-(void) enableReadNotification:(CBPeripheral *)p;
-(void) read;
-(void) writeValue:(CBUUID *)serviceUUID
characteristicUUID:(CBUUID *)characteristicUUID p:(CBPeripheral
*)p data:(NSData *)data;

-(BOOL) isConnected;
-(void) write:(NSData *)d;
-(void) readRSSI;

-(void) controlSetup;
-(int) findBLEPeripherals:(int) timeout;
-(void) connectPeripheral:(CBPeripheral *)peripheral;

-(UInt16) swap:(UInt16) s;
-(const char *) centralManagerStateToString:(int)state;
-(void) scanTimer:(NSTimer *)timer;
-(void) printKnownPeripherals;
-(void) printPeripheralInfo:(CBPeripheral*)peripheral;

-(void) getAllServicesFromPeripheral:(CBPeripheral *)p;
-(void) getAllCharacteristicsFromPeripheral:(CBPeripheral *)p;
-(CBService *) findServiceFromUUID:(CBUUID *)UUID p:(CBPeripheral
*)p;
-(CBCharacteristic *) findCharacteristicFromUUID:(CBUUID *)UUID
service:(CBService*)service;

//-(NSString *) NSUUIDToString:(NSUUID *) UUID;
-(NSString *) CBUUIDToString:(CBUUID *) UUID;

-(int) compareCBUUID:(CBUUID *) UUID1 UUID2:(CBUUID *)UUID2;
-(int) compareCBUUIDToInt:(CBUUID *) UUID1 UUID2:(UInt16)UUID2;
-(UInt16) CBUUIDToInt:(CBUUID *) UUID;
-(BOOL) UUIDSAreEqual:(NSUUID *)UUID1 UUID2:(NSUUID *)UUID2;

@end
```

BLEDefines.h

```
/*

 Copyright (c) 2013 RedBearLab

 Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy,
modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

 The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.

 */

// RBL Service
#define RBL_SERVICE_UUID                        "713D0000-503E-
4C75-BA94-3148F18D941E"
#define RBL_CHAR_TX_UUID                        "713D0002-503E-
4C75-BA94-3148F18D941E"
#define RBL_CHAR_RX_UUID                        "713D0003-503E-
4C75-BA94-3148F18D941E"

#define RBL_BLE_FRAMEWORK_VER                   0x0200
```

BLE.m

```
/*

 Copyright (c) 2013 RedBearLab

 Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy,
modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

 The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.

*/

#import "BLE.h"
#import "BLEDefines.h"

@implementation BLE

@synthesize delegate;
@synthesize CM;
@synthesize peripherals;
@synthesize activePeripheral;

static bool isConnected = false;
static int rssi = 0;

-(void) readRSSI
{
    [activePeripheral readRSSI];
}

-(BOOL) isConnected
{
    return isConnected;
}

-(void) read
{
```

```
    CBUUID *uuid_service = [CBUUID
UUIDWithString:@RBL_SERVICE_UUID];
    CBUUID *uuid_char = [CBUUID UUIDWithString:@RBL_CHAR_TX_UUID];

    [self readValue:uuid_service characteristicUUID:uuid_char
p:activePeripheral];
}

-(void) write:(NSData *)d
{
    CBUUID *uuid_service = [CBUUID
UUIDWithString:@RBL_SERVICE_UUID];
    CBUUID *uuid_char = [CBUUID UUIDWithString:@RBL_CHAR_RX_UUID];

    [self writeValue:uuid_service characteristicUUID:uuid_char
p:activePeripheral data:d];
}

-(void) enableReadNotification:(CBPeripheral *)p
{
    CBUUID *uuid_service = [CBUUID
UUIDWithString:@RBL_SERVICE_UUID];
    CBUUID *uuid_char = [CBUUID UUIDWithString:@RBL_CHAR_TX_UUID];

    [self notification:uuid_service characteristicUUID:uuid_char
p:p on:YES];
}

-(void) notification:(CBUUID *)serviceUUID
characteristicUUID:(CBUUID *)characteristicUUID p:(CBPeripheral
*)p on:(BOOL)on
{
    CBService *service = [self findServiceFromUUID:serviceUUID
p:p];

    if (!service)
    {
        NSLog(@"Could not find service with UUID %@ on peripheral
with UUID %@",
              [self CBUUIDToString:serviceUUID],
              p.identifier.UUIDString);

        return;
    }

    CBCharacteristic *characteristic = [self
findCharacteristicFromUUID:characteristicUUID service:service];

    if (!characteristic)
    {
        NSLog(@"Could not find characteristic with UUID %@ on
service with UUID %@ on peripheral with UUID %@",
              [self CBUUIDToString:characteristicUUID],
              [self CBUUIDToString:serviceUUID],
```

```
                p.identifier.UUIDString);

        return;
    }

    [p setNotifyValue:on forCharacteristic:characteristic];
}

-(UInt16) frameworkVersion
{
    return RBL_BLE_FRAMEWORK_VER;
}

-(NSString *) CBUUIDToString:(CBUUID *) cbuuid;
{
    NSData *data = cbuuid.data;

    if ([data length] == 2)
    {
        const unsigned char *tokenBytes = [data bytes];
        return [NSString stringWithFormat:@"%02x%02x",
tokenBytes[0], tokenBytes[1]];
    }
    else if ([data length] == 16)
    {
        NSUUID* nsuuid = [[NSUUID alloc] initWithUUIDBytes:[data
bytes]];
        return [nsuuid UUIDString];
    }

    return [cbuuid description];
}

-(void) readValue: (CBUUID *)serviceUUID
characteristicUUID:(CBUUID *)characteristicUUID p:(CBPeripheral
*)p
{
    CBService *service = [self findServiceFromUUID:serviceUUID
p:p];

    if (!service)
    {
        NSLog(@"Could not find service with UUID %@ on peripheral
with UUID %@",
                [self CBUUIDToString:serviceUUID],
                p.identifier.UUIDString);

        return;
    }

    CBCharacteristic *characteristic = [self
findCharacteristicFromUUID:characteristicUUID service:service];

    if (!characteristic)
```

```
    {
        NSLog(@"Could not find characteristic with UUID %@ on
service with UUID %@ on peripheral with UUID %@",
              [self CBUUIDToString:characteristicUUID],
              [self CBUUIDToString:serviceUUID],
              p.identifier.UUIDString);

        return;
    }

    [p readValueForCharacteristic:characteristic];
}

-(void) writeValue:(CBUUID *)serviceUUID
characteristicUUID:(CBUUID *)characteristicUUID p:(CBPeripheral
*)p data:(NSData *)data
{
    CBService *service = [self findServiceFromUUID:serviceUUID
p:p];

    if (!service)
    {
        NSLog(@"Could not find service with UUID %@ on peripheral
with UUID %@",
              [self CBUUIDToString:serviceUUID],
              p.identifier.UUIDString);

        return;
    }

    CBCharacteristic *characteristic = [self
findCharacteristicFromUUID:characteristicUUID service:service];

    if (!characteristic)
    {
        NSLog(@"Could not find characteristic with UUID %@ on
service with UUID %@ on peripheral with UUID %@",
              [self CBUUIDToString:characteristicUUID],
              [self CBUUIDToString:serviceUUID],
              p.identifier.UUIDString);

        return;
    }

    [p writeValue:data forCharacteristic:characteristic
type:CBCharacteristicWriteWithoutResponse];
}

-(UInt16) swap:(UInt16)s
{
    UInt16 temp = s << 8;
    temp |= (s >> 8);
    return temp;
}
```

```
- (void) controlSetup
{
    self.CM = [[CBCentralManager alloc] initWithDelegate:self
queue:nil];
}

- (int) findBLEPeripherals:(int) timeout
{
    if (self.CM.state != CBCentralManagerStatePoweredOn)
    {
        NSLog(@"CoreBluetooth not correctly initialized !");
        NSLog(@"State = %d (%s)\r\n", self.CM.state, [self
centralManagerStateToString:self.CM.state]);
        return -1;
    }

    [NSTimer scheduledTimerWithTimeInterval:(float)timeout
target:self selector:@selector(scanTimer:) userInfo:nil
repeats:NO];

#if TARGET_OS_IPHONE
    [self.CM scanForPeripheralsWithServices:[NSArray
arrayWithObject:[CBUUID UUIDWithString:@RBL_SERVICE_UUID]]
options:nil];
#else
    [self.CM scanForPeripheralsWithServices:nil options:nil]; //
Start scanning
#endif

    NSLog(@"scanForPeripheralsWithServices");

    return 0; // Started scanning OK !
}

- (void)centralManager:(CBCentralManager *)central
didDisconnectPeripheral:(CBPeripheral *)peripheral error:(NSError
*)error;
{
    done = false;

    [[self delegate] bleDidDisconnect];

    isConnected = false;
}

- (void) connectPeripheral:(CBPeripheral *)peripheral
{
    NSLog(@"Connecting to peripheral with UUID : %@",
peripheral.identifier.UUIDString);

    self.activePeripheral = peripheral;
    self.activePeripheral.delegate = self;
    [self.CM connectPeripheral:self.activePeripheral
```

```
                        options:[NSDictionary
dictionaryWithObject:[NSNumber numberWithBool:YES]
forKey:CBConnectPeripheralOptionNotifyOnDisconnectionKey]];
}

- (const char *) centralManagerStateToString: (int)state
{
    switch(state)
    {
        case CBCentralManagerStateUnknown:
            return "State unknown (CBCentralManagerStateUnknown)";
        case CBCentralManagerStateResetting:
            return "State resetting
(CBCentralManagerStateUnknown)";
        case CBCentralManagerStateUnsupported:
            return "State BLE unsupported
(CBCentralManagerStateResetting)";
        case CBCentralManagerStateUnauthorized:
            return "State unauthorized
(CBCentralManagerStateUnauthorized)";
        case CBCentralManagerStatePoweredOff:
            return "State BLE powered off
(CBCentralManagerStatePoweredOff)";
        case CBCentralManagerStatePoweredOn:
            return "State powered up and ready
(CBCentralManagerStatePoweredOn)";
        default:
            return "State unknown";
    }

    return "Unknown state";
}

- (void) scanTimer:(NSTimer *)timer
{
    [self.CM stopScan];
    NSLog(@"Stopped Scanning");
    NSLog(@"Known peripherals : %lu", (unsigned
long)[self.peripherals count]);
    [self printKnownPeripherals];
}

- (void) printKnownPeripherals
{
    NSLog(@"List of currently known peripherals :");

    for (int i = 0; i < self.peripherals.count; i++)
    {
        CBPeripheral *p = [self.peripherals objectAtIndex:i];

        if (p.identifier != NULL)
            NSLog(@"%d  |  %@", i, p.identifier.UUIDString);
        else
            NSLog(@"%d  |  NULL", i);
```

```
        [self printPeripheralInfo:p];
    }
}

- (void) printPeripheralInfo:(CBPeripheral*)peripheral
{
    NSLog(@"-------------------------------------");
    NSLog(@"Peripheral Info :");

    if (peripheral.identifier != NULL)
        NSLog(@"UUID : %@", peripheral.identifier.UUIDString);
    else
        NSLog(@"UUID : NULL");

    NSLog(@"Name : %@", peripheral.name);
    NSLog(@"-------------------------------------");
}

- (BOOL) UUIDSAreEqual:(NSUUID *)UUID1 UUID2:(NSUUID *)UUID2
{
    if ([UUID1.UUIDString isEqualToString:UUID2.UUIDString])
        return TRUE;
    else
        return FALSE;
}

-(void) getAllServicesFromPeripheral:(CBPeripheral *)p
{
    [p discoverServices:nil]; // Discover all services without
filter
}

-(void) getAllCharacteristicsFromPeripheral:(CBPeripheral *)p
{
    for (int i=0; i < p.services.count; i++)
    {
        CBService *s = [p.services objectAtIndex:i];
        //        printf("Fetching characteristics for service
with UUID : %s\r\n",[self CBUUIDToString:s.UUID]);
        [p discoverCharacteristics:nil forService:s];
    }
}

-(int) compareCBUUID:(CBUUID *) UUID1 UUID2:(CBUUID *)UUID2
{
    char b1[16];
    char b2[16];
    [UUID1.data getBytes:b1];
    [UUID2.data getBytes:b2];

    if (memcmp(b1, b2, UUID1.data.length) == 0)
        return 1;
    else
```

```
        return 0;
}

-(int) compareCBUUIDToInt:(CBUUID *)UUID1 UUID2:(UInt16)UUID2
{
    char b1[16];

    [UUID1.data getBytes:b1];
    UInt16 b2 = [self swap:UUID2];

    if (memcmp(b1, (char *)&b2, 2) == 0)
        return 1;
    else
        return 0;
}

-(UInt16) CBUUIDToInt:(CBUUID *) UUID
{
    char b1[16];
    [UUID.data getBytes:b1];
    return ((b1[0] << 8) | b1[1]);
}

-(CBUUID *) IntToCBUUID:(UInt16)UUID
{
    char t[16];
    t[0] = ((UUID >> 8) & 0xff); t[1] = (UUID & 0xff);
    NSData *data = [[NSData alloc] initWithBytes:t length:16];
    return [CBUUID UUIDWithData:data];
}

-(CBService *) findServiceFromUUID:(CBUUID *)UUID p:(CBPeripheral
*)p
{
    for(int i = 0; i < p.services.count; i++)
    {
        CBService *s = [p.services objectAtIndex:i];
        if ([self compareCBUUID:s.UUID UUID2:UUID])
            return s;
    }

    return nil; //Service not found on this peripheral
}

-(CBCharacteristic *) findCharacteristicFromUUID:(CBUUID *)UUID
service:(CBService*)service
{
    for(int i=0; i < service.characteristics.count; i++)
    {
        CBCharacteristic *c = [service.characteristics
objectAtIndex:i];
        if ([self compareCBUUID:c.UUID UUID2:UUID]) return c;
    }
```

```
    return nil; //Characteristic not found on this service
}

#if TARGET_OS_IPHONE
    //-- no need for iOS
#else
- (BOOL) isLECapableHardware
{
    NSString * state = nil;

    switch ([CM state])
    {
        case CBCentralManagerStateUnsupported:
            state = @"The platform/hardware doesn't support
Bluetooth Low Energy.";
            break;

        case CBCentralManagerStateUnauthorized:
            state = @"The app is not authorized to use Bluetooth
Low Energy.";
            break;

        case CBCentralManagerStatePoweredOff:
            state = @"Bluetooth is currently powered off.";
            break;

        case CBCentralManagerStatePoweredOn:
            return TRUE;

        case CBCentralManagerStateUnknown:
        default:
            return FALSE;

    }

    NSLog(@"Central manager state: %@", state);

    NSAlert *alert = [[NSAlert alloc] init];
    [alert setMessageText:state];
    [alert addButtonWithTitle:@"OK"];
    [alert setIcon:[[NSImage alloc]
initWithContentsOfFile:@"AppIcon"]];
    [alert beginSheetModalForWindow:nil modalDelegate:self
didEndSelector:nil contextInfo:nil];

    return FALSE;
}
#endif

- (void)centralManagerDidUpdateState:(CBCentralManager *)central
{
#if TARGET_OS_IPHONE
    NSLog(@"Status of CoreBluetooth central manager changed %d
(%s)", central.state, [self
```

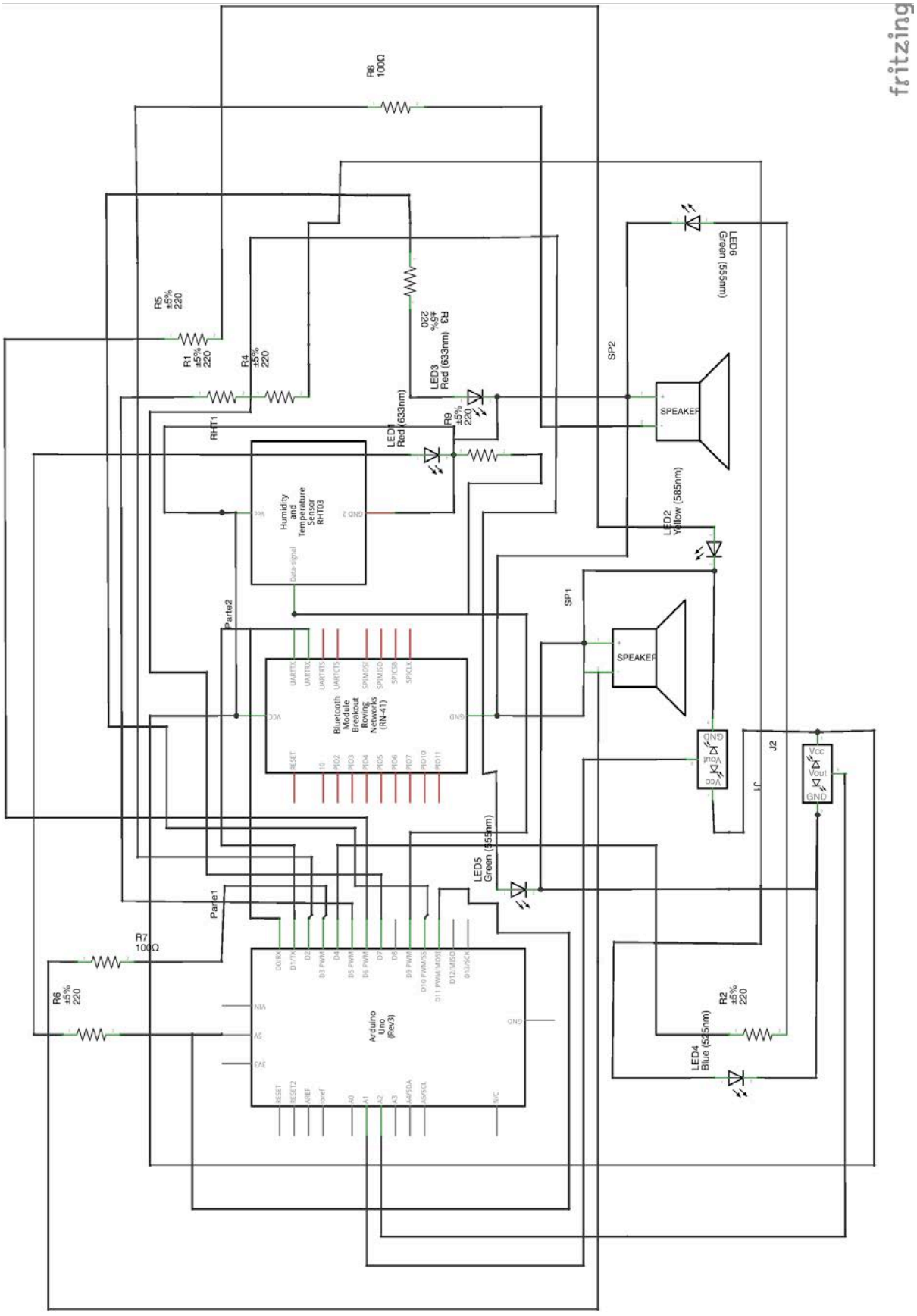
```
centralManagerStateToString:central.state]);
#else
    [self isLECapableHardware];
#endif
}

- (void)centralManager:(CBCentralManager *)central
didDiscoverPeripheral:(CBPeripheral *)peripheral
advertisementData:(NSDictionary *)advertisementData RSSI:(NSNumber
*)RSSI
{
    if (!self.peripherals)
        self.peripherals = [[NSMutableArray alloc]
initWithObjects:peripheral,nil];
    else
    {
        for(int i = 0; i < self.peripherals.count; i++)
        {
            CBPeripheral *p = [self.peripherals objectAtIndex:i];

            if ((p.identifier == NULL) || (peripheral.identifier
== NULL))
                continue;

            if ([self UUIDSAreEqual:p.identifier
UUID2:peripheral.identifier])
            {
                [self.peripherals replaceObjectAtIndex:i
withObject:peripheral];
                NSLog(@"Duplicate UUID found updating...");
                return;
            }
        }

        [self.peripherals addObject:peripheral];

        NSLog(@"New UUID, adding");
    }

    NSLog(@"didDiscoverPeripheral");
}

- (void)centralManager:(CBCentralManager *)central
didConnectPeripheral:(CBPeripheral *)peripheral
{
    if (peripheral.identifier != NULL)
        NSLog(@"Connected to %@ successful",
peripheral.identifier.UUIDString);
    else
        NSLog(@"Connected to NULL successful");

    self.activePeripheral = peripheral;
    [self.activePeripheral discoverServices:nil];
    [self getAllServicesFromPeripheral:peripheral];
```

```
}

static bool done = false;

- (void)peripheral:(CBPeripheral *)peripheral
didDiscoverCharacteristicsForService:(CBService *)service
error:(NSError *)error
{
    if (!error)
    {
        //        printf("Characteristics of service with UUID :
%s found\n",[self CBUUIDToString:service.UUID]);

        for (int i=0; i < service.characteristics.count; i++)
        {
            //            CBCharacteristic *c =
[service.characteristics objectAtIndex:i];
            //            printf("Found characteristic %s\n",[
self CBUUIDToString:c.UUID]);
            CBService *s = [peripheral.services
objectAtIndex:(peripheral.services.count - 1)];

            if ([service.UUID isEqual:s.UUID])
            {
                if (!done)
                {
                    [self
enableReadNotification:activePeripheral];
                    [[self delegate] bleDidConnect];
                    isConnected = true;
                    done = true;
                }

                break;
            }
        }
    }
    else
    {
        NSLog(@"Characteristic discorvery unsuccessful!");
    }
}

- (void)peripheral:(CBPeripheral *)peripheral
didDiscoverServices:(NSError *)error
{
    if (!error)
    {
        //        printf("Services of peripheral with UUID : %s
found\n",[self UUIDToString:peripheral.UUID]);
        [self getAllCharacteristicsFromPeripheral:peripheral];
    }
    else
    {
```

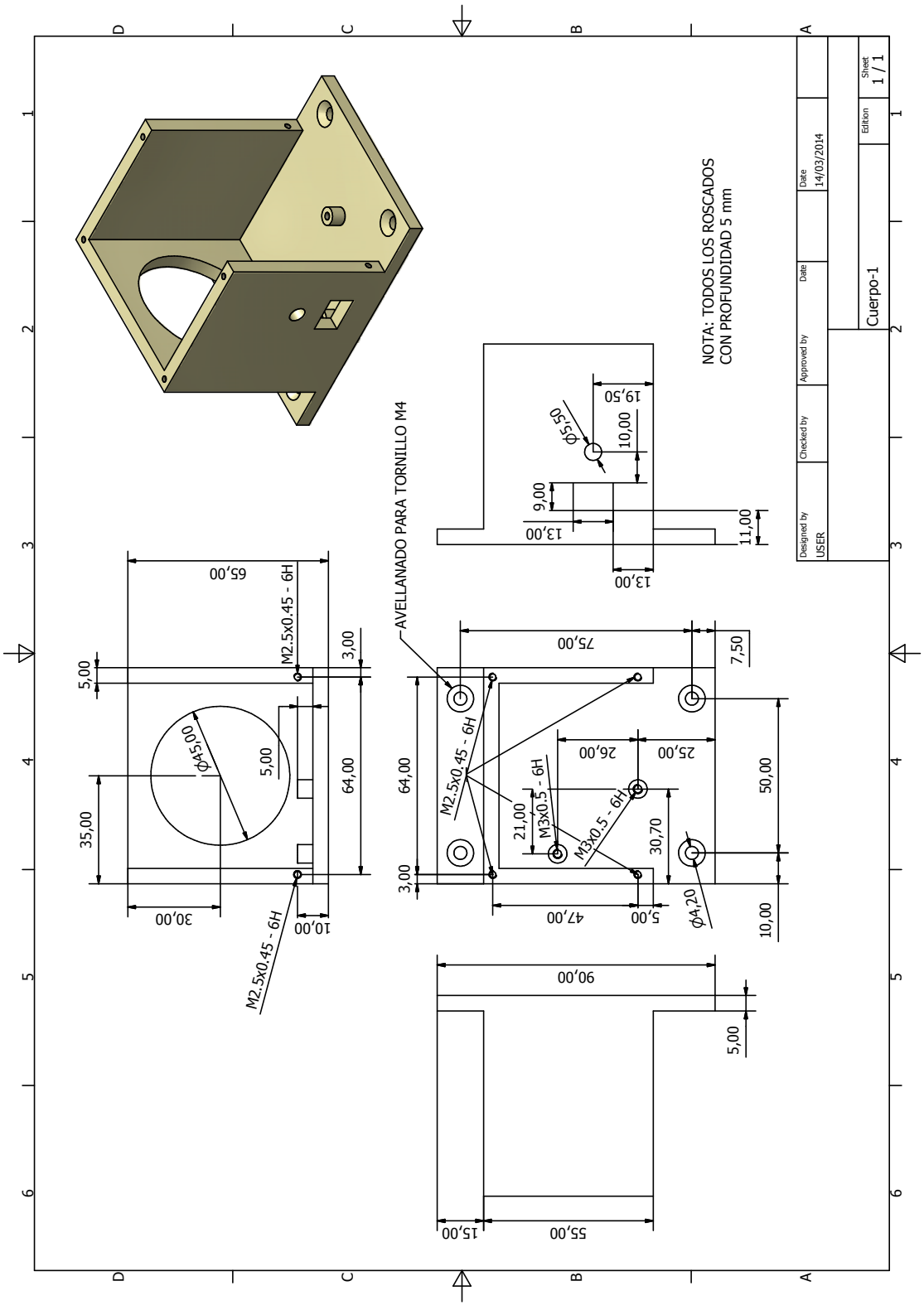
```
        NSLog(@"Service discovery was unsuccessful!");
    }
}

- (void)peripheral:(CBPeripheral *)peripheral
didUpdateNotificationStateForCharacteristic:(CBCharacteristic
*)characteristic error:(NSError *)error
{
    if (!error)
    {
        //        printf("Updated notification state for
characteristic with UUID %s on service with  UUID %s on peripheral
with UUID %s\r\n",[self CBUUIDToString:characteristic.UUID],[self
CBUUIDToString:characteristic.service.UUID],[self
UUIDToString:peripheral.UUID]);
    }
    else
    {
        NSLog(@"Error in setting notification state for
characteristic with UUID %@ on service with UUID %@ on peripheral
with UUID %@",
              [self CBUUIDToString:characteristic.UUID],
              [self CBUUIDToString:characteristic.service.UUID],
              peripheral.identifier.UUIDString);

        NSLog(@"Error code was %s", [[error description]
cStringUsingEncoding:NSStringEncodingConversionAllowLossy]);
    }
}

- (void)peripheral:(CBPeripheral *)peripheral
didUpdateValueForCharacteristic:(CBCharacteristic *)characteristic
error:(NSError *)error
{
    unsigned char data[20];

    static unsigned char buf[512];
    static int len = 0;
    NSInteger data_len;

    if (!error)
    {
        if ([characteristic.UUID isEqual:[CBUUID
UUIDWithString:@RBL_CHAR_TX_UUID]])
        {
            data_len = characteristic.value.length;
            [characteristic.value getBytes:data length:data_len];

            if (data_len == 20)
            {
                memcpy(&buf[len], data, 20);
                len += data_len;

                if (len >= 64)
```

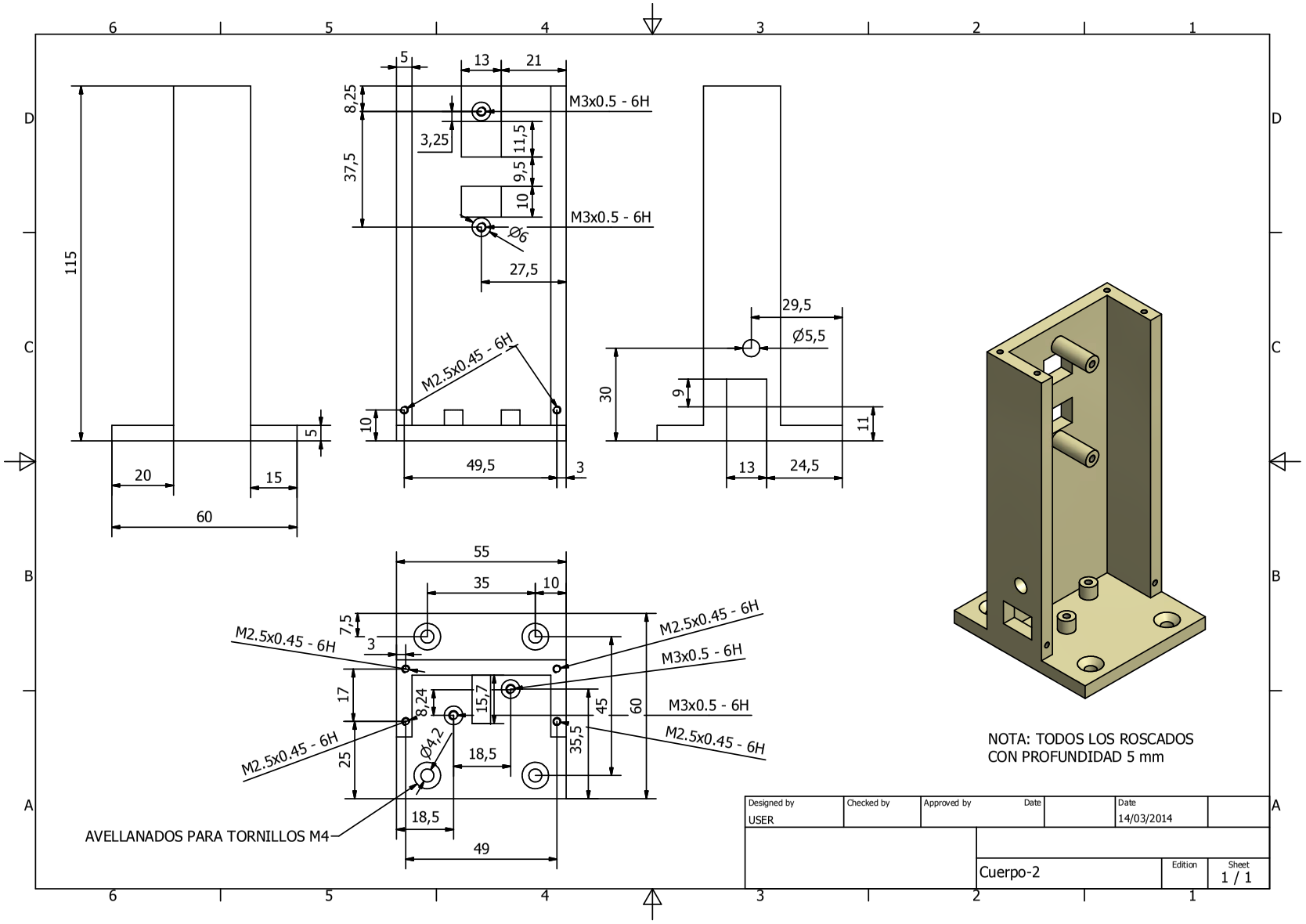
```
                {
                    [[self delegate] bleDidReceiveData:buf
length:len];
                    len = 0;
                }
            }
            else if (data_len < 20)
            {
                memcpy(&buf[len], data, data_len);
                len += data_len;

                [[self delegate] bleDidReceiveData:buf
length:len];
                len = 0;
            }
        }
    }
    else
    {
        NSLog(@"updateValueForCharacteristic failed!");
    }
}

- (void)peripheralDidUpdateRSSI:(CBPeripheral *)peripheral
error:(NSError *)error
{
    if (!isConnected)
        return;

    if (rssi != peripheral.RSSI.intValue)
    {
        rssi = peripheral.RSSI.intValue;
        [[self delegate] bleDidUpdateRSSI:activePeripheral.RSSI];
    }
}

@end
```

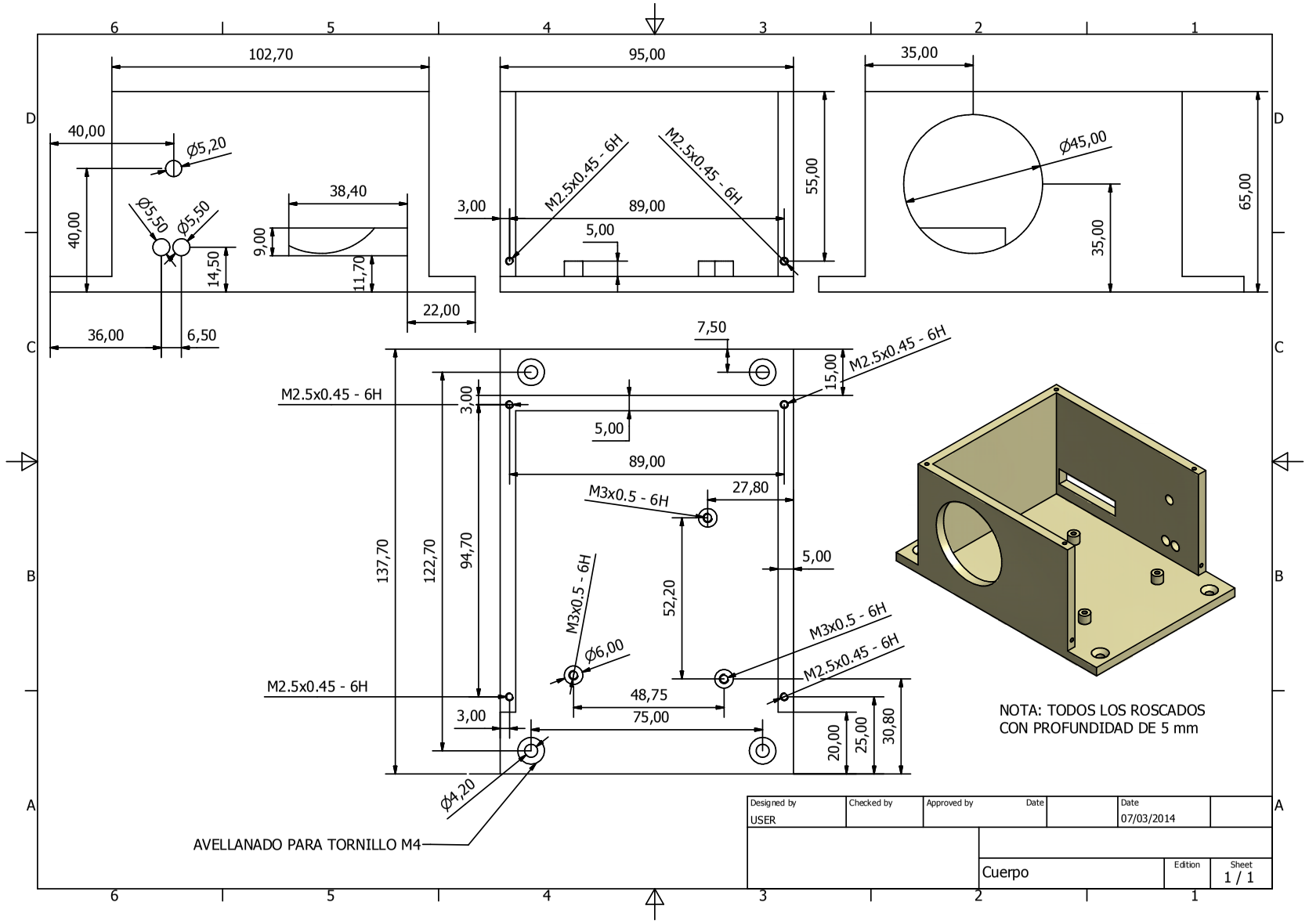
E Esquema electrónico del sistema

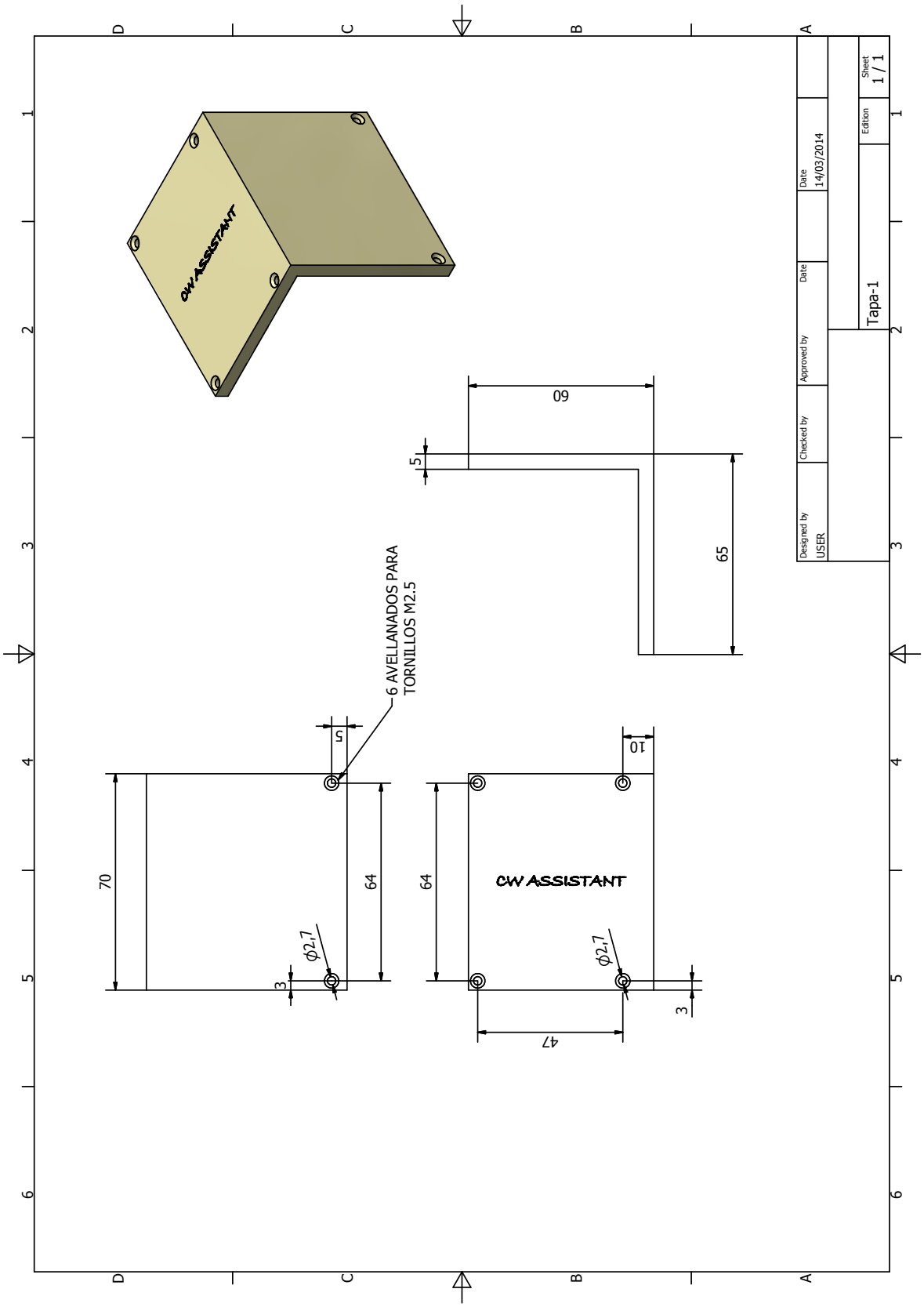


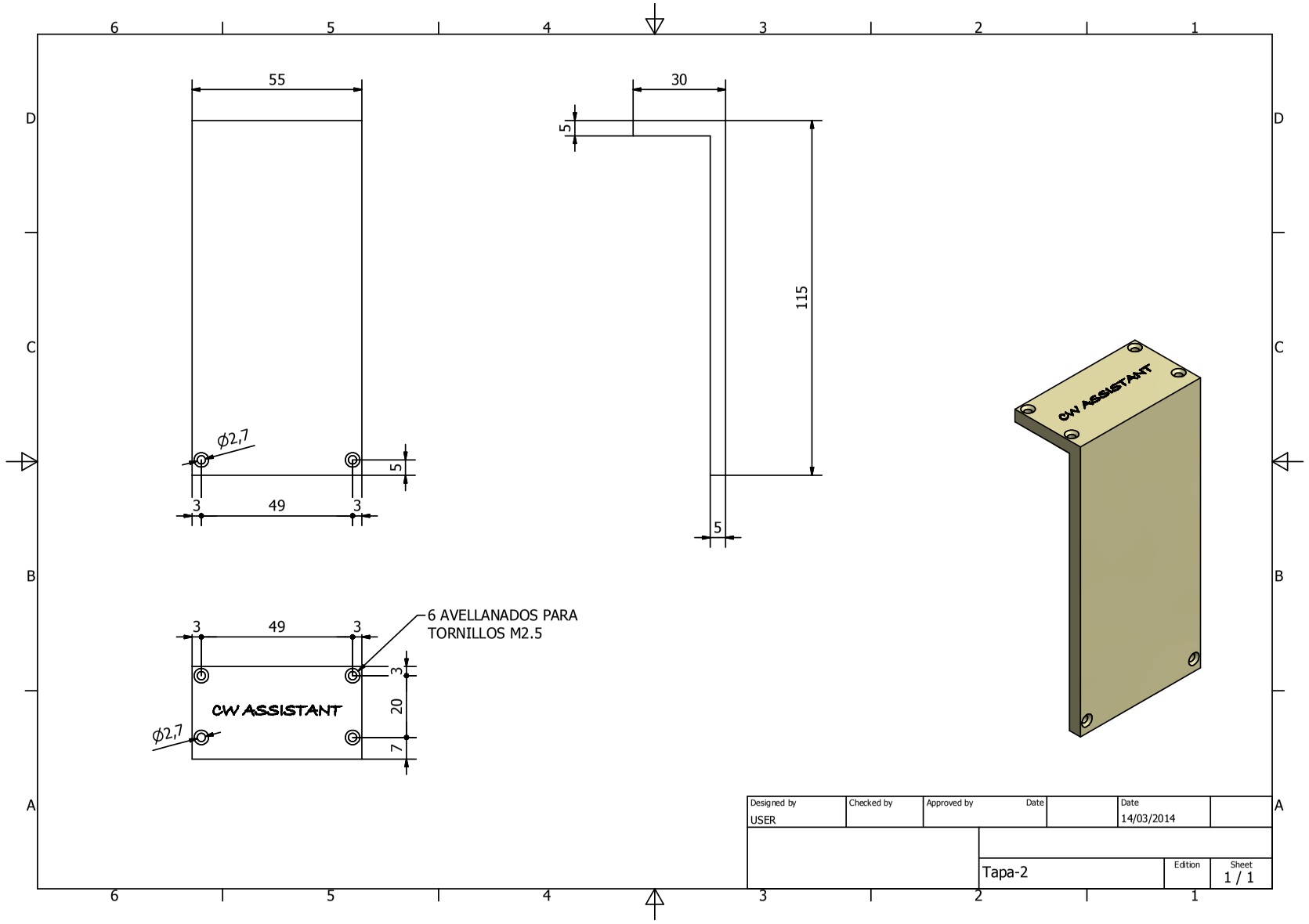
F Planos mecánicos de los módulos electrónicos

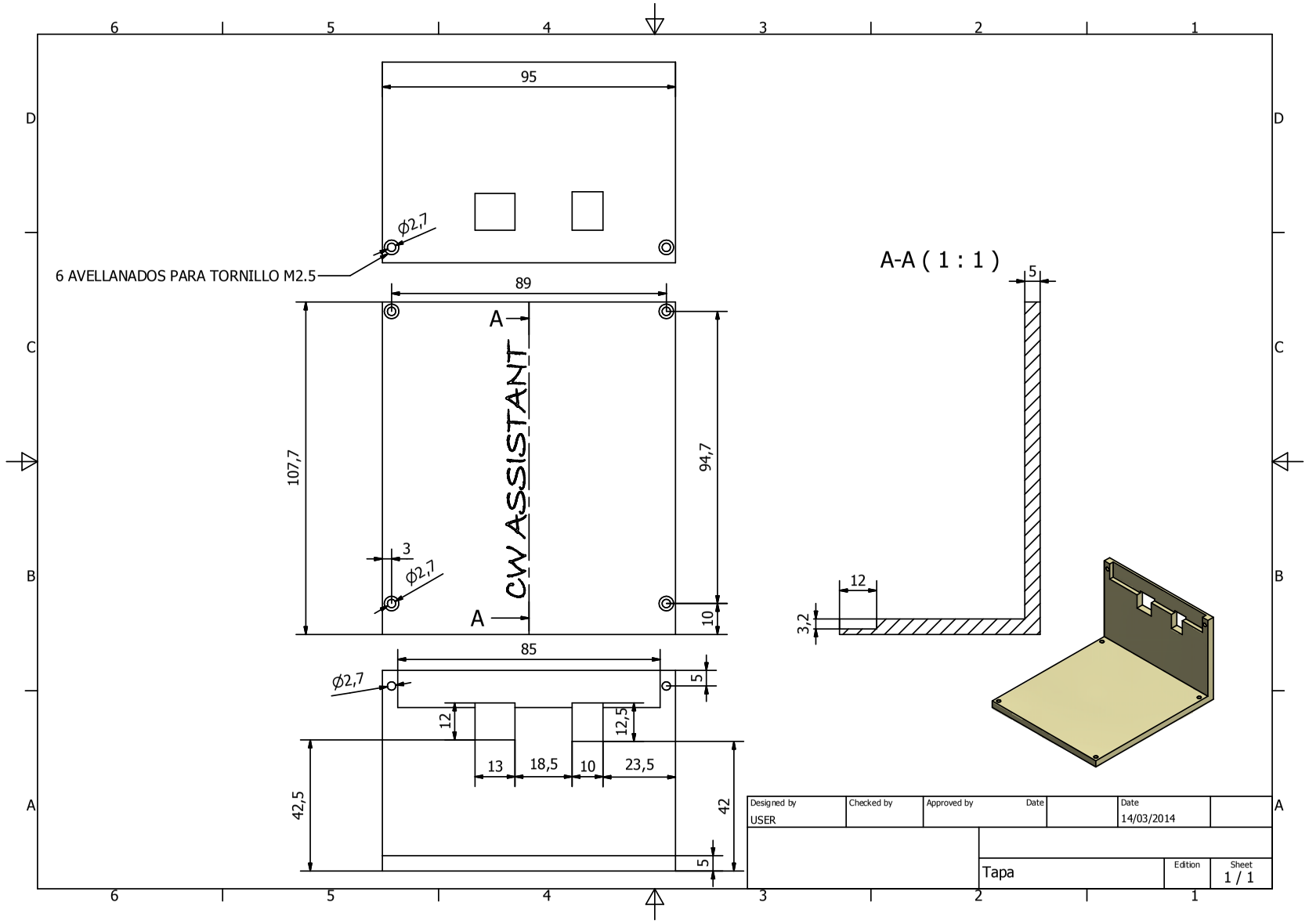



6
5
4
3
2
1
D
C
B
A
5
13
21
8,25
M3x0.5 - 6H
3,25
11,5
9,5
37,5
10
M3x0.5 - 6H
Ø6
27,5
115
M2.5x0.45 - 6H
10
49,5
3
5
20
15
60
29,5
Ø5,5
30
9
11
13
24,5
55
35
10
M2.5x0.45 - 6H
7,5
3
M2.5x0.45 - 6H
M3x0.5 - 6H
17
8,24
15,7
45
60
M3x0.5 - 6H
M2.5x0.45 - 6H
Ø4,2
35,5
M2.5x0.45 - 6H
25
18,5
18,5
AVELLANADOS PARA TORNILLOS M4
49
NOTA: TODOS LOS ROSCADOS
CON PROFUNDIDAD 5 mm
Designed by
USER
Checked by
Approved by
Date
Date
14/03/2014
Cuerpo-2
Edition
Sheet
1 / 1


102,70
95,00
35,00
40,00
Ø5,20
Ø5,50
Ø5,50
38,40
9,00
11,70
14,50
40,00
3,00
M2.5x0.45 - 6H
M2.5x0.45 - 6H
89,00
5,00
55,00
Ø45,00
35,00
65,00
22,00
36,00
6,50
7,50
M2.5x0.45 - 6H
15,00
M2.5x0.45 - 6H
3,00
5,00
89,00
27,80
M3x0.5 - 6H
137,70
122,70
94,70
5,00
52,20
M3x0.5 - 6H
Ø6,00
M3x0.5 - 6H
M2.5x0.45 - 6H
M2.5x0.45 - 6H
48,75
75,00
3,00
20,00
25,00
30,80
Ø4,20
AVELLANADO PARA TORNILLO M4
NOTA: TODOS LOS ROSCADOS
CON PROFUNDIDAD DE 5 mm
Designed by
USER
Checked by
Approved by
Date
Date
07/03/2014
Cuerpo
Edition
Sheet
1 / 1


CW ASSISTANT
60
5
65
6 AVELLANADOS PARA
TORNILLOS M2.5
5
70
64
ϕ2,7
3
10
64
CW ASSISTANT
ϕ2,7
3
47
Designed by
USER
Checked by
Approved by
Date
Date
14/03/2014
Edition
Sheet
1 / 1
Tapa-1


55
30
5
115
5
ϕ2,7
5
3
49
3
6 AVELLANADOS PARA
TORNILLOS M2.5
3
49
3
3
20
7
CW ASSISTANT
ϕ2,7
CW ASSISTANT
Designed by
USER
Checked by
Approved by
Date
Date
14/03/2014
Tapa-2
Edition
Sheet
1 / 1


95
6 AVELLANADOS PARA TORNILLO M2.5
Ø2,7
89
A
CW ASSISTANT
A
107,7
94,7
3
Ø2,7
10
85
Ø2,7
5
12
12,5
13
18,5
10
23,5
42,5
42
5
A-A (1 : 1)
5
12
3,2
Designed by
USER
Checked by
Approved by
Date
Date
14/03/2014
Tapa
Edition
Sheet
1 / 1

PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal Mac Mini (Software incluido)..... 2.000 €
- Alquiler de impresora láser durante 6 meses ..... 50 €
- Material de oficina ..... 150 €
- Electrónica ..... 103€
- Mecánica de la electrónica ..... 40€
- Total de ejecución material ..... 2.343 €

2) Gastos generales

- 16 % sobre Ejecución Material..... 374.88 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material..... 1405.8 €

4) Honorarios Proyecto

- 640 horas a 15 € / hora ..... 9600 €

5) Material fungible

- Gastos de impresión ..... 60 €
- Encuadernación ..... 200 €

6) Subtotal del presupuesto

- Subtotal Presupuesto ..... 13983.68 €

7) I.V.A. aplicable

- 21% Subtotal Presupuesto ..... 2936.6 €

8) Total presupuesto

- Total Presupuesto ..... 16920.25 €

Madrid, Abril de 2014

El Ingeniero Jefe de Proyecto

Fdo.: Daniel Gómez Fernández
Ingeniero Superior de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un sistema de sensores inteligentes para detección de automóviles.

. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.