ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

ÁREA DE MECÁNICA DE MEDIOS CONTINUOS Y TEORÍA DE ESTRUCTURAS

TRABAJO DE FIN DE MÁSTER Nº 18010122

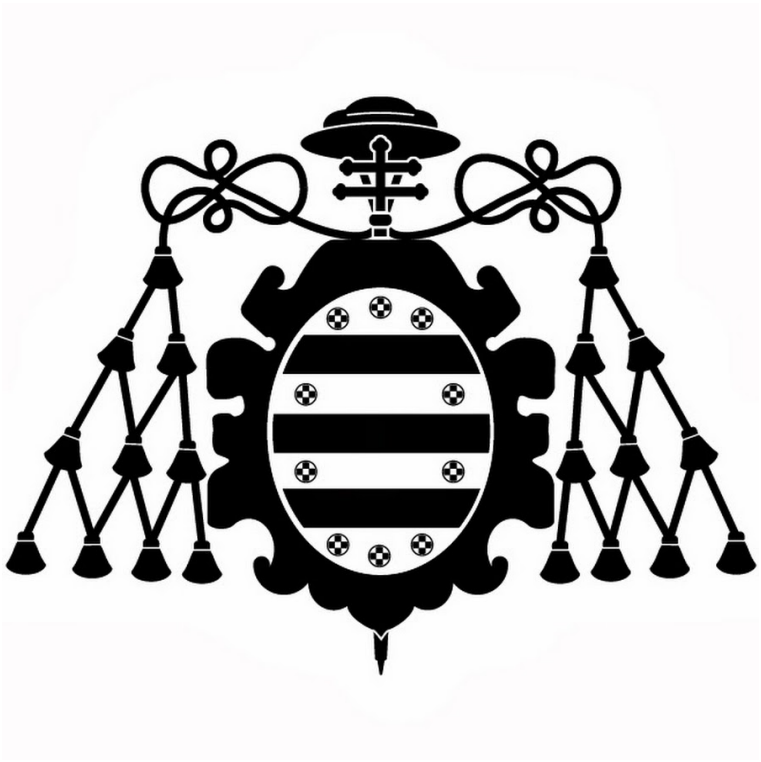
# DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS

ALUMNO: D. CRISTÓBAL GARCÍA CAMOIRA

TUTORA: DOÑA MARIÁN GARCÍA PRIETO

COTUTOR: DON JUÁN CARLOS CAMPO RODRÍGUEZ

FECHA: JULIO DE 2018



# MEMORIA

# Índice del documento MEMORIA

# Índice de figuras



# Índice de tablas

# 1 Glosario y palabras clave

En este apartado se incluyen todas las definiciones y abreviaturas que se utilizaron a lo largo del presente proyecto.

Por tanto, este capítulo no es más que de consulta de posibles palabras o abreviaturas que pueden no ser comprendidas.

- **string:** Matriz de datos, que comúnmente en programación se conoce como array.
- **Arduino:** es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.
- **USB:** Del inglés, Universal Serial Bus. Bus serie universal.
- **Sketch:** Entorno de programación en Arduino.
- **Jumper:** Conector minúsculo que hace posible a unión entre dos pines separados a cierta distancia.
- **EPI:** Escuela Politécnica de Ingeniería.
- **LCD:** Lyquid Crystal Display. Pantalla de cristal liquido.
- **TFM:** Trabajo de fin de máster.
- **Polling:** Operación de consulta constante, generalmente hacia un dispositivo de hardware, para crear una actividad sincrónica sin el uso de interrupciones, aunque también puede suceder lo mismo para recursos de software (p. ej. comprobar continuamente el estado de un pulsador).
- **I2C:** Inter-Integrated Circuit (Inter-Circuitos Integrados).
- **Hyperterminal:** Aplicación de Windows que le permite establecer una comunicación ordenador a ordenador o a cualquier otro dispositivo a través de una conexión telefónica convencional o por puerto serial.
- **Latitud:** Es la distancia angular que existe entre un punto cualquiera y el Ecuador.
- **Longitud:** Es la distancia angular que existe entre un punto cualquiera y el Meridiano de Greenwich.

- **Smartphone:** Es un tipo de teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional.

- **Maestro/esclavo:**es un modelo de comunicación en el que un dispositivo o proceso tiene control unidireccional sobre uno o más dispositivos. En algunos sistemas, se selecciona un maestro a partir de un grupo de dispositivos elegibles, y los otros dispositivos actúan como esclavos.

  Ejemplo: “La configuración maestro / esclavo se utiliza básicamente para el uso compartido de la carga cuando dos motores idénticos conectados a dos impulsiones diferentes se acoplan a una carga común”. Una unidad se define como el amo y se configura para funcionar En el modo de control de velocidad mientras que el otro definido como esclavo está configurado para funcionar en modo de control de par.

# 2 Objeto

El objeto principal del presente proyecto es la realización de la parte electrónica (diseño y montaje) que constituirá la base de un sistema genérico para el control y localización de objetos.

De una forma mas técnica, se podría decir que el objeto principal del presente proyecto consiste en utilizar las tecnologías, GSM, GPS GPRS y Bluetooth para efectuar el control (comunicación, configuración, localización y envío de una notificación ya sea mediante correo electrónico o SMS) sobre un dispositivo genérico, aunque se podría extrapolar a cualquier elemento utilizando la red de datos móviles GPRS como canal de comunicación principal.

El sistema ha de ser capaz de poder tener localizado al objeto y con la opción de informar a través de cualquier tipo de notificación (sms, email...) de cualquier percance que le ocurra al mismo.

La elaboración del presente proyecto permitirá conocer las diferentes formas de comunicación ( medios existentes para el envió de datos: Bluetooth, GPRS, SMS, Correos electrónicos) que existen hoy día, así como el conocimiento del funcionamiento del sistema de posicionamiento global.

Cabe destacar que aunque el objeto principal de la realización del presente documento es el ***"Diseño y construcción de un sistema de localización en tiempo real para objetos"***, también posee fines didácticos, tanto personales para cualquier persona que sienta curiosidad acerca de como implementar de forma practica un sistema que posea estas características, además permitira adquirir conocimientos acerca de cómo controlar un dispositivo desde cualquier punto, mediante la utilización del servicio GPRS y de la tecnología móvil.

Se deja abierta la posibilidad de diseñar sistemas que, basados en el desarrollo del localizador, proporcionen nuevas funcionalidades al mismo.

# 3 Alcance

El presente trabajo ha sido realizado con la intención de ser implementado físicamente, con lo cual el alcance del mismo ha de cubrir todas las fases que se enumeran a continuación:

1. Fase de aprendizaje en la programación de Arduino en nuevas plataformas (eclipse, AVR studio).
2. Fase de aprendizaje en la creación de una pagina web mediante java y PHP, implementación de una base de datos y la utilización de múltiples servicios de Google.
3. Fase de desarrollo e implementación del hardware y software asociado.
4. Fase de pruebas y comprobación del funcionamiento del dispositivo de localización en tiempo real.

En la primera fase, una de las más importantes de todo el proyecto, se dedica al aprendizaje de los diferentes lenguajes que se han utilizado para la elaboración del presente proyecto (lenguaje Arduino ). En esta fase, las pruebas se han realizado e implementado físicamente en la placa Arduino.

En la segunda fase trata de ver como se efectuaría la conexión del dispositivo localizador con internet y la forma de enviar los datos a una web creada para el uso de este dispositivo, para ello se deben adquirir algunos conocimientos de PHP y java, así mismo también se trata la forma de como se utilizan algunas «APIs» de Google (google maps), tratando de incorporarlas a la pagina web.

La tercera fase es la que le da forma al proyecto, el dispositivo debe incluir múltiples opciones de configuración desde un Smartphone mediante Bluetooth. Ademas, la configuración del dispositivo ha de realizarse de forma intuitiva, y que, por supuesto realice las funciones que se indican en el capítulo (5) etc.

En la cuarta y última fase denominada como fase de pruebas es en la que se verificara que el prototipo realizado funcione de forma adecuada y de acuerdo a los parámetros que se hayan configurado previamente. Algunas de las pruebas que serán realizadas son las siguientes:

1. Comprobar que en el menú no exista ninguna errata en la sintaxis de programación a la vista del funcionamiento del dispositivo.
2. Comprobar que el dispositivo de localización sea capaz de enviar una trama de información a través de GPRS a un servidor con el contenido de la hora, el nombre o identificador y posición en que se encuentra.
3. Verificar si la posición que adquiere el GPS que incorpora el dispositivo localizador se corresponde con la real, esta prueba se realiza introduciendo en Google Maps las coordenadas recibidas en

el servidor enviadas previamente desde el dispositivo teniendo en cuenta que las coordenadas solo nos indican los grados y los minutos (formato Dm “degrees, minutes” ). Se puede utilizar el apartado de cálculos (capitulo 11.3) para convertir las coordenadas a un formato correcto (grados, minutos, segundos y dirección.

4. Comprobar la comunicación entre el Smartphone y el dispositivo, es decir, comprobar que tras haber sido presionado el botón de emergencia del prototipo envíe una notificación al smartphone, ya sea un correo electrónico o un SMS a un numero de teléfono o dirección de correo electrónico que haya sido configurado previamente.

# 4 Antecedentes

El sistema para el control de objetos incorpora muchas de las características que poseen el resto de dispositivos similares, destacando la tecnología GPS para detectar la posición del objeto al que va adherido, la tecnología GPRS para enviar los datos de la posición a un servidor para poder almacenarlos y visualizarlos en un mapa.Dispone también de un sistema Bluetooth y su uso sera única y exclusivamente para la configuración del aparato a través de la aplicación móvil.

Existen multitud de dispositivos que tratan el tema de geoposicionamiento y envían esa información a un servidor web o bien por SMS a un numero de teléfono. La gran mayoría de estos dispositivos no disponen de una aplicación de smartphone para configurarlo y, por regla general no es sencillo configurar los dispositivos que dispone el mercado actual para tal efecto. Aunque no se ha desarrollado en este trabajo una aplicación para Android para el mismo, resultaría bastante sencillo realizarla, y, configurar el dispositivo localizador mediante bluetooth resultaría muy cómodo.

## 4.1. Conclusiones

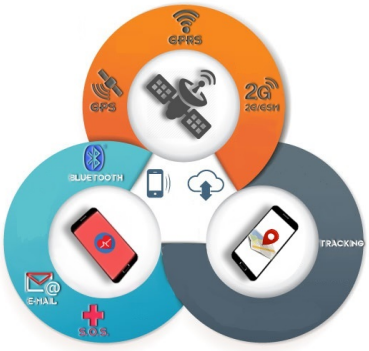



**Figura 4.1.0.1 –** Tecnologías incorporadas por nuestro sistema de localización en tiempo real

Aunque las características que se desean incorpora a este innovador sistema para el control de objetos en tiempo real son significativamente mejores que los que ofrecen la mayoría de los diseños existentes en el mercado, no se debe olvidar que el mercado chino lleva ofreciendo durante años atrás muchas de las tecnologías que van a ser aplicadas en este sistema, entre ellas GPS, GPRS, Bluetooth,

Se realiza cierto hincapié en el mercado chino suele ser el que rompe el mercado con precios sorprendentemente bajos, por ello es necesario ser cauto y estar actualizándose constantemente en las tecnologías aplicadas para la elaboración de este tipo de sistemas de control de objetos, no obstante se

calcula que el diseño no sobrepase los 60€, un precio que se considera razonable por todas las características que se han mencionado anteriormente. Aunque este precio podría oscilar dependiendo de la evolución del mercado para ello se debería tener en cuenta el volumen de ventas del dispositivo, el precio de venta de dispositivos similares etc...

# 5 Requisitos de diseño

En este apartado se abarcarán los requisitos de diseño impuestos de forma previa a la realización del proyecto. Los requisitos que se imponen le afectan a las distintas partes del proyecto y, de alguna forma, concreta la forma de realización del presente trabajo en cuestiones de software, hardware y ergonomía. Estos se exponen a continuación:

1. Se empleara el software y hardware acorde con las necesidades del dispositivo. Se recomienda la utilización de la plataforma Arduino y su hardware debido a la gran cantidad de información que existe en la web, aunque puede ser utilizada cualquier otra.

2. En la medida de lo posible el software asociado sea portable para otras placas del mismo fabricante.

3. El sistema debe ser capaz de comunicarse con cualquier servidor que disponga de una ip fija conocida por el usuario, es decir, la persona que va a controlar el objeto al que se la va a hacer el seguimiento.

4. El contenido de la información enviada al servidor web es la siguiente:

    - Hora.
    - Fecha.
    - Latitud y longitud de la Posición actual.
    - Velocidad.
    - Numero de Satélites.
    - Altitud.

5. El diseño hardware del dispositivo localizador ha de ser implementada de forma que pueda ser fácilmente transportado por una persona (obviamente este es un prototipo y por tanto se permite un tamaño mayor).

6. El dispositivo de control ha de disponer de software asociado como por ejemplo una aplicación en Android instalada en un smartphone de los actuales que le permita poder ser configurado de forma sencilla.

7. La aplicación a desarrollar ha de ser lo más simple posible para poder ser utilizada y entendida por cualquier tipo de usuario.

8. El dispositivo tiene que poder funcionar de forma autónoma, sin necesidad de conectarlo a un ordenador o a una fuente de alimentación.

9. El dispositivo dispondrá de bluetooth para que el usuario pueda configurar los parámetros de forma sencilla.

10. El dispositivo dispondrá de GPS para la obtención de los datos de la posición.

11. El dispositivo dispondrá de GPRS/GSM para poder comunicarse a través de internet, poder enviar correos electrónicos, sms...

12. El dispositivo ha de disponer de como mínimo 2 pulsadores, uno para activar la configuración del dispositivo y el otro para dar a conocer la posición a una segunda persona en caso de cualquier imprevisto mediante el envío de un correo electrónico o un E-mail.

13. Complementando a los pulsadores, el dispositivo localizador ha de poseer tres leds que indicaran del estado del dispositivo.

    - Uno de los leds indicara si el dispositivo ha establecido conexión con internet.
    - Otro led indicara si el dispositivo esta enviando un E-mail.
    - El ultimo led indicara si el dispositivo esta o no en modo configuración.

14. El dispositivo ha de poseer de dos modos de funcionamiento a los que se denominan (modo normal) y (modo auxilio):

    - Durante el **modo de funcionamiento normal**, el dispositivo enviara los datos referentes a su posición durante un intervalo de entre diez y veinte segundos.
    - El **modo auxilio** ha sido implementado por si el dispositivo es transportado por una persona. El dispositivo de seguimiento en tiempo real posee un botón, que sera presionado en caso de cualquier percance o imprevisto, en este instante el dispositivo enviara una notificación a través de SMS o bien mediante un correo electrónico a la persona que esta visualizando el seguimiento indicando que la persona ha sufrido un percance, el SMS o el correo electrónico enviado ha de contener como mínimo la posición y hora en la que se encontraba esa persona en el momento de pulsar dicho botón.

15. No se requiere la utilización de hardware especifico para el envío y recepción de de datos referentes a la posición del dispositivo, así como para la configuración del mismo por el usuario.

16. Para la configuración del dispositivo el usuario deberá incluir la siguiente información:

    - Nombre del usuario.
    - Pin de la tarjeta SIM introducida en el dispositivo de localización.
    - Contraseña de la cuenta de E-mail.
    - Servidor SMTP utilizado para enviar el E-mail.
    - Puerto que utiliza el servidor SMTP para enviar el E-mail.

- Dirección E-mail del usuario.
- Nombre de la persona que desea recibir el E-mail.
- Dirección de E-mail de la persona a la que se le desea enviar el E-mail.
- Numero de teléfono al cual enviara el SMS de aviso (en el caso de que se desee enviar un SMS y no un E-mail).
- Coordenadas de la latitud y la longitud del punto de partida.
- Datos sobre la red Gprs tales como el APN, usuario y contraseña (estos dependen del operador de telefonía a utilizar).
- Ip del servidor (la ip es recomendable que sea fija, de no ser así, hay que introducirla cada vez que se requiera utilizar el dispositivo).
- Numero de teléfono de la persona a la que se le desea llamar o enviar un SMS (en su caso).
- Nombre del dispositivo bluetooth.
- Contraseña de emparejamiento del dispositivo bluetooth.

# 6 Análisis de las soluciones

En este capitulo se verán qué aspectos han sido determinantes para seleccionar cada uno de los dispositivos que se han elegido para conformar la parte electrónica del sistema de localización de objetos y aquellas características fundamentales y requisitos de cada uno. Para ello se parte de las premisas que se han realizado en el capitulo de requisitos de diseño.

Lo anteriormente citado definirá las entradas y salidas necesarias para que el proyecto funcione, es decir, cuantos pines digitales se necesitan (entradas/salidas ), cuantas entradas analógicas, interrupciones, entradas para la comunicación serie... etc.

Este capitulo concluirá con la selección del Arduino en función de las entradas y salidas que necesitemos para la realización del proyecto.

## 6.1. Introducción

Los requisitos de diseño impuestos en el capítulo [5] ya restringen bastante las soluciones a las que se pueden optar para la realización de este proyecto.

Uno de los requisitos menciona que el sistema de control en tiempo real se realice mediante la interfaz de Arduino, y no con cualquier otro micro-controlador, ya sea de la familia PIC de Microchip o STmicroelectronics.

Existen en el mercado diferentes placas de Arduino que se adaptan a las necesidades del presente proyecto, algunas con un microcontrolador más potente que otras, también los requisitos de diseño imponen una condición clave y es que la aparamenta electrónica utilizada ocupe un espacio lo mas reducido posible.

La selección del microcontrolador se realizara en el apartado (6.6) según los requisitos de diseño anteriormente descritos.

En el capitulo de requisitos de diseño [5] se indica que:

1. El dispositivo dispondrá de bluetooth para que el usuario pueda configurar los parámetros de forma sencilla.

2. El dispositivo dispondrá de GPS para la obtención de los datos de la posición.

3. El dispositivo dispondrá de GPRS/GSM para poder comunicarse a través de internet, poder enviar correos electrónicos, sms...

4. El dispositivo ha de disponer de como mínimo 2 pulsadores, uno para activar la configuración del dispositivo y el otro para dar a conocer la posición a una segunda persona en caso de cualquier

imprevisto mediante el envío de un correo electrónico o un E-mail.

5. Complementando a los pulsadores, el dispositivo localizador ha de poseer tres leds que indicaran del estado del dispositivo.

Estos requisitos son fundamentales para saber el número de entradas y salidas que se necesitan y poder decidir entre cualquiera de los Arduinos disponibles en el mercado.

## 6.2. Selección del modulo de comunicaciones GPRS-GSM

Para satisfacer la primera premisa,“comunicarse con cualquier servidor”, existen en el mercado multitud de módulos GPRS-GSM que realizan estas funciones que contienen generalmente el chip SIM900X o SIM800X, estos módulos o placas generalmente son fabricadas para insertar en Arduino UNO o MEGA y muchos ademas contienen GPS, este tipo de placas cumplen las especificaciones funcionales , pues al final lo que se necesita es enviar datos de posicionamiento a través de GPRS y ofrecen también la posibilidad de enviar SMS, el principal problema es que muchos de estos módulos han sido diseñados para ser introducidos directamente en Arduino UNO o Arduino MEGA, y la necesidad de que estos módulos sean pequeños (dado que son casi como la palma de una mano), y hacer que el sistema sea portátil, no se va recurrir a este tipo de placas, ademas de que su precio en el mercado es relativamente elevado incrementando así los costes del sistema de localización tanto de fabricación como de venta al publico.

(a) GPS/GPRS/GSM shield de DFRobot

(b) Arduino GSM Shield V.2

(c) Adafruit FONA 808 Shield

**Figura 6.2.0.1 –** Módulos GPS-GPRS-GSM para Arduino.

La placa FONA 808 Shield es bastante pequeña y se adecua a las especificaciones, pues ademas de poseer GPRS y GSM también posee GPS, sus excepcionales especificaciones podrían colocarla como una de las opciones mas factibles y de esta forma ya no se tendría que colocar un GPS a mayores, el principal problema es su coste, que sigue siendo relativamente elevado, pues en el mercado ronda los 50-60 €, con lo cual obliga al desarrollo de este proyecto a explorar otras alternativas.

Habiendo realizado una pequeña investigación se ha llegado a la conclusión de que no es necesario encontrar una placa que reúna todo junto, quizá para el caso de este proyecto sea mejor buscar varias placas pequeñas y que cada una se encargue de su función.

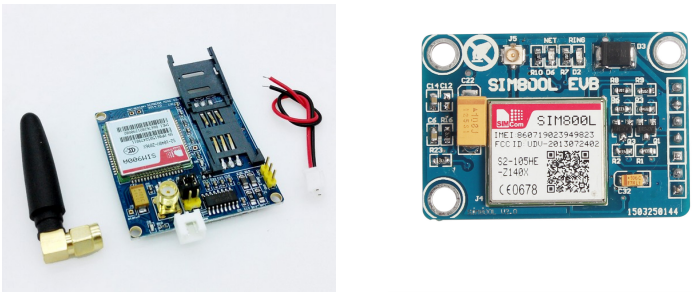
Indagando y consultando en la red acerca de módulos GSM-GPRS se han encontrado mas de los que se esperaba, todos cumplen las funcionalidades requeridas, y la verdad son realmente pequeños, aunque pueden presentar problemas en su funcionamiento y la mayoría de estos problemas se resuelven

instalando en estos dispositivos el firmware adecuado o bien alimentándolos de forma correcta, siendo en el capitulo “Guía de implementación” que se encuentra en la sección 4.3.1 del pliego de condiciones, donde se especifican los detalles.

Se han probado 2 módulos en concreto y ambos funcionan correctamente. Se ha escogido el modelo Sim800L porque no es necesario cambiarle el firmware para que funcione en Europa y ademas es mas pequeño (sección 4.3.1 del pliego de condiciones) estos se pueden ver representados en la figura 6.2.0.2, hay que tener mucho cuidado a la hora de seleccionar este tipo de dispositivos, puesto que no todos funcionan en todos los países, dado que las frecuencias de las comunicaciones cambian, las características del chip escogido son las siguientes:

- Módulo cuatribanda GSM de 850 / 900 / 1800 / 1900 Mhz
- GPRS multi slot clase 12/10 estación móvil clase B
- Control mediante comandos AT
- Bajo consumo de corriente: 15 mA en modo sleep
- Potencia de transmisión 2W @ 850 / 900 Mhz
- Rango de tensión en la alimentación 3.4  4.4V
- Temperatura de trabajo:-40℃ 85℃

El resto de características de este chip pueden seer consultadas en la pagina web del fabricante simcomm2m.




(a) Sim900A mini v3.8.2 (b) Sim800L evb

**Figura 6.2.0.2 –** Módulos GPRS-GSM para Arduino

Este dispositivo (Sim800L) posee 4 pines, 2 de alimentación (GND y VCC) y 2 para la comunicación serie RX y TX ( salidas que corresponden a niveles TTL de tensión).

## 6.3. Selección del modulo GPS

Para la elección del GPS, se ha tenido en cuenta la premisa de que sea lo mas pequeño posible. Para la elección del GPS se ha tenido en cuenta el sistema de posicionamiento que utilizan algunos

drones, basados en el chip de Ublox-NEo-8M, y similares (algunas de las imágenes de los diferentes tipos existentes se muestran a continuación en la figura **??**.

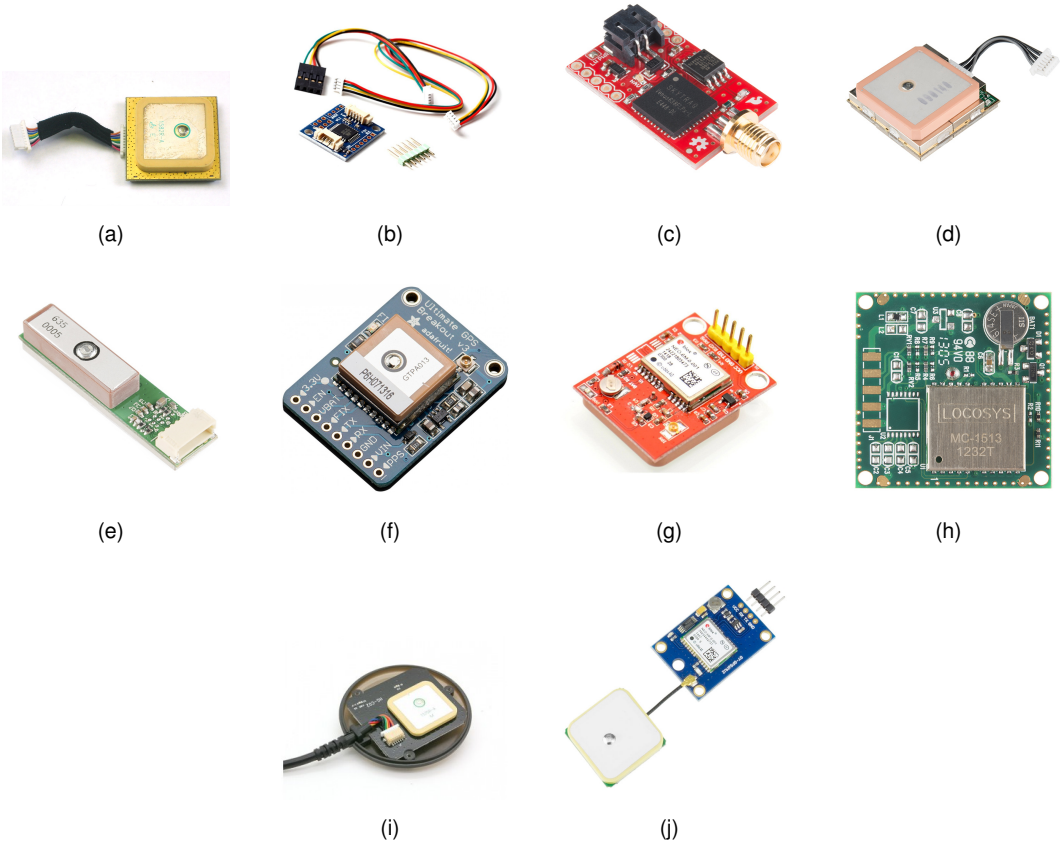



**Figura 6.3.0.1 –** Módulos GPS para Arduino
(a).32 Channel San Jose Navigation GPS 5Hz Receiver with Antenna(24$)
(b).I2C-GPS NAV navigation Navigation Module GPS board for CRIUS MultiWii MWC(8$)
(c).SparkFun Venus GPS Logger - SMA Connector(60$)
(d).GPS Receiver - EM-506(40$)
(e).GPS Receiver - GP-635T(40$)
(f).Adafruit Ultimate GPS Breakout - 66 channel w/10 Hz updates - Version 3(40$)
(g).JBtek Raspberry Pi GPS Module with U-BLOX NEO-6M Modular and High-Performance Ceramic Antenna(30$)
(h).GPS Receiver - LS20031 5Hz (60$)
(i).u-blox NEO-7 GPS module by 3DR(90$)
(j)UBLOX-NEO 6M (7$)

De entre todos estos módulos se va a elegir el mas económico el designado en la imagen anterior 6.3.0.1 (j)UBLOX-NEO 6M, dado que este modulo comete un error máximo de aproximadamente 10-15 metros sobre la posición real del objeto y es capaz de adquirir datos con una frecuencia de 10Hz, mas que suficiente para cumplir con el objeto del proyecto, que es calcular la posición de una persona u objeto que porte el dispositivo y que no viajara a gran velocidad a menos que se desplace un un vehículo.

Este modulo posee 4 pines, 2 para alimentación (GND y VCC) y 2 para comunicación serie (RX y TX).

## 6.4. Selección del modulo Bluetooth

Para la configuración del sistema electrónico se necesita una interfaz que interaccione con el usuario y que sea simple y fácilmente comprensible. Se han pensado dos formas que se describen a continuación.

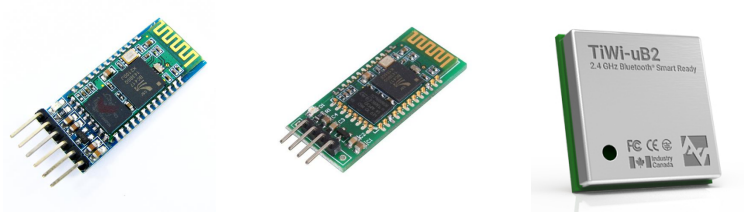
1. Que el usuario conecte un teclado matricial al dispositivo de localización y una pantalla LCD de 16x2 lineas para su primera configuración, y que se extraigan una vez configurado el mismo. Para la realización de esta idea se ha pensado que, para consumir el menor numero de pines en nuestro Arduino, ambos elementos serán controlados por el puerto I2C.

2. Que el usuario a través de una aplicación móvil introduzca los parámetros necesarios para que el dispositivo funcione de forma correcta, comunicándose con éste mismo a través de bluetooth.

Se ha decidido escoger, sin lugar a dudas la segunda opción, ahora queda escoger el dispositivo bluetooth que sera incorporado a nuestro sistema.

Indagando en la red se ha encontrado mucha información, como siempre, acerca de este tipo de dispositivos. Existen muchos tipos de módulos bluetooth, la gran mayoría se diferencian en el alcance, el tipo de versión de bluetooth, la tasa de transferencia de datos y demás. Los datos que se se van a transferir van a ser cadenas de caracteres que serán enviadas desde el teléfono móvil al dispositivo y viceversa, no se va a transmitir audio ni nada por el estilo. Por otra parte, no es necesario que el sistema a desarrollar en el presente proyecto pueda ser detectado por un dispositivo a 100 metros de distancia (que es mas o menos el alcance actual de este tipo de dispositivos), con que se pueda tener el sistema de localización a unos dos o tres metros de distancia del teléfono para su configuración es suficiente.

A continuación se puede ver una imagen de los diferentes módulos bluetooth que existen en el mercado.




(a) B-HC-05 (b) B-HC-06 (c) B-TiWi-uB2

**Figura 6.4.0.1 –** Módulos Bluetooth para Arduino

Por todo esto se ha decidido utilizar en el prototipo a desarrollar el modulo Bluetooth HC-05, se podría utilizar el modulo bluetooth HC-06 perfectamente para el prototipo, la diferencia esta en que el modulo HC-05 puede comportarse como maestro y esclavo (ver el apartado 1), posee doble configuración, el modulo HC-06 su configuración es en modo esclavo, por tanto el modulo HC-05 puede ofrecer alguna prestación mas que aun no se ha estudiado en profundidad, y dado que todavía el proyecto se encuentra en la fase preliminar, y que no hay prácticamente diferencia de coste entre ambos módulos, se asegura que este módulo va a funcionar correctamente, el resto de funcionalidades son idénticas que las del modulo HC-06.

En la figura 6.4.0.1 se observan los dos módulos anteriormente citados y un ultimo, en este caso, es un chip que hace la función de Bluettooth, existen multitud de chips como este con la función de Bluetooth, que serán mas baratos o mas caros dependiendo del fabricante, y especificaciones de las que disponga. Este tipo de chips Bluetooth no se han tenido en cuenta para el diseño del prototipo debido a que son difíciles de manipular y están preparados para ser soldados en una placa de circuito impreso, no obstante si se pretende en un futuro la comercialización de este sistema, seria necesario tener en cuenta este tipo de chips e implementarlos en una placa junto al resto de componentes tratando de minimizar las dimensiones del circuito.

## 6.5. Selección del circuito para acoplar los pulsadores

Esta parte es bastante importante, pues es absolutamente necesario que al presionar un botón el sistema responda con la orden asignada para el mismo.

El numero de botones que se necesitan (tal y como se observa en el apartado 5) serán 2 de momento, uno para avisar de la existencia de algún contratiempo y otro para encender/apagar el Bluetooth y de esta forma consumir menos.

Conectar los botones simplemente a as entradas digitales del microcontrolador a través de un divisor de tensión y gestionarlos a través de la técnica “polling” un método que todavía se sigue utilizando pero que es poco eficiente, tanto a nivel de consumo energético del microcontrolador como de recursos del micro, dado que se estaria ocupando una entrada por cada pulsador, y es necesario ocupar el mínimo numero de entradas/salidas digitales, dado que todavía no se conoce que recursos adicionales van a ser utilizados para el sistema de localización.

Esta técnica requiere una frecuencia de muestreo del pulsador razonable, podría ocurrir que el sistema no reaccione a la pulsación si no se muestrea con una frecuencia elevada.

Son tres las opciones que ofrecen solucion al problema anteriormente citado y se comentan a continuación:

1. **OPCION1:**Que el microcontrolador posea 2 interrupciones externas en los pines digitales de esta forma el sistema no tendría que muestrear cada X periodo de tiempo el estado de esas entradas, puesto que mediante interrupciones externas, cualquier cambio de estado (transición 0-1 ó 1-0) en el pin al que se tenga conectado el pulsador, el sistema interrumpirá su proceso normal de funcionamiento para realizar el subproceso asociado a la interrupción del pulsador asociado.

   Si se poseen suficientes salidas del microcontrolador con interrupciones externas, esta es la mejor opción, puesto que no seria necesario hardware adicional para el desarrollo del sistema de localización.

   Si los recursos del microcontrolador (en cuanto a entradas y salidas) son muy limitados, la idea de conectar cada pulsador a un pin del Arduino no es la optima, en este caso la opción numero 3 seria la correcta.

2. **OPCION2:**Lo que aquí se va a desarrollar es la idea de poder conectar múltiples pulsadores pero haciendo uso de un solo pin del Arduino, pero realizando una conversión Analógico a Digital. Para

ello es necesario conseguir que al pulsar cada una de las teclas se obtenga una tensión distinta en una única línea. Leyendo esta tensión con el conversor AD del Arduino se puede llegar a saber qué tecla es la que se ha pulsado. Aunque requiere la técnica “Polling” para leer el estado del conversor analógico digital cada X tiempo, esta solución es viable, dado que el numero de recursos que consume del microcontrolador es mínimo, pues solo consume una entrada analógica del Arduino y se pueden conectar todos los pulsadores que se necesiten. En la en la sección 2.4 del pliego de condiciones denominado se muestra información detallada acerca de este tipo de conexionado representándose también de forma esquemática.

3. **OPCION3:**Es quizá una opción bastante mas complicada que la anterior, no obstante esta llena de ventajas. Se trata de la utilización de un modulo I2C, este tipo de módulos se utilizan para expandir las entradas/salidas digitales de nuestro microcontrolador (muy utilizado Con LCDs).Este tipo de módulos están basados en el chip PCF8574 ver imagen(6.5.0.1), que posee una interrupción en el caso de cambiar de estado cualquiera de sus entradas/salidas, este recurso es ideal para el conexionado de pulsadores, puesto que ademas de poder conectar numerosos pulsadores, pueden ser gestionados de forma inmediata mediante la utilización de dicha interrupción al producirse un flanco de subida o bajada en cualquiera de los pines.De esta forma el microcontrolador Arduino no tiene que muestrear el estado de una salida cada x tiempo, arriesgándose a que el sistema no reaccione ante esa pulsación si la frecuencia de muestreo es baja, puesto que automáticamente se ejecutaría un subproceso que detectaría que botón ha sido presionado y la acción asociada al mismo, solo con la utilización de un pin del Arduino, en este caso un pin digital que posea una interrupción externa. Como ya se ha dicho anteriormente, se puede observar la forma de conexionado de esta opción así como el programa asociado al control y su esquema en la sección 2.4.2 del pliego de condiciones.

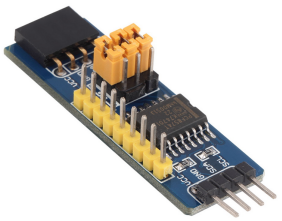


**Figura 6.5.0.1 –** Modulo expansor de entradas/salidas con chip PCF8574

Con todo lo que se ha dicho anteriormente, y aunque la opción tres parezca la optima, se escoge la opción 1 puesto que se va a disponer un microcontrolador que posea los recursos necesarios que se presentan en este proyecto para que no sea necesaria la utilización de un modulo a mayores tal y como se explica en la tercera opción.

## 6.6. Selección del microcontrolador Arduino

El número de recursos que se necesitan desde el punto de vista del microcontrolador a utilizar, y sabiendo que se necesita el mayor numero de pines digitales libres posible son:

1. Tres puertos de comunicación serie, uno para Recibir datos de GPS (RX1,TX1) otro para transmitir/recibir los datos de configuración mediante Bluetooth (RX2,TX2) y un ultimo puerto serie para transmitir los datos de posición al servidor (RX3,TX3).
2. Dos pines digitales de Arduino actuando como entrada dedicado a interrupciones externas.
3. Tres pines digitales para indicadores led acerca de el estado del dispositivo, es decir, si esta enviando un SMS/E-MAIL, o si esta encendido el bluetooth o si el dispositivo esta conectado a la red GPRS.
4. Otro pin digital que activa o desactiva el modulo bluetooth.
5. Un pin digital mas para habilitar/deshabilitar el modulo GPRS.

En la tabla (6.6.0.2) se pueden ver las diferentes características que ofrecen las placas citadas.

Arduino NANO aparentemente cumple las expectativas, dispone de 2 interrupciones externas y de comunicación I2C, pero existe un problema, no dispone de 3 puertos de comunicación serie.

Arduino UNO es similar al anterior y tampoco dispone de puertos de comunicación suficientes.

Si se desea una solución que realmente sea efectiva, se necesita un microcontrolador que ofrezca como mínimo 3 puertos de comunicación serie, este microcontrolador es el ATMEGA2560, que contiene el Arduino MEGA.

En el capitulo [5] se comentaba que el sistema debe ser lo mas diminuto posible para hacer que este sistema sea una solución portable, seleccionando un arduino MEGA se viola ese requisito, pero sin embargo se mejoraría el funcionamiento del sistema. Poniendo estos dos aspectos en una balanza, se considera mas factible que nuestro sistema funcione correctamente en lugar de tener un sistema mucho mas compacto y un funcionamiento mediocre.

De todas formas, Arduino MEGA es una placa de desarrollo con software libre y sus planos están descritos por toda la web, lo único que se necesita de este Arduino son los 4 puertos serie (los 3 que hacen falta para el GPS, Modem SIM800L, y el bluetooth) y el ultimo puerto serie seria utilizado para la depuración del código, y 7 pines adicionales, el resto de los pines de los que dispone no serian necesarios, con esto se quiere decir que se puede miniaturizar el circuito dejando accesibles unicamente aquellos recursos que son estrictamente necesarios del microcontrolador ATMEL2550, pudiendo reducir el tamaño de a placa en mas de un 50 %.

- Los pines digitales 16 y 17 estarán dedicados para la comunicación serie con el modulo GPS UBLOX NEO 6M.
- Los pines digitales 14 y 15 estarán dedicados para la comunicación serie con el modulo Bluetooth.
- Los pines digitales 18 y 19 estarán dedicados para la comunicación serie con el modulo GPRS-GSM SIM 800L.

Para los pulsadores se utilizaran los pines asociados a una interrupción externa, configurados como entradas están los pines 2 y 3, conectándose al pin 2 el pulsador que activa la función del bluetooth y al pin 3 el pulsador que activa la función del aviso de emergencia (SMS/E-MAIL).

El ridículo consumo del modulo utilizado en nuestro proyecto permite ser alimentado desde un pin digital, de esta forma podremos encender y apagar el modulo Bluetooth activando y desactivando este pin, este pin en cuestión estará configurado como salida en Arduino MEGA y es el pin 5.

Por ultimo existe un pin digital mas, configurado como salida, este pin es el pin 7 de Arduino, encargado de resetear el modulo GPRS en caso de que sea necesario.

A continuación se muestra una tabla resumen con los recursos que se van a utilizar de Arduino MEGA para implementar en el presente proyecto.

| Pines digitales Arduino | Conectado a... | Pines analógicos Arduino | Conectado a... |
|---|---|---|---|
| 19 | TX sim 800L | A4(SDA) | SDA LCD (Opcional) |
| 18 | RX sim 800L | A5(SCL) | SCL LCD (Opcional) |
| 17 | TX GPS | | |
| 16 | RX GPS | | |
| 15 | TX Bluetooth | | |
| 14 | RX Bluetooth | | |
| D7 | RESET SIM800L | | |
| D2 | Pulsador Bluetooth | | |
| D3 | Pulsador E-mail | | |
| D5 | VDD bluetooth | | |
| 32 | Led Bluetooth | | |
| 36 | Led E-mail | | |
| 40 | Led GPRS | | |

**Tabla 6.6.0.1 –** Recursos de Arduino MEGA utilizados en el proyecto

| |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| Modelo | Pro Mini | Nano | Uno | Mega/ Mega 2560 | Leonardo | Micro | Due | Netduino 2 | Stellaris Launchpad LM4F120 | Raspberry Pi |
| Fabricante | Arduino | Arduino | Arduino | Arduino | Arduino | Arduino | Arduino | Netduino | Texas instruments | Fundación Raspberry |
| Micro controlador | AVR atmega 168 ó 328 8bits | AVR atmega 168 ó 328 8bits | AVR atmega 328 8bits | AVR atmega 2560 8bits | AVR atmega 32u4 8bits | AVR atmega 32u4 8bits | ARM SAM3X8E Cortex M3 32 bits | ARM STM32f2 Cortex M3 32 bits | ARM LM4F120H5QR Cortex M4 32 bits | ARM broadcom BCM2835 |
| Frecuencia | 16Mhz | 16Mhz | 16Mhz | 16Mhz | 16Mhz | 16Mhz | 84Mhz | 120Mhz | 80Mhz | 700Mhz |
| Mem RAM | 2kib | 2kib | 2kib | 8kib | 2,5kib | 2,5kib | 96kib (64+32 Kib) | 60kib | 32kib | 512kib |
| Mem EEPROM | 1kib | 1kib | 1kib | 4kib | 1kib | 1kib | 0 | 0 | - | - |
| Mem FLASH | 16 ó 32 KiB | 16 ó 32 KiB | 32 KiB | 128 ó 256 KiB | 32 KiB | 32 KiB | 512 KiB | 192 KiB | 256 KiB | - |
| Pines digitales E/S | 14/14 | 14/14 | 14/14 | 54/54 | 20/20 | 20/20 | 54/54 | 20/20 | 43/43 | 8/8 |

**Tabla 6.6.0.2 –** Tabla comparativa sobre diferentes tipos de Arduinos en el mercado

| |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| Modelo | Pro Mini | Nano | Uno | Mega/ Mega 2560 | Leonardo | Micro | Due | Netduino 2 | Stellaris Launchpad LM4F120 | Raspberry Pi |
| Tensión/ corriente pines digitales | 3,3 ó 5V 40mA | 5V 40mA | 5V 40mA | 5V 40mA | 5V 40mA | 5V 40mA | 3,3V 3 15 mA (130mA entre todos) | 3,3 5V 25 mA (125mA entre todos) | 5v | - |
| Pines analógicos E/S | 6/0 | 8/0 | 6/0 | 16/0 | 12/0 | 12/0 | 12/2 | 6/0 | - | - |
| Tensión/resolución pines analógicos | 3,3 ó 5V 10 bits (1024 valores) | 5V 10 bits (1024 valores) | 5V 10 bits (1024 valores) | 5V 10 bits (1024 valores) | 5V 10 bits (1024 valores) | 5V 10 bits (1024 valores) | 3,3V 12 bits (4096 valores) | 5V 12 bits (4096 valores) | - | - |
| Pines con interrupción externa | 2 | 2 | 2 | 6 | 2 | 2 | - | - | - | - |
| Pines PWM | 6 | 6 | 6 | 15 | 7 | 7 | 12 | 6 | - | - |
| Conexiones serial/UART | 1 | 1 | 1 | 4 | 1 | 1 | 4 | 4 | 8 | SI |

**Tabla 6.6.0.2 –** Tabla comparativa sobre diferentes tipos de Arduinos en el mercado

| |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| Modelo | Pro Mini | Nano | Uno | Mega/ Mega 2560 | Leonardo | Micro | Due | Netduino 2 | Stellaris Launchpad LM4F120 | Raspberry Pi |
| Conexiones I2C/TWI | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 4 | SI |
| Conexiones ISP/ICSP | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | SI |
| Conexiones USB | No, (necesita adaptador externo) | Si | Si, USB-B | USB-B | Si,Nativa, Micro USB | Si,Nativa, Micro USB | Si,Nativa, Micro USB | Si,Nativa, Micro USB | Si,Nativa, Micro USB | Si, Micro USB |
| Conexión USB de depuración | No | No | No | No | No | No | Si, Micro USB | Si, Micro USB | Si, Micro USB | - |
| Conexión bluetooth | No | No | No | No | No | No | No | No | No | - |
| Conexión WiFi | No | No | No | No | No | No | No | No | No | - |
| Conexión Ethernet | No | No | No | No | No | No | No | No | No | Si |
| Conexión USB host | No | No | No | No | No | No | Si | No | Si | Si |

**Tabla 6.6.0.2 –** Tabla comparativa sobre diferentes tipos de Arduinos en el mercado

| |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| Modelo | Pro Mini | Nano | Uno | Mega/ Mega 2560 | Leonardo | Micro | Due | Netduino 2 | Stellaris Launchpad LM4F120 | Raspberry Pi |
| Almacenamiento por SD | No | No | No | No | No | No | No | No | No | Si |
| Corriente por el pin de 5V | - | 500 | 500 800mA | 500 800mA | 500 800mA | 500mA | 800mA | - | - | - |
| Corriente por el pin de 3.3V | - | 50mA | 50mA | 50mA | 50mA | 50mA | 800mA | - | - | - |
| Voltaje de alimentación por el USB | 3,3V ó 5V (sin usb) | 5V | 5V | 5V | 5V | 5V | 5V | 5V | 5V | 5V |
| Voltaje de alimentación recomendado por el jack | 3,35 12V (modelo 3,3V) ó 5 12V (modelo 5V) | 7 12V | 7 12V | 7 12V | 7 12V | 7 12V | 7 12V | 7,5 9V | - | - |

**Tabla 6.6.0.2 –** Tabla comparativa sobre diferentes tipos de Arduinos en el mercado

| |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| Modelo | Pro Mini | Nano | Uno | Mega/ Mega 2560 | Leonardo | Micro | Due | Netduino 2 | Stellaris Launchpad LM4F120 | Raspberry Pi |
| Voltaje de alimentación limite por el jack | - | 6 20V | 6 20V | 6 20V | 6 20V | 6 20V | 6 20V | - | - | - |
| Precio oficial | 15€+Ge | - | 20€+Ge | 40€+Ge | 18€+Ge | 18€+Ge | 39€+Ge | 35€+Ge | 13€+Ge | 43€+Ge |
| Precio BBB | 4€ | 9€ | 10€ | 12€ | 11€ | 16€ | 38€ | 25 30€ | 15€ | 35€ |

**Tabla 6.6.0.2 –** Tabla comparativa sobre diferentes tipos de Arduinos en el mercado

## 6.7. Selección de la forma de envío de la información ante un evento de emergencia

En los requisitos de diseño de este dispositivo se especifica que el mismo debe enviar algún tipo de comunicado en caso de emergencia, pero se ha dejado abierta la forma a través de la cual el dispositivo la establezca, es decir, que el dispositivo de localización avise mediante un SMS o bien mediante un correo electrónico.

Se considera (por lo menos en España) que enviar un correo electrónico es mucho mas económico que enviar un SMS, por el simple motivo de que el coste de enviar un correo electrónico con cualquier tipo de operador sale al mismo precio, dado que repercute en la tarifa de datos del operador que se disponga. Hoy en día, la tarifa de datos con el coste mensual mas asequible ya proporciona megas suficientes para poder enviar una gran cantidad de E-mails sin sobrepasar el consumo. No ocurre lo mismo con los SMS, pues dependiendo del operador de telefonía que se disponga, los SMSs podrán salir mas baratos o mas caros.

Es cierto que existen compañías de telefonía que ofrecen los SMSs gratuitos si se envía una cantidad inferior de 1000 al mes, siendo la cantidad de destinatarios diferentes inferior a 100, mas que suficiente para el uso del dispositivo localizador, pero no es factible que tengan que utilizarse ciertos operadores de telefonía para el uso de este dispositivo.

También existen limites a la hora de enviar correos electrónicos, aunque son menos restrictivos que para el caso de los SMSs. Como ejemplo, se pueden observar los limites que presenta Google para el envío de correo electrónico.

- https://support.google.com/a/answer/176600?hl=es
- https://support.google.com/a/answer/166852

También puede surgir la duda de, en caso de el sistema de localización envíe la emergencia a través de correo electrónico, la forma mediante la cual el usuario se percate de que ha recibido un correo electrónico.

Se suele tender a no consultar el correo con la frecuencia adecuada, por la contra, cuando llega un SMS al dispositivo de telefonía del que se disponga, el teléfono rápidamente avisa mediante un tono de notificación y se tiende a ver que es.

Hoy en día es mas común poseer un smartphone, la mayoría de estos dispositivos piden una cuenta de correo (generalmente de Google) para poder acceder a la APP store, almacenar contactos y demás. Es muy sencillo hacer que cualquier smartphone notifique que ha llegado un correo electrónico si se ha configurado previamente una cuenta de correo en el propio smartphone, pero, ¿quien no lo ha hecho?.

Cabe comentar que configurar los parámetros de la notificación de un smartphone dependerá del modelo de dispositivo y la versión de Android/Ios que tenga instalada, de todas formas existe gran cantidad de información en internet en este aspecto.

Por los motivos que se han descrito anteriormente, se considera que es mejor opción enviar E-mails que SMS.

# 7 Resultados finales

En este capítulo se obsevará el resultado del presente proyecto y se realizara un pequeño resumen de lo que se ha conseguido tras el desarrollo del proyecto. No se explicara el funcionamiento del mismo, ni tampoco como se construye este sistema, para ello se han realizado unas guías que lo explican de forma detallada y que se enumeran a continuación.

1. En la sección 4.4 situado en el Pliego de condiciones se muestra la implementación física del dispositivo de localizacion.

2. En el Manual de usuario se muestra el manual de utilización del dispositivo.

Si se sigue la guía de implementación, se puede llegar al resultado que se manifiesta en la figura 7.0.0.1.

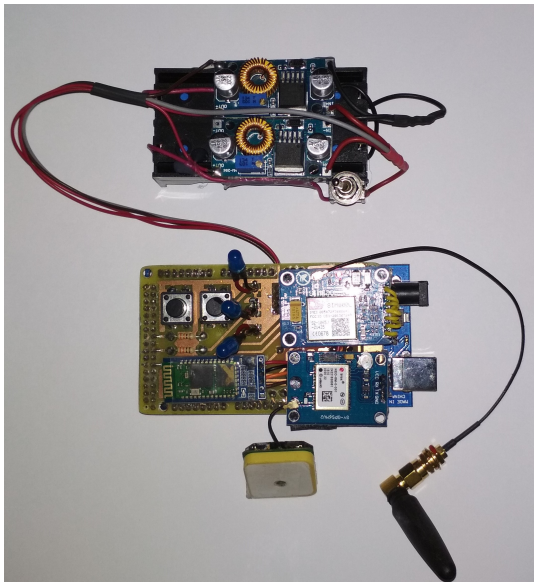
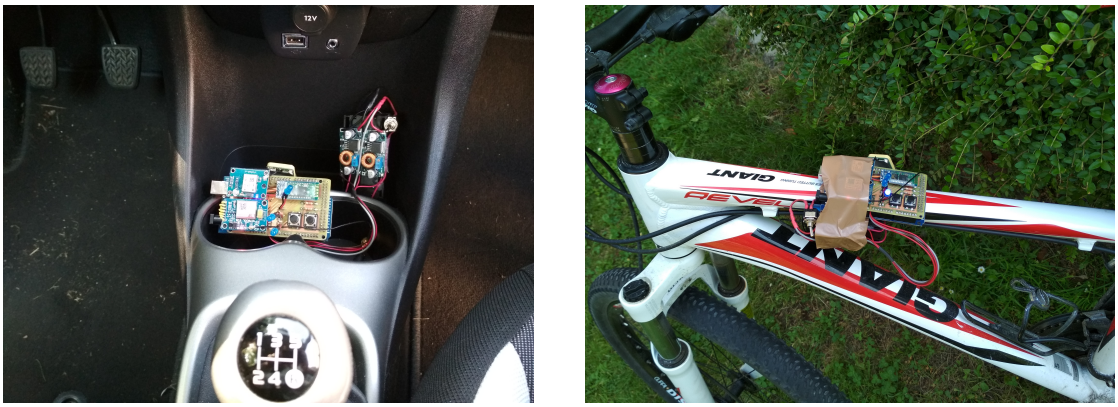


**Figura 7.0.0.1 –** Imagen del sistema de localización de objetos en tiempo real terminado

La pagina web creada para la visualización de la posición el dispositivo de localización posee esta url `http://mapalocgps.appspot.com` y en la figura 7.0.0.2 se puede ver su apariencia.


**Figura 7.0.0.2 –** Imagen de la pagina WEB realizada para el sistema de localización

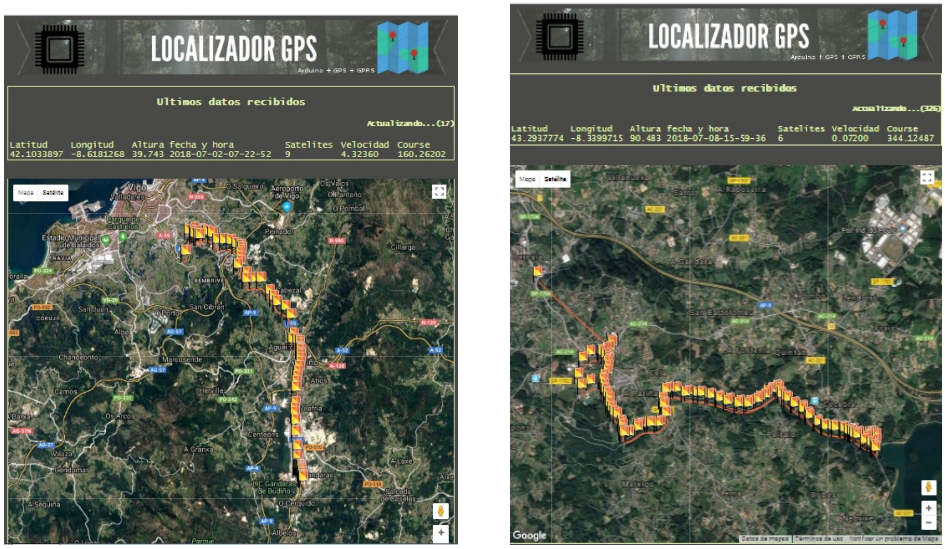
Se han realizado numerosas pruebas con el dispositivo de localización en tiempo real aunque la mayoría de ellas se han hecho a pie, el dispositivo también ha sido instalado en una bicicleta y en el salpicadero de un automóvil tal y como muestra la figura 7.0.0.3 y los resultados han sido realmente buenos, estos se muestran en la figura 7.0.0.4.



(a) Sistema de localización instalado en un vehículo

(b) Sistema de localización instalado en una bicicleta

**Figura 7.0.0.3 –** Instalación del dispositivo de localización en diferentes medios de transporte




(a) Resultados obtenidos de la pagina web del dispositivo tras su instalación en un vehículo

(b) Resultados obtenidos de la pagina web del dispositivo tras su instalación en una bicicleta

**Figura 7.0.0.4 –** Resultados obtenidos en la pagina web del dispositivo de localización

El dispositivo del localización de objetos en tiempo real realizado en el presente proyecto es un prototipo funcional que cumple con los requisitos indicados en el apartado [5], si es cierto que pese a que funciona correctamente, seria necesario realizar dos o tres iteraciones tras esta versión, reducir mucho el tamaño y por supuesto mejorar el software para que sea mantenible, realizar una APP para Android para controlarlo y visualizar la posición, posibilitar al usuario que suba el firmware y de esta forma actualizar el dispositivo, se podrían seguir diciendo muchas mejoras que se podrían implementar en el dispositivo.

No hay que olvidar que tal y como se adelantaba en la sección [4.1] que el mercado chino ofrece dispositivos similares con un coste muy bajo, la gran mayoría se comunica a través de GSM utilizando SMSs y los que ofrecen tecnología GPRS son complicados de configurar, por otro lado también existen compañías que por una cuota mensual ofrecen el servicio de localización. El dispositivo que se ha desarrollado en este trabajo ha pretendido reunir desde el principio estas tres características:

- Que sea fácil de transportar.
- Que sea sencillo de configurar.
- Que su coste en el mercado sea asequible.

Con el proyecto finalizado, la característica «Que sea fácil de transportar» no la cumple al pie de la letra, aunque el resultado es un prototipo, no un producto final, aun faltaría un largo camino para que este proyecto pueda considerarse un producto final, de todas formas, es factible reducir su tamaño, el resto de características se cumplen de forma satisfactoria.

La experiencia tras el desarrollo de este proyecto se puede valorar de forma muy positiva, se han aprendido muchos conocimientos, sobre todo relativos a la conectividad entre los dispositivos y la forma de interactuar con el «Internet de las cosas», en estos tiempos que corren todo esta conectado con todo y este proyecto ha servido y sirve para comprender la forma practica de como realizarlo.

# 8 Disposiciones legales y normativa aplicada

En este apartado se contempla el conjunto de disposiciones legales, es decir, leyes y reglamentos de obligado cumplimiento que se han tenido en cuenta para la realización del presente documento. La normativa en la cual se basa este proyecto se puede subdividir en dos partes:

1. Normativa de la Escuela Politécnica de Ingeniería de Gijón para la realización de trabajos de fin de máster.
    - Reglamento sobre la elaboración y defensa de los trabajos fin de máster (Acuerdo de 30 de abril de 2010), en la Universidad de Oviedo.día 1 de junio de 2010[Num 125 (págs. 1 a 3)]
    - Reglamento de Trabajos Fin del Máster en la universidad de Oviedo (aprobado por la comisión académica del master el 18/01/2016).
    - UNE-EN-ISO 216.“Papel de escritura y ciertos tipos de impresos. Formatos acabados, series A y B”.
    - UNE 82100.“Magnitudes y unidades (partes de 0 a 13)”.
    - UNE 50132.“Numeración de las divisiones y subdivisiones en los documentos escritos”.
2. Normativas que afectan a la calidad y exigencias de fabricación de los componentes utilizados para su comercialización y uso en la Unión Europea.
    - DIRECTIVA 2004/108/CE del Parlamento Europeo y del Consejo de 15 de diciembre de 1004 relativa a la aproximación de las legislaciones de los Estados Miembros en materia de compatibilidad electromagnética.
    - UNE 20620-4:1980.“Materiales de base con recubrimiento metálico para circuitos impresos.Hoja de cobre”.
    - UNE 21302-521:2004.“Vocabulario electrotécnico. Capitulo 521: Dispositivos semiconductores y circuitos integrados”.
    - EN 123000/A1:1995.“Especificación genérica: Placas de circuitos impresos”.(Ratificada por AENOR en junio de 1996).
    - EN 12300/A1:199.“Especificación intermedia:placas de circuitos impresos de simple y doble cara con agujeros no metalizadas”.(Ratificada por AENOR en junio de 1996).

# 9 Software utilizado

En este apartado se incluyen los programas que han sido claves para la realización del presente proyecto. Ver la tabla (9.0.0.1).

**Arduino IDE:** Para flashear el programa a nuestra placa Arduino.

**Eclipse:** Para escribir el código de una forma mas estructurada que con el IDE de Arduino.

**Platform IO IDE:** Idem que en el caso de eclipse.

**Proteus Professional V8.1:** Utilizado para elaborar los esquemas electronicos del proyecto.

**Coolterm:** Terminal serie que nos ha servido para poder enviar comandos AT con los diferentes modulos utilizados en el proyecto. También nos ha servido para testear el GPS.

**Autocad:** Para generar los planos de este documento.

**Microsoft Excel:** Para realizar los calculos del coste de todos los materiales utilizados para la elaboracion del presente proyecto.

**Doxygen:** Para generar la documentación asociada al codigo de programación.

**Tabla 9.0.0.1 –** Software utilizado para la realización del proyecto

# 10 Bibliografía

En este apartado se incluyen otras referencias utilizadas para la realización del presente proyecto. En este caso la consulta de referencias a páginas de Internet supera la la consulta de libros realizada debido sobre todo a que la mayor parte de la información consultada es referida a los datasheets de los fabricantes de los diferentes dispositivos utilizados en el TFM (Trabajo de fin de máster).A continuación se elabora una lista de las referencias consultadas:

## 10.1. Bibliografía

[1] Anónimo. *HC-03/05 Embedded Bluetooth Serial Communication Module*. Ultima revisión : Abril, 2011.[Consultado el 10 Noviembre 2017]. Disponible en: https://cdn.instructables.com/ORIG/F8O/SMME/IU5NM2JJ/F8OSMMEIU5NM2JJ.pdf

[2] UBlox Company. *U-blox-6-Receiver-Description-Including-Protocol-Specification*. Publicado el 18 Abril 2013. [Consultado el 10 Enero 2018]. Disponible en: https://www.u-blox.com/sites/default/files/products/documents/u-blox6_ReceiverDescrProtSpec_%28GPS.G6-SW-10018%29_Public.pdf.

[3] Simcom Company. $SIM800Series\_Email\_Application\_Note\_V1{,}00$. Publicado el 1 de Agosto de 2013. Versión 1.00.[Consultado el 15 de enero de 2018]. Disponible en:http://dostmuhammad.com/blog/sim800-series-firmware-update-appnotes/

[4] Simcom Company. $SIM800Series\_SSL\_Application\_Note\_V1{,}01$. Publicado el 30 de junio de 2014. Versión 1.01.[Consultado el 10 de febrero de 2018]. Disponible en:http://dostmuhammad.com/blog/sim800-series-firmware-update-appnotes/

[5] Simcom Company. $SIM800Series\_ATCommand\_Manual\_V1{,}09$. Publicado el 3 de agosto de 2015. Versión 1.09.[Consultado el 20 de febrero de 2018]. Disponible en:http://dostmuhammad.com/blog/sim800-series-firmware-update-appnotes/

[6] *Cooking-Hacks.com [Internet]*. España:cooking-hacks; 2006 [actualizado 14 Feb 2013; citado 5 may 2017]. Disponible en: https://www.cooking-hacks.com/projects/arduino-realtime-gps-gprs-vehicle-tracking.

[7] *librearduino.blogspot.com.es[Internet]*. [Consulta 8 agosto 2017]. Disponible en: https://tinyurl.com/librearduino.

[8] *playground.arduino.cc/es/es[Internet]*.[Consulta 8 septiembre 2017]. Disponible en: http://playground.arduino.cc/es/es

[9] *tallerarduino.com[Internet]*.[Consulta 10 diciembre 2017]. Disponible en:http://tallerarduino.com/category/videotutoriales/arduino-tutorials/

# 11 Anexos

## 11.1. Documentación de partida

En este anexo figura el justificante de asignación del trabajo de fin de grado donde se recoge el título del trabajo de fin de grado, el tutor, el número del TFG asignado y de forma breve los objetivos del proyecto.

| **Título** | **Tipo de Propuesta** | **Tutor** | **Departamento** | **Máster al que se oferta** |
|---|---|---|---|---|
| DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | Plantas, Instalaciones, Construcción o Equipos Industriales, Informáticos o de Telecomunicación | MARIA ANTONIA GARCIA PRIETO | Construcción e Ingeniería de Fabricación | MIIND |
| El objeto principal del presente proyecto es desarrollar la parte electrónica que constituirá la base de un prototipo funcional de un seguidor localizador de objetos en tiempo real. Concretando más, el objeto principal del presente proyecto consiste en utilizar las tecnologías, GSM, GPS GPRS y Bluetooth para efectuar el control sobre un dispositivo, aunque se podría extrapolar a cualquier elemento utilizando la red de datos móviles GPRS como canal de comunicación principal. El usuario ha de ser capaz de tener localizado el objeto y recibir cualquier percance que le ocurra al mismo a través de algún tipo de notificación (sms, email...). La documentación prestará especial detalle a los elementos necesarios, eligiendo la opción más económica y fiable de todas las posibles. Se incluirá una discusión entre distintas opciones de diseño del sistema y también una valoración económica, dentro del documento Presupuesto, para tener una idea aproximada del coste de ejecución del proyecto. | | | | |

**Tabla 11.1.0.1 –** Tabla resumen de la documentación de partida

## 11.2. Comandos AT utilizados

Los comandos AT, también conocidos como comandos Hayes (en honor a su desarrollador Dennis Hayes), son una serie de instrucciones que conforman un interfaz de comunicación entre usuario y módem. Su abreviatura AT por la que son mundialmente conocidos estos comandos proviene de la palabra 'attention'.

Aunque la finalidad principal de los comandos AT fue la comunicación con módems, la telefonía móvil GSM/GPRS también adoptó este lenguaje como estándar de comunicación.

En la actualidad, todos los terminales móviles GSM poseen una serie específica de comandos AT que nos permiten configurarlos por medio de estas instrucciones e indicarles una serie de acciones que queremos que ejecuten, tales como marcar un número de teléfono, enviar o leer un SMS, consultar el estado de conexión a la red, leer o escribir en la agenda de contactos, etc.

Gracias a que la transmisión de comandos AT no depende del canal de comunicación a través del cual estos sean enviados (cable, infrarrojos, Bluetooth, etc.), podremos utilizar nuestra placa Arduino para transmitir dichos comandos a un módulo GPRS/GSM que sea capaz de interpretarlos y actuar en consecuencia.

Como son muchos los comandos existentes (véase el manual de comandos AT oficial de nuestro módulo GPRS/GSM(SIM800) , únicamente vamos a mencionar junto con una breve descripción, aquellos que puedan resultarnos de principal interés en el desarrollo del proyecto:

### 11.2.1. Comandos AT utilizados por el modulo GPRS-GSM

#### 11.2.1.1. Comandos AT de comprobación e inicializacion de la tarjeta SIM

| Comando AT | Descripción | Respuesta |
|---|---|---|
| **AT** | Con este comando verificaremos que el módulo está recibiendo nuestras instrucciones. | **OK** |
| **AT+CPIN?** | Utilizaremos esta instrucción para conocer si se requiere introducir el código PIN para proceder con el desbloqueo de la misma. | La respuesta es: +CPIN: <code>, donde:<br><code>= READY : Ya se ha introducido el PIN en el dispositivo, o bien no se requiere PIN.<br><code>= SIM PIN: Espera a que se introduzca el PIN. |

**Tabla 11.2.1.1 –** Tabla resumen de comandos para la comprobación del estado del modulo SIM800L

| Comando AT | Descripción | Respuesta |
|---|---|---|
| **AT+CPIN=<pin>** | En el caso de que necesitemos introducir el PIN, éste es el comando que debemos enviar, escribiendo los 4 dígitos correspondientes al código de desbloqueo. | OK En el caso de que la tarjeta este insertada y el PIN sea correcto.<br>ERROR en cualquier otro caso. |
| **AT+CREG?** | Con este comando estamos preguntando por el estado de conexión a la red. | La respuesta recibida seguirá la siguiente notación: +CREG: <n>,<stat>[,<lac>,<ci>] , donde:<br><stat>= 0 : No registrado, no está buscando una conexión de red.<br><stat>= 1 : Registrado a la red nacional.<br><stat>= 2 : No registrado, pero buscando la red.<br><stat>= 3 : Registro denegado.<br><stat>= 4 : Desconocido.<br><stat>= 5 : Registro tipo roaming.<br>El parametro <n>por defecto es 0 a no ser que hayamos configurado el comando previamnete, con lo cual los parametros <lac>y <ci>no seran mostrados. |
| **AT+COPS?** | Mediante esta instrucción recibiremos la confirmación de la compañía en la que está registrada nuestra tarjeta SIM (Movistar, Orange, Vodafone, etc.) | |

**Tabla 11.2.1.1 –** Tabla resumen de comandos para la comprobación del estado del modulo SIM800L

#### 11.2.1.2. Comandos AT para el envío y recepción de un SMS

| Comando AT | Descripción | Respuesta |
|---|---|---|
| **AT+CMGF=1** | Modo texto para el envío del SMS | OK |

**Tabla 11.2.1.2 –** Tabla resumen de comandos para el envío y recepción de un SMS

| Comando AT | Descripción | Respuesta |
|---|---|---|
| AT+CSCS="GSM"<br>AT+CMGS="+861391<br>818xxxx"<br>>This is a test<br><Ctrl+Z> | Nos enviamos un SMS a nosotros mismos. | OK<br>+CMGS:34<br>OK |
| | Recibimos una notificación indicando que hemos recibido un nuevo SMS, y se nos indica también la posición de memoria en la que se ubica. | +CMTI: "SM",1 |
| **AT+CMGR=1** | Leemos el SMS que nos ha llegado. Nota: El numero debe ser el mismo que nos aporta la notificación previamente recibida +CMTI. | +CMGR: REC UNREAD",<br>"+8613918186089", "","02<br>/01/30,20:40:31+00"<br>This is a test<br>OK |
| **AT+CMGR=1** | Volvemos a leer el SMS de nuevo, vemos que en la respuesta se cambia el estado , anteriormente era "UNREAD" y ahora es "READ". | +CMGR: "REC READ",<br>"+8613918186089","",<br>"02/01/30,20:40:31+00"<br>This is a test<br>OK |
| **AT+CMGS="+861391<br>818xxxx"<br>>Test again<Ctrl+Z>** | Volvemos a enviarnos a nosotros mismos un nuevo SMS. | +CMGS:35<br>OK |
| | Recibimos una notificación indicando que hemos recibido un nuevo SMS, y se nos indica también la posición de memoria en la que se ubica. | +CMTI: "SM",2 |

**Tabla 11.2.1.2 –** Tabla resumen de comandos para el envío y recepción de un SMS

| Comando AT | Descripción | Respuesta |
|---|---|---|
| **AT+CMGL=“ALL”**<br>Nota:“ALL” debe estar en mayúsculas. | Vemos todos los SMS que hemos recibido. | +CMGL: 1, “REC READ”,“+8613918186089”, “”,“02/01/30,20:40:31+00”<br>This is a test<br>+CMGL: 2,“REC UNREAD”,“”, “+8613918186089”, “” ,“02/01/30,20:45:12+00”<br>Test again<br>OK |
| **AT+CMGD=1** | Borra el SMS situado en 1. | OK |
| **AT+CMGL=“ALL”** | Vemos todos los SMS que hemos recibido.Notamos que ahora ha desaparecido el SMS que se ha borrado anteriormente. | +CMGL: 2, “REC READ”, “+8613918186 089”,“”,“02/01/30,20:45:12+ 00”<br>Test again<br>OK |

**Tabla 11.2.1.2 –** Tabla resumen de comandos para el envió y recepción de un SMS

#### 11.2.1.3. Comandos AT para la realización de una petición HTTP

| Comando AT | Descripción | Respuesta |
|---|---|---|
| **AT+HTTPINIT** | Inicia el servicio HTTP | **OK** |
| **AT+HTTPPARA = “CID”, 1** | Se introducen los parámetros de la sesión HTTP | **OK** |
| **AT+HTTPPARA = “URL”,“www.gmail.com”** | | **OK** |
| **AT+HTTPACTION = 0** | Verifica si la sesión ha comenzado correctamente. | OK<br>+HTTPACTION: 0,200,84200<br>NOTA: 200 significa que se ha conectado correctamente. |
| **AT+HTTPTERM** | Finaliza el servicio HTTP. | **OK** |

**Tabla 11.2.1.3 –** Tabla resumen de comandos para la realización de una petición HTTP

#### 11.2.1.4. Comandos AT para el envío de un E-MAIL

| Comando AT | Descripción | Respuesta |
|---|---|---|
| **AT+SAPBR=3,1, "APN", "CMNET"** | Se configura el APN de nuestra tarjeta SIM. El nombre varia dependiendo del operador de telefonía que estemos usando. | **OK** |
| **AT+SAPBR = 1,1** | To open a GPRS context. | **OK** |
| **AT+EMAILCID = 1** | Se establece el identificador de perfil de portador de correo electrónico. | **OK** |
| **AT+EMAILTO = 30** | Se introduce el tiempo de envío del E-mail | **OK** |
| **AT+EMAILSSL = 1** | El contenido del E-mail y su puerto se transmite con encriptacion SSL | **OK** |
| **AT+SMTPSRV = "SMTP.GMAIL.COM","465"** | Se introduce la dirección del servidor SMTP y el puerto. | **OK** |
| **AT+SMTPAUTH = 1,"account" ,"password"** | Se introduce el nombre de usuario y la contraseña. | **OK** |
| **AT+SMTPFROM = "account@GMAIL.COM"** | Se introduce la dirección de correo del remitente. | **OK** |
| **AT+SMTPSUB="Test"** | Se introduce el asunto del E-mail | **OK** |
| **AT+SMTPRCPT = 1,0, "john@sim.com", "john"** | Se introduce la dirección de correo del destinatario. | **OK** |
| **AT+SMTPBODY = 19** | Se escribe el numero de bytes "letras" que contiene el cuerpo del E-mail. Una vez se escriba el numero de letras correspondientes, o bien transcurra el tiempo de escritura, el e-mail estará listo para ser enviado. | DOWNLOAD<br><Escribir contenido del E-mail><br>OK |
| **AT+SMTPSEND** | Se envía el E-mail | **OK**<br>**+SMTPSEND: 1** |

**Tabla 11.2.1.4 –** Tabla resumen de comandos para el envío de un correo electrónico

NOTA: Para el envío de cualquiera de los comandos AT a través de Hyperterminal o del entorno de desarrollo de Arduino, debemos seleccionar siempre los caracteres fin de línea y retorno de carro.

### 11.2.2. Comandos AT utilizados por el modulo Bluetooth

Al igual que en apartado anterior son muchos los comandos AT existentes para nuestro Bluetooth (HC-05) unicamente mencionaremos aquellos se que han utilizado para el desarrollo del presente trabajo y mas comúnmente utilizados, estos son:

Aunque hay multitud de comandos AT, los que se representan en la siguiente tabla son los que vamos a utilizar con una mayor frecuencia.

Al entrar en el modo de comandos AT (LED parpadea cada 2 segundos).

| **Comando AT** | **Descripción** | **Respuesta** |
|---|---|---|
| **AT+ORGL** | *Restaura los valores por defecto del módulo Bluetooth HC-05* | *Responde con un **OK**. (+ROLE:0; +CMC)+CMOD:0; +UART:38400,0,0; +PSWD:1234 y Nombre: "HC-2010-06-01" o "HC-05").* |
| **AT** | Test de comunicación. | Responde con un OK. |
| **AT+VERSION** | Retorna la versión del módulo | Responde con la versión **(+VERSI0N:2.0-20100601)**. |
| **AT+UART=X, Y,Z** | Configura la velocidad de trasmisión del módulo según el valor de "x": 1200 bps, 2400 bps, 4800 bps, 9600 bps, 19200 bps, 38400 bps (por defecto), 57600 bps, 115200 bps, 230400 bps, 460800 bps, 921600 bps, 1382400 bps. Con el valor de "Y"se configura el bit de parada: 0→1bit. 1→2 bits. Con el valor de "Z" se configura el bit de paridad: 0 → Sin paridad. 1→Paridad impar. 2→Paridad Par. | AT+UART=38400,0,0 configura el módulo Bluetooth HC-05 con una velocidad de comunicación de 38400 bps, con un bit de parada y sin paridad. Responde con un **OK**. AT+UART. Preguntamos por el estado de la comunicación UART. Responde con **+UART:38400,0,0**. |
| ***AT+NAME=XXXX*** | Configura el nombre con el que se visualizara el módulo, soporta hasta 20 caracteres. | AT+NAME=Arduino_HC-0 configura el nombre del módulo a Arduino_HC-05. Responde con un **OK**. |

**Tabla 11.2.2.1 –** Tabla resumen de comandos AT del modulo HC-05

| Comando AT | Descripción | Respuesta |
|---|---|---|
| ***AT+PSWD=XXXX*** | *Configura el Pin o la contraseña de acceso al módulo Bluetooth HC-05.* | AT+PSWD=1234 configura el pin a 1234. Responde con un OK. AT+PSWD. Preguntamos por el pin. Nos responde con **+PSWD:1234**. |
| **AT+CMODE = X** | *Cambiamos el modo de conexión del módulo según el valor de "X": 0→ Se conecta al dispositivo Bluetooth especificado. 1 → Se conecta a cualquier dispositivo Bluetooth disponible.* | *AT+CMODE = 1 configura el módulo Bluetooth HC-05 para que pueda conectarse a cualquier dispositivo Bluetooth disponible (sin dirección requerida). Responde con un **OK**. AT+CMODE: Preguntamos por el modo de conexión del módulo. Responde con **+CMOD:1**.* |
| **AT+ROLE = X** | *Cambiamos el modo de trabajo del módulo, según el valor de "X": 0 → Modo Esclavo. 1 → Modo Maestro.* | *AT+ROLE = 0 configura el módulo Bluetooth para que trabaje como Esclavo de otro dispositivo Bluetooth. Responde con un **OK**. AT+ROLE: Preguntamos por el modo de trabajo del módulo. Responde con **+ROLE:O**.* |

**Tabla 11.2.2.1 –** Tabla resumen de comandos AT del modulo HC-05

## 11.3. Cálculos

Para la realización del presente trabajo, los cálculos realizados se reducen prácticamente a la conversión de grados decimales a grados sexagesimales para la representación en un mapa proporcionado por Google, las coordenadas enviadas por el dispositivo localizador.

Haciendo uso de las recomendaciones de los fabricantes de los diferentes dispositivos utilizados para la elaboración de nuestro localizador, display LCD, encoder, hemos reducido los cálculos considerablemente.

### 11.3.1. Notación sexagesimal

Podemos expresar una cantidad en grados, minutos y segundos, las partes de grado inferiores al segundo se expresan como parte decimal de segundo, ejemplo:

- $1234'34''$

- $133'23,8''$
- $12445'34,70''$
- $-234'10''$

Teniendo cuidado como norma de notación, no dejar espacio entre las cifras, es decir: escribir $1234'34''$ y no $1234'34''$. Podemos también representar en forma decimal la medida de un ángulo en representación sexagesimal teniendo en cuenta que:

- $1' = (1/60) = 0,01666667$ (redondeando a ocho dígitos)
- $1'' = (1/60)' = (1/3600) = 0,00027778$
- Así $1215'23'' = 12 + 15(1/60) + 23(1/3600) = 12,25639$

### 11.3.2. Notación decimal

Una cantidad en grados se puede expresar en forma decimal, separando la parte entera de la fraccionaria con la coma decimal, se divide en 60 en la forma normal de expresar cantidades decimales, lo que se busca es transformar en minuto y el segundo números decimales, por ejemplo:

- $23,2345$
- $12,32$
- $-50,265$
- $123,696$

### 11.3.3. Ejemplo de conversión

Nuestro dispositivo localizador nos ha enviado un SMS indicándonos la posición en la cual se encuentra.

- La longitud es la siguiente: 08°20,423613'W
- La latitud es la siguiente: 43°17,610714'N

Los datos que nos envía el dispositivo son números con notación decimal hasta los minutos, dado que por defecto nuestro localizador nos envía las coordenadas en formato DM (degrees minutes), con lo cual la conversión la tenemos bastante sencilla.

Necesitamos saber cuáles son las coordenadas sexagesimales tanto de la longitud como de la latitud a partir de los minutos.

Comenzamos con la longitud.

Sabemos cuales son los grados (08) y los minutos (20), nos queda por saber cuáles son los segundos, para ello debemos realizar la siguiente operación.

Sabemos que los minutos son 20,423613.

Restamos los minutos y el resultado lo multiplicamos por 60 para obtener los segundos:

- $20,423613 - 20 = 0,423613$
- $0,423613 * 60 = 25,41678$

Ya sé que son 25”

Luego obtenemos que la longitud es con exactitud 08°20’25,41678”.

Continuamos con la latitud:

Sabemos cuales son los grados (43) y los minutos (17), nos queda por saber cuáles son los segundos, para ello debemos realizar la siguiente operación.

Sabemos que los minutos son 17,610714.

Restamos los minutos y el resultado lo multiplicamos por 60 para obtener los segundos:

- $17,610714 - 17 = 0,610714$
- $0,610714 * 60 = 36,64284$

Ya sé que son 36”

Luego obtenemos que la longitud es con exactitud 43°17’36,64284”.

Nos queda por averiguar el significado de las letras N (norte) y W (oeste, en inglés, west) que se observan al final del SMS, estas nos indican el cuadrante del hemisferio en el que estamos, la figura 11.3.3.1 nos aclara de forma gráfica lo anteriormente citado.

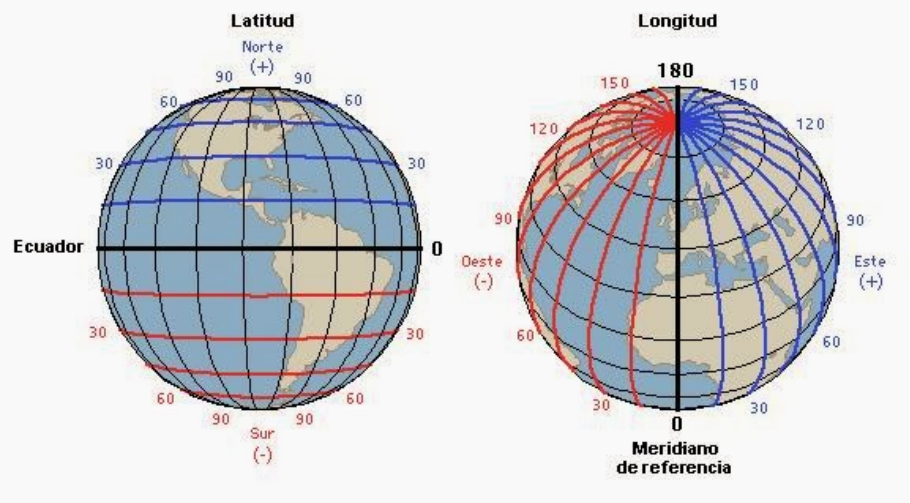



**Figura 11.3.3.1 –** Aclaración del sistema de posicionamiento mediante coordenadas

## 11.4. Código fuente Arduino

En este anexo se presenta todo el código fuente que debe ser grabado en Arduino MEGA.

### 11.4.1. Estructura de datos

Lista de estructuras con una breve descripcion:

### 11.4.2. Lista de archivos

Lista de todos los archivos con descripciones breves:



### 11.4.3. Referencia de la Estructura NAV_PVT

#### Campos de datos

- unsigned char cls
- unsigned char id
- unsigned short len
- unsigned long iTOW
- unsigned short year
- unsigned char month
- unsigned char day
- unsigned char hour
- unsigned char minute
- unsigned char second
- char valid
- unsigned long tAcc
- long nano
- unsigned char fixType
- char flags
- unsigned char reserved1

- unsigned char numSV
- long lon
- long lat
- long height
- long hMSL
- unsigned long hAcc
- unsigned long vAcc
- long velN
- long velE
- long velD
- long gSpeed
- long heading
- unsigned long sAcc
- unsigned long headingAcc
- unsigned short pDOP
- short reserved2
- unsigned long reserved3

#### 11.4.3.1. Descripcion detallada

Estructura de datos de la informacion que va a ser recibida del GPS

Definicion en la linea 368 del archivo location_system.cpp.

#### 11.4.3.2. Documentacion de los campos

#### 11.4.3.3. cls

```
unsigned char NAV_PVT::cls
```

Definicion en la linea 370 del archivo location_system.cpp.

#### 11.4.3.4. day

```
unsigned char NAV_PVT::day
```

Dia del mes, rango 1..31 (UTC)

Definicion en la linea 377 del archivo location_system.cpp.

#### 11.4.3.5. fixType

`unsigned char NAV_PVT::fixType`

GNSS fix Type, rango 0..5

Definicion en la linea 384 del archivo location_system.cpp.

#### 11.4.3.6. flags

`char NAV_PVT::flags`

Fix Status Flags

Definicion en la linea 385 del archivo location_system.cpp.

#### 11.4.3.7. gSpeed

`long NAV_PVT::gSpeed`

Ground Speed (2-D) (mm/s)

Definicion en la linea 399 del archivo location_system.cpp.

#### 11.4.3.8. hAcc

`unsigned long NAV_PVT::hAcc`

Precision horizontal estimada (mm)

Definicion en la linea 393 del archivo location_system.cpp.

#### 11.4.3.9. heading

`long NAV_PVT::heading`

Heading of motion 2-D (deg)

Definicion en la linea 400 del archivo location_system.cpp.

#### 11.4.3.10. headingAcc

`unsigned long NAV_PVT::headingAcc`

Heading Accuracy Estimate

Definicion en la linea 402 del archivo location_system.cpp.

#### 11.4.3.11. height

`long NAV_PVT::height`

Height above Ellipsoid (mm)

Definicion en la linea 391 del archivo location_system.cpp.

#### 11.4.3.12. hMSL

`long NAV_PVT::hMSL`

Altura sobre el nivel del mar (mm)

Definicion en la linea 392 del archivo location_system.cpp.

#### 11.4.3.13. hour

`unsigned char NAV_PVT::hour`

Hora del dia, rango 0..23 (UTC)

Definicion en la linea 378 del archivo location_system.cpp.

#### 11.4.3.14. id

`unsigned char NAV_PVT::id`

Definicion en la linea 371 del archivo location_system.cpp.

#### 11.4.3.15. iTOW

`unsigned long NAV_PVT::iTOW`

GPS time of week of the navigation epoch (ms)

Definicion en la linea 373 del archivo location_system.cpp.

#### 11.4.3.16. lat

`long NAV_PVT::lat`

Latitud (deg).

Definicion en la linea 390 del archivo location_system.cpp.

#### 11.4.3.17. len

`unsigned short NAV_PVT::len`

Definicion en la linea 372 del archivo location_system.cpp.

#### 11.4.3.18. lon

`long NAV_PVT::lon`

Longitud (deg).

Definicion en la linea 389 del archivo location_system.cpp.

#### 11.4.3.19. minute

`unsigned char NAV_PVT::minute`

Minutos de la hora, rango 0..59 (UTC)

Definicion en la linea 379 del archivo location_system.cpp.

#### 11.4.3.20. month

`unsigned char NAV_PVT::month`

Mes, rango 1..12

Definicion en la linea 376 del archivo location_system.cpp.

#### 11.4.3.21. nano

`long NAV_PVT::nano`

Fraccion de segundo, rango -1e9 .. 1e9 (UTC) (ns)

Definicion en la linea 383 del archivo location_system.cpp.

#### 11.4.3.22. numSV

`unsigned char NAV_PVT::numSV`

Numero de satelites.

Definicion en la linea 387 del archivo location_system.cpp.

#### 11.4.3.23. pDOP

`unsigned short NAV_PVT::pDOP`

Incertidumbre de posicionamiento (hdop).

Definicion en la linea 403 del archivo location_system.cpp.

#### 11.4.3.24. reserved1

`unsigned char NAV_PVT::reserved1`

reservado

Definicion en la linea 386 del archivo location_system.cpp.

#### 11.4.3.25. reserved2

`short NAV_PVT::reserved2`

Reservada

Definicion en la linea 404 del archivo location_system.cpp.

#### 11.4.3.26. reserved3

`unsigned long NAV_PVT::reserved3`

Reservada.

Definicion en la linea 405 del archivo location_system.cpp.

#### 11.4.3.27. sAcc

`unsigned long NAV_PVT::sAcc`

Precision estimada de la velocidad.

Definicion en la linea 401 del archivo location_system.cpp.

#### 11.4.3.28. second

`unsigned char NAV_PVT::second`

Segundos del minuto, rango 0..60 (UTC)

Definicion en la linea 380 del archivo location_system.cpp.

#### 11.4.3.29. tAcc

`unsigned long NAV_PVT::tAcc`

Precision estimada de la hora (UTC) (ns)

Definicion en la linea 382 del archivo location_system.cpp.

#### 11.4.3.30. vAcc

`unsigned long NAV_PVT::vAcc`

Precision vertical estimada (mm)

Definicion en la linea 394 del archivo location_system.cpp.

#### 11.4.3.31. valid

`char NAV_PVT::valid`

Validity Flags (see graphic below)

Definicion en la linea 381 del archivo location_system.cpp.

#### 11.4.3.32. velD

`long NAV_PVT::velD`

NED down velocity (mm/s)

Definicion en la linea 398 del archivo location_system.cpp.

#### 11.4.3.33. velE

`long NAV_PVT::velE`

NED east velocity (mm/s)

Definicion en la linea 397 del archivo location_system.cpp.

#### 11.4.3.34. velN

`long NAV_PVT::velN`

NED north velocity (mm/s)

Definicion en la linea 396 del archivo location_system.cpp.

#### 11.4.3.35. year

`unsigned short NAV_PVT::year`

Anio (UTC)

Definicion en la linea 375 del archivo location_system.cpp.

La documentacion para esta estructura fue generada a partir del siguiente fichero:

- location_system.cpp

### 11.4.4. Referencia del Archivo location_system.cpp

En este modulo se realizan todas las funciones que el programa principal necesita para un correcto funcionamiento del sistema.

```
#include <Arduino.h>
#include <HardwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "location_system.h"
#include "location_system_hc05.h"
#include "location_system_gsm_gprs.h"
```

#### Estructuras de datos

- struct NAV_PVT

#### defines

- #define LOCATION_SYSTEM_STRING_MAX_LENGHT (50U)
- #define LOCATION_SYSTEM_USER_NAME_LENGHT (15U)
- #define LOCATION_SYSTEM_PIN_NUMBER_LENGHT (6U)
- #define LOCATION_SYSTEM_USER_EMAIL_KEY_LENGHT (20U)
- #define LOCATION_SYSTEM_EMAIL_SERVER_LENGHT (20U)
- #define LOCATION_SYSTEM_EMAIL_PORT_LENGHT (5U)
- #define LOCATION_SYSTEM_EMAIL_RMTE_LENGHT (30U)
- #define LOCATION_SYSTEM_EMAILDST_NAME_LENGHT (20U)
- #define LOCATION_SYSTEM_EMAILDST_LENGHT (30U)
- #define LOCATION_SYSTEM_HOME_LAT_LENGHT (15U)
- #define LOCATION_SYSTEM_HOME_LONG_LENGHT (15U)
- #define LOCATION_SYSTEM_MOBILE_APN_NAME_LENGHT (10U)
- #define LOCATION_SYSTEM_MOBILE_APN_LENGHT (20U)
- #define LOCATION_SYSTEM_PUBLIC_IP_LENGHT (20U)
- #define LOCATION_SYSTEM_PHONE_NUMBER_LENGHT (10U)
- #define LOCATION_SYSTEM_BLUETOOTH_NAME_LENGHT (15U)
- #define LOCATION_SYSTEM_BLUETOOTH_PIN_LENGHT (6U)
- #define LOCATION_SYSTEM_DATETIME_LENGTH (20U)
- #define LOCATION_SYSTEM_URL_LENGHT (32U)

- #define LOCATION_SYSTEM_DATETIME_FORMAT "%04d-%02d-%02d-%02d-%02d-%02d"
- #define LOCATION_SYSTEM_STRING_VARIABLES (16U)
- #define LOCATION_SYSTEM_EEPROM_WRITTEN (5U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_USER_NAME_ADDRESS (0U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_PIN_NUMBER_ADDRESS (1U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_USER_EMAIL_KEY_ADDRESS (2U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_SERVER_ADDRESS (3U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_PORT_ADDRESS (4U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_RMTE_ADDRESS (5U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DEST_NAME_ADDRESS (6U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DST_ADDRESS (7U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_HOME_LAT_ADDRESS (8U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_HOME_LONG_ADDRESS (9U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_NAME_ADDRESS (10U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_ADDRESS (11U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_PUBLIC_IP_ADDRESS (12U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_PHONE_NUMBER_ADDRESS (13U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_NAME_ADDRESS (14U)
- #define LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_PIN_ADDRESS (15U)
- #define LOCATION_SYSTEM_EEPROM_USER_NAME_ADDRESS (16U)
- #define LOCATION_SYSTEM_EEPROM_PIN_NUMBER_ADDRESS (LOCATION_SYSTEM_EEPROM_USER + LOCATION_SYSTEM_USER_NAME_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_USER_EMAIL_KEY_ADDRESS (LOCATION_SYSTEM_EEPROM_P + LOCATION_SYSTEM_PIN_NUMBER_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_ADDRESS (LOCATION_SYSTEM_EEPROM_US + LOCATION_SYSTEM_USER_EMAIL_KEY_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_PORT_ADDRESS (LOCATION_SYSTEM_EEPRO + LOCATION_SYSTEM_EMAIL_SERVER_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_EMAIL_RMTE_ADDRESS (LOCATION_SYSTEM_EEPROM_EMAIL + LOCATION_SYSTEM_EMAIL_PORT_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_EMAIL_DEST_NAME_ADDRESS (LOCATION_SYSTEM_EEPROM + LOCATION_SYSTEM_EMAIL_RMTE_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_EMAIL_DST_ADDRESS (LOCATION_SYSTEM_EEPROM_EMAIL_ + LOCATION_SYSTEM_EMAILDST_NAME_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_HOME_LAT_ADDRESS (LOCATION_SYSTEM_EEPROM_EMAIL_D + LOCATION_SYSTEM_EMAILDST_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_HOME_LONG_ADDRESS (LOCATION_SYSTEM_EEPROM_HOME + LOCATION_SYSTEM_HOME_LAT_LENGHT)

- #define LOCATION_SYSTEM_EEPROM_MOBILE_APN_NAME_ADDRESS (LOCATION_SYSTEM_EEPROM_HO + LOCATION_SYSTEM_HOME_LONG_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_MOBILE_APN_ADDRESS (LOCATION_SYSTEM_EEPROM_MOBILE_A + LOCATION_SYSTEM_MOBILE_APN_NAME_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_PUBLIC_IP_ADDRESS (LOCATION_SYSTEM_EEPROM_MOBILE_API + LOCATION_SYSTEM_MOBILE_APN_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_PHONE_NUMBER_ADDRESS (LOCATION_SYSTEM_EEPROM_PUBL + LOCATION_SYSTEM_PUBLIC_IP_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_BLUETOOTH_NAME_ADDRESS (LOCATION_SYSTEM_EEPROM_PH + LOCATION_SYSTEM_PHONE_NUMBER_LENGHT)
- #define LOCATION_SYSTEM_EEPROM_BLUETOOTH_PIN_ADDRESS (LOCATION_SYSTEM_EEPROM_BLUE + LOCATION_SYSTEM_BLUETOOTH_NAME_LENGHT)
- #define LOCATION_SYSTEM_GPS_2D_FIX_TYPE (2U)
- #define LOCATION_SYSTEM_GPS_3D_FIX_TYPE (3U)
- #define LOCATION_SYSTEM_GPS_OLD_BAUDRATE (9600U)
- #define LOCATION_SYSTEM_GPS_NEW_BAUDRATE (57600U)
- #define LOCATION_SYSTEM_EMAIL_LENGHT (85U)
- #define LOCATION_SYSTEM_EMAIL_TIMEOUT (10U)
- #define LOCATION_SYSTEM_TO (20U)
- #define LOCATION_SYSTEM_BLUETOOTTH_BUTTON_TRESHOLD (250U)
- #define LOCATION_SYSTEM_EMAIL_BUTTON_TRESHOLD (250U)

**Funciones**

- void location_system_initialize (void)

  *Inicializa el sistema de localizacion.*
- void location_system_send_email (void)

  *Esta funcion realiza las comprobacioes necesarias para saber si el modulo GPRS esta conectado. En caso de estar conectado, envia el E-mail.*
- void location_system_buttons_pressed (void)

  *Esta funcion detecta si alguno de los botones ha sido presionado.*
- void location_system_email_interrupt (void)

  *Interrupcion del pulsador del E-mail.*
- void location_system_bluetooth_interrupt (void)

  *Interrupcion del pulsador del bluetooth.*
- void location_system_bluetooth_receiver (void)

  *Recibe los caracteres por bluetooth.*
- void location_system_isr_timer (void)

  *Cada vez que salta la interrupcion del timer se acude a esta funcion.*

- void location_system_1_second_timing (void)

  *Esta funcion se encarga de llevar el tiempo.*

- void location_system_get_coordinates (void)

  *Obtiene las coordenadas GPS.*

**Variables**

- LiquidCrystal_I2C lcd
- volatile location_system_email_state_t location_system_email_mode
- volatile byte location_system_bluetooth_state
- volatile location_system_bluetooth_state_t location_system_bluetooth_mode
- const unsigned char UBX_HEADER [2] = { 0xB5, 0x62 }
- const unsigned char UBLOX_INIT [] PROGMEM
- NAV_PVT pvt

### 11.4.5. Descripcion detallada

En este modulo se realizan todas las funciones que el programa principal necesita para un correcto funcionamiento del sistema.

**Version**

1.0

**Fecha**

11/11/2017

**Autor**

Cristobal Garcia Camoira

Definicion en el archivo location_system.cpp.

#### 11.4.5.1. Documentacion de los 'defines'

#### 11.4.5.2. LOCATION_SYSTEM_BLUETOOTH_NAME_LENGHT

```
#define LOCATION_SYSTEM_BLUETOOTH_NAME_LENGHT (15U)
```

Longitud del nombre del bluetooth que le deseamos poner a nuestro dispositivo

Definicion en la linea 54 del archivo location_system.cpp.

#### 11.4.5.3. LOCATION_SYSTEM_BLUETOOTH_PIN_LENGHT

```
#define LOCATION_SYSTEM_BLUETOOTH_PIN_LENGHT (6U)
```

Longitud del pin del bluetooth

Definicion en la linea 56 del archivo location_system.cpp.

#### 11.4.5.4. LOCATION_SYSTEM_BLUETOOTTH_BUTTON_TRESHOLD

```
#define LOCATION_SYSTEM_BLUETOOTTH_BUTTON_TRESHOLD (250U)
```

Tiempo antirrebote del pulsador del bluetooth

Definicion en la linea 146 del archivo location_system.cpp.

#### 11.4.5.5. LOCATION_SYSTEM_DATETIME_FORMAT

```
#define LOCATION_SYSTEM_DATETIME_FORMAT "%04d-%02d-%02d-%02d-%02d-%02d"
```

Formato de la fecha y la hora

Definicion en la linea 62 del archivo location_system.cpp.

#### 11.4.5.6. LOCATION_SYSTEM_DATETIME_LENGTH

```
#define LOCATION_SYSTEM_DATETIME_LENGTH (20U)
```

Longitud de la fecha y la hora

Definicion en la linea 58 del archivo location_system.cpp.

#### 11.4.5.7. LOCATION_SYSTEM_EEPROM_BLUETOOTH_NAME_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_BLUETOOTH_NAME_ADDRESS (LOCATION_SYSTEM_EEPROM_PHONE_NUMBER_ADDR
+ LOCATION_SYSTEM_PHONE_NUMBER_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 128 del archivo location_system.cpp.

#### 11.4.5.8. LOCATION_SYSTEM_EEPROM_BLUETOOTH_PIN_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_BLUETOOTH_PIN_ADDRESS (LOCATION_SYSTEM_EEPROM_BLUETOOTH_NAME_ADD
+ LOCATION_SYSTEM_BLUETOOTH_NAME_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 130 del archivo location_system.cpp.

#### 11.4.5.9. LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_NAME_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_NAME_ADDRESS (14U)
```

Direcciones de la EEPROM

Definicion en la linea 96 del archivo location_system.cpp.

#### 11.4.5.10. LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_PIN_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_PIN_ADDRESS (15U)
```

Direcciones de la EEPROM

Definicion en la linea 98 del archivo location_system.cpp.

#### 11.4.5.11. LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DEST_NAME_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DEST_NAME_ADDRESS (6U)
```

Direcciones de la EEPROM

Definicion en la linea 80 del archivo location_system.cpp.

#### 11.4.5.12. LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DST_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DST_ADDRESS (7U)
```

Direcciones de la EEPROM

Definicion en la linea 82 del archivo location_system.cpp.

#### 11.4.5.13. LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_PORT_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_PORT_ADDRESS (4U)
```

Direcciones de la EEPROM

Definicion en la linea 76 del archivo location_system.cpp.

#### 11.4.5.14. LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_RMTE_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_RMTE_ADDRESS (5U)
```

Direcciones de la EEPROM

Definicion en la linea 78 del archivo location_system.cpp.

#### 11.4.5.15. LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_SERVER_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_SERVER_ADDRESS (3U)
```

Direcciones de la EEPROM

Definicion en la linea 74 del archivo location_system.cpp.

#### 11.4.5.16. LOCATION_SYSTEM_EEPROM_CHECK_HOME_LAT_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_HOME_LAT_ADDRESS (8U)
```

Direcciones de la EEPROM

Definicion en la linea 84 del archivo location_system.cpp.

#### 11.4.5.17. LOCATION_SYSTEM_EEPROM_CHECK_HOME_LONG_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_HOME_LONG_ADDRESS (9U)
```

Direcciones de la EEPROM

Definicion en la linea 86 del archivo location_system.cpp.

#### 11.4.5.18. LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_ADDRESS (11U)
```

Direcciones de la EEPROM

Definicion en la linea 90 del archivo location_system.cpp.

#### 11.4.5.19. LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_NAME_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_NAME_ADDRESS (10U)
```

Direcciones de la EEPROM

Definicion en la linea 88 del archivo location_system.cpp.

#### 11.4.5.20. LOCATION_SYSTEM_EEPROM_CHECK_PHONE_NUMBER_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_PHONE_NUMBER_ADDRESS (13U)
```

Direcciones de la EEPROM

Definicion en la linea 94 del archivo location_system.cpp.

#### 11.4.5.21. LOCATION_SYSTEM_EEPROM_CHECK_PIN_NUMBER_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_PIN_NUMBER_ADDRESS (1U)
```

Direcciones de la EEPROM

Definicion en la linea 70 del archivo location_system.cpp.

#### 11.4.5.22. LOCATION_SYSTEM_EEPROM_CHECK_PUBLIC_IP_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_PUBLIC_IP_ADDRESS (12U)
```

Direcciones de la EEPROM

Definicion en la linea 92 del archivo location_system.cpp.

#### 11.4.5.23. LOCATION_SYSTEM_EEPROM_CHECK_USER_EMAIL_KEY_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_USER_EMAIL_KEY_ADDRESS (2U)
```

Direcciones de la EEPROM

Definicion en la linea 72 del archivo location_system.cpp.

#### 11.4.5.24. LOCATION_SYSTEM_EEPROM_CHECK_USER_NAME_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_CHECK_USER_NAME_ADDRESS (0U)
```

Direcciones de la EEPROM

Definicion en la linea 68 del archivo location_system.cpp.

#### 11.4.5.25. LOCATION_SYSTEM_EEPROM_EMAIL_DEST_NAME_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_EMAIL_DEST_NAME_ADDRESS (LOCATION_SYSTEM_EEPROM_EMAIL_RMTE_AD
+ LOCATION_SYSTEM_EMAIL_RMTE_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 112 del archivo location_system.cpp.

#### 11.4.5.26. LOCATION_SYSTEM_EEPROM_EMAIL_DST_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_EMAIL_DST_ADDRESS (LOCATION_SYSTEM_EEPROM_EMAIL_DEST_NAME_ADD
+ LOCATION_SYSTEM_EMAILDST_NAME_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 114 del archivo location_system.cpp.

#### 11.4.5.27. LOCATION_SYSTEM_EEPROM_EMAIL_RMTE_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_EMAIL_RMTE_ADDRESS (LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_PORT_ADD
+ LOCATION_SYSTEM_EMAIL_PORT_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 110 del archivo location_system.cpp.

#### 11.4.5.28. LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_ADDRESS (LOCATION_SYSTEM_EEPROM_USER_EMAIL_KEY_ADDR
+ LOCATION_SYSTEM_USER_EMAIL_KEY_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 106 del archivo location_system.cpp.

#### 11.4.5.29. LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_PORT_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_PORT_ADDRESS (LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_A
+ LOCATION_SYSTEM_EMAIL_SERVER_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 108 del archivo location_system.cpp.

#### 11.4.5.30. LOCATION_SYSTEM_EEPROM_HOME_LAT_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_HOME_LAT_ADDRESS (LOCATION_SYSTEM_EEPROM_EMAIL_DST_ADDRESS
+ LOCATION_SYSTEM_EMAILDST_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 116 del archivo location_system.cpp.

#### 11.4.5.31. LOCATION_SYSTEM_EEPROM_HOME_LONG_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_HOME_LONG_ADDRESS (LOCATION_SYSTEM_EEPROM_HOME_LAT_ADDRESS
+ LOCATION_SYSTEM_HOME_LAT_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 118 del archivo location_system.cpp.

#### 11.4.5.32. LOCATION_SYSTEM_EEPROM_MOBILE_APN_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_MOBILE_APN_ADDRESS (LOCATION_SYSTEM_EEPROM_MOBILE_APN_NAME_ADDRE
+ LOCATION_SYSTEM_MOBILE_APN_NAME_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 122 del archivo location_system.cpp.

#### 11.4.5.33. LOCATION_SYSTEM_EEPROM_MOBILE_APN_NAME_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_MOBILE_APN_NAME_ADDRESS (LOCATION_SYSTEM_EEPROM_HOME_LONG_ADD
+ LOCATION_SYSTEM_HOME_LONG_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 120 del archivo location_system.cpp.

#### 11.4.5.34. LOCATION_SYSTEM_EEPROM_PHONE_NUMBER_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_PHONE_NUMBER_ADDRESS (LOCATION_SYSTEM_EEPROM_PUBLIC_IP_ADDRES
+ LOCATION_SYSTEM_PUBLIC_IP_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 126 del archivo location_system.cpp.

#### 11.4.5.35. LOCATION_SYSTEM_EEPROM_PIN_NUMBER_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_PIN_NUMBER_ADDRESS (LOCATION_SYSTEM_EEPROM_USER_NAME_ADDRESS
+ LOCATION_SYSTEM_USER_NAME_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 102 del archivo location_system.cpp.

#### 11.4.5.36. LOCATION_SYSTEM_EEPROM_PUBLIC_IP_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_PUBLIC_IP_ADDRESS (LOCATION_SYSTEM_EEPROM_MOBILE_APN_ADDRESS
+ LOCATION_SYSTEM_MOBILE_APN_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 124 del archivo location_system.cpp.

#### 11.4.5.37. LOCATION_SYSTEM_EEPROM_USER_EMAIL_KEY_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_USER_EMAIL_KEY_ADDRESS (LOCATION_SYSTEM_EEPROM_PIN_NUMBER_ADD
+ LOCATION_SYSTEM_PIN_NUMBER_LENGHT)
```

Direcciones de la EEPROM

Definicion en la linea 104 del archivo location_system.cpp.

#### 11.4.5.38. LOCATION_SYSTEM_EEPROM_USER_NAME_ADDRESS

```
#define LOCATION_SYSTEM_EEPROM_USER_NAME_ADDRESS (16U)
```

Direcciones de la EEPROM

Definicion en la linea 100 del archivo location_system.cpp.

#### 11.4.5.39. LOCATION_SYSTEM_EEPROM_WRITTEN

```
#define LOCATION_SYSTEM_EEPROM_WRITTEN (5U)
```

Valor que se escribe en la EEPROM para verificar que la variable esta escrita

Definicion en la linea 66 del archivo location_system.cpp.

#### 11.4.5.40. LOCATION_SYSTEM_EMAIL_BUTTON_TRESHOLD

```
#define LOCATION_SYSTEM_EMAIL_BUTTON_TRESHOLD (250U)
```

Tiempo antirrebote del pulsador del e mail

Definicion en la linea 148 del archivo location_system.cpp.

#### 11.4.5.41. LOCATION_SYSTEM_EMAIL_LENGHT

```
#define LOCATION_SYSTEM_EMAIL_LENGHT (85U)
```

Longitud del E-mail

Definicion en la linea 140 del archivo location_system.cpp.

#### 11.4.5.42. LOCATION_SYSTEM_EMAIL_PORT_LENGHT

```
#define LOCATION_SYSTEM_EMAIL_PORT_LENGHT (5U)
```

Longitud del puerto SMTP por el que se envia el email

Definicion en la linea 34 del archivo location_system.cpp.

#### 11.4.5.43. LOCATION_SYSTEM_EMAIL_RMTE_LENGHT

```
#define LOCATION_SYSTEM_EMAIL_RMTE_LENGHT (30U)
```

Longitud del correo electronico del usuario (remite)

Definicion en la linea 36 del archivo location_system.cpp.

#### 11.4.5.44. LOCATION_SYSTEM_EMAIL_SERVER_LENGHT

```
#define LOCATION_SYSTEM_EMAIL_SERVER_LENGHT (20U)
```

Longitud del servidor de correo electronico

Definicion en la linea 32 del archivo location_system.cpp.

#### 11.4.5.45. LOCATION_SYSTEM_EMAIL_TIMEOUT

```
#define LOCATION_SYSTEM_EMAIL_TIMEOUT (10U)
```

Tiempo que debe transcurrir entre pulsacion y pulsacion del boton de envio del E-mail

Definicion en la linea 142 del archivo location_system.cpp.

#### 11.4.5.46. LOCATION_SYSTEM_EMAILDST_LENGHT

```
#define LOCATION_SYSTEM_EMAILDST_LENGHT (30U)
```

Longitud del correo electronico del remitente

Definicion en la linea 40 del archivo location_system.cpp.

#### 11.4.5.47. LOCATION_SYSTEM_EMAILDST_NAME_LENGHT

```
#define LOCATION_SYSTEM_EMAILDST_NAME_LENGHT (20U)
```

Longitud del nombre de la persona que recibira el correo electronico

Definicion en la linea 38 del archivo location_system.cpp.

#### 11.4.5.48. LOCATION_SYSTEM_GPS_2D_FIX_TYPE

```
#define LOCATION_SYSTEM_GPS_2D_FIX_TYPE (2U)
```

Valor que indica que el GPS ha encontrado la posicion

Definicion en la linea 132 del archivo location_system.cpp.

#### 11.4.5.49. LOCATION_SYSTEM_GPS_3D_FIX_TYPE

```
#define LOCATION_SYSTEM_GPS_3D_FIX_TYPE (3U)
```

Valor que indica que el GPS ha encontrado la posicion

Definicion en la linea 134 del archivo location_system.cpp.

#### 11.4.5.50. LOCATION_SYSTEM_GPS_NEW_BAUDRATE

```
#define LOCATION_SYSTEM_GPS_NEW_BAUDRATE (57600U)
```

Nueva velocidad de transmision del puerto serie al que va conectado el GPS tras la configuracion inicial

Definicion en la linea 138 del archivo location_system.cpp.

#### 11.4.5.51. LOCATION_SYSTEM_GPS_OLD_BAUDRATE

```
#define LOCATION_SYSTEM_GPS_OLD_BAUDRATE (9600U)
```

Velocidad de transmision del puerto serie al que va conectado el GPS

Definicion en la linea 136 del archivo location_system.cpp.

#### 11.4.5.52. LOCATION_SYSTEM_HOME_LAT_LENGHT

```
#define LOCATION_SYSTEM_HOME_LAT_LENGHT (15U)
```

Longitud de la latitud del punto de partida

Definicion en la linea 42 del archivo location_system.cpp.

#### 11.4.5.53. LOCATION_SYSTEM_HOME_LONG_LENGHT

```
#define LOCATION_SYSTEM_HOME_LONG_LENGHT (15U)
```

Longitud de la longitud del punto de partida

Definicion en la linea 44 del archivo location_system.cpp.

#### 11.4.5.54. LOCATION_SYSTEM_MOBILE_APN_LENGHT

```
#define LOCATION_SYSTEM_MOBILE_APN_LENGHT (20U)
```

Longitud del del servidor APN de nuestra compania telefonica

Definicion en la linea 48 del archivo location_system.cpp.

#### 11.4.5.55. LOCATION_SYSTEM_MOBILE_APN_NAME_LENGHT

```
#define LOCATION_SYSTEM_MOBILE_APN_NAME_LENGHT (10U)
```

Longitud del nombre del APN de nuestra compania telefonica

Definicion en la linea 46 del archivo location_system.cpp.

#### 11.4.5.56. LOCATION_SYSTEM_PHONE_NUMBER_LENGHT

```
#define LOCATION_SYSTEM_PHONE_NUMBER_LENGHT (10U)
```

Longitud de nuestro numero de telefono

Definicion en la linea 52 del archivo location_system.cpp.

#### 11.4.5.57. LOCATION_SYSTEM_PIN_NUMBER_LENGHT

```
#define LOCATION_SYSTEM_PIN_NUMBER_LENGHT (6U)
```

Longitud del numero PIN

Definicion en la linea 28 del archivo location_system.cpp.

#### 11.4.5.58. LOCATION_SYSTEM_PUBLIC_IP_LENGHT

```
#define LOCATION_SYSTEM_PUBLIC_IP_LENGHT (20U)
```

Longitud de la IP publica de nuestro ordenador presonal o servidor

Definicion en la linea 50 del archivo location_system.cpp.

#### 11.4.5.59. LOCATION_SYSTEM_STRING_MAX_LENGHT

```
#define LOCATION_SYSTEM_STRING_MAX_LENGHT (50U)
```

Longitud maxima de las tramas del bluetooth

Definicion en la linea 24 del archivo location_system.cpp.

#### 11.4.5.60. LOCATION_SYSTEM_STRING_VARIABLES

```
#define LOCATION_SYSTEM_STRING_VARIABLES (16U)
```

Numero de variables que se envian por bluetooth

Definicion en la linea 64 del archivo location_system.cpp.

#### 11.4.5.61. LOCATION_SYSTEM_TO

```
#define LOCATION_SYSTEM_TO (20U)
```

Tiempo que el modem SIM800L va a esperar para enviar el E-mail en caso de que no se escriban todos los caracteres

Definicion en la linea 144 del archivo location_system.cpp.

#### 11.4.5.62. LOCATION_SYSTEM_URL_LENGHT

```
#define LOCATION_SYSTEM_URL_LENGHT (32U)
```

Longitud de la direccion web a la que se enviaran los datos de posicionamiento

Definicion en la linea 60 del archivo location_system.cpp.

#### 11.4.5.63. LOCATION_SYSTEM_USER_EMAIL_KEY_LENGHT

```
#define LOCATION_SYSTEM_USER_EMAIL_KEY_LENGHT (20U)
```

Longitud de la clave de la direccion de correo electronico

Definicion en la linea 30 del archivo location_system.cpp.

#### 11.4.5.64. LOCATION_SYSTEM_USER_NAME_LENGHT

```
#define LOCATION_SYSTEM_USER_NAME_LENGHT (15U)
```

Longitud del nombre de usuario

Definicion en la linea 26 del archivo location_system.cpp.

### 11.4.6. Documentacion de las funciones

#### 11.4.6.1. location_system_1_second_timing()

```
void location_system_1_second_timing (
        void  )
```

Esta funcion se encarga de llevar el tiempo.

Definicion en la linea 884 del archivo location_system.cpp.

#### 11.4.6.2. location_system_bluetooth_interrupt()

```
void location_system_bluetooth_interrupt (
        void  )
```

Interrupcion del pulsador del bluetooth.

Definicion en la linea 828 del archivo location_system.cpp.

#### 11.4.6.3. location_system_bluetooth_receiver()

```
void location_system_bluetooth_receiver (
        void  )
```

Recibe los caracteres por bluetooth.

Definicion en la linea 843 del archivo location_system.cpp.

#### 11.4.6.4. location_system_buttons_pressed()

```
void location_system_buttons_pressed (
        void  )
```

Esta funcion detecta si alguno de los botones ha sido presionado.

Definicion en la linea 784 del archivo location_system.cpp.

#### 11.4.6.5. location_system_email_interrupt()

```
void location_system_email_interrupt (
        void  )
```

Interrupcion del pulsador del E-mail.

Definicion en la linea 815 del archivo location_system.cpp.

#### 11.4.6.6. location_system_get_coordinates()

```
void location_system_get_coordinates (
        void  )
```

Obtiene las coordenadas GPS.

Definicion en la linea 915 del archivo location_system.cpp.

#### 11.4.6.7. location_system_initialize()

```
void location_system_initialize (
        void  )
```

Inicializa el sistema de localizacion.

Definicion en la linea 672 del archivo location_system.cpp.

#### 11.4.6.8. location_system_isr_timer()

```
void location_system_isr_timer (
        void  )
```

Cada vez que salta la interrupcion del timer se acude a esta funcion.

Definicion en la linea 875 del archivo location_system.cpp.

#### 11.4.6.9. location_system_send_email()

```
void location_system_send_email (
        void  )
```

Esta funcion realiza las comprobacioes necesarias para saber si el modulo GPRS esta conectado. En caso de esatr conectado, envia el E-mail.

Definicion en la linea 735 del archivo location_system.cpp.

### 11.4.7. Documentacion de las variables

#### 11.4.7.1. lcd

```
LiquidCrystal_I2C lcd
```

#### 11.4.7.2. location_system_bluetooth_mode

```
volatile location_system_bluetooth_state_t location_system_bluetooth_mode
```

Definicion en la linea 156 del archivo location_system.cpp.

#### 11.4.7.3. location_system_bluetooth_state

```
volatile byte location_system_bluetooth_state
```

Definicion en la linea 154 del archivo location_system.cpp.

#### 11.4.7.4. location_system_email_mode

```
volatile location_system_email_state_t location_system_email_mode
```

Definicion en la linea 152 del archivo location_system.cpp.

#### 11.4.7.5. PROGMEM

```
const unsigned char UBLOX_INIT [] PROGMEM
```

**Valor inicial:**

```
=
{

        0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x24,
        0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x01,0x00,0x00,0x00,0x00,0x00,0x01,0x01,0x2B,
        0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x02,0x00,0x00,0x00,0x00,0x00,0x01,0x02,0x32,
        0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x03,0x00,0x00,0x00,0x00,0x00,0x01,0x03,0x39,
        0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x04,0x00,0x00,0x00,0x00,0x00,0x01,0x04,0x40,
        0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x05,0x00,0x00,0x00,0x00,0x00,0x01,0x05,0x47,
```

```
        0xB5,0x62,0x06,0x01,0x08,0x00,0x01,0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x12,0xB9,
        0xB5,0x62,0x06,0x01,0x08,0x00,0x01,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x13,0xC0,

        0xB5,0x62,0x06,0x01,0x08,0x00,0x01,0x07,0x00,0x01,0x00,0x00,0x00,0x00,0x18,0xE1,




        0xB5,0x62,0x06,0x08,0x06,0x00,0xE8,0x03,0x01,0x00,0x01,0x00,0x01,0x39,

        0xB5,0x62,0x06,0x00,0x14,0x00,0x01,0x00,0x00,0x00,0xD0,0x08,0x00,0x00,0x00,0xE1,0x00,0x00,0x07,0x00
      ,0x03,0x00,0x00,0x00,0x00,0x00,0xDE,0xC9
}
```

Definicion en la linea 338 del archivo location_system.cpp.

#### 11.4.7.6. pvt

```
NAV_PVT pvt
```

Definicion en la linea 408 del archivo location_system.cpp.

#### 11.4.7.7. UBX_HEADER

```
const unsigned char UBX_HEADER[2] = { 0xB5, 0x62 }
```

Cabecera de los mensajes que van a ser recibidos

Definicion en la linea 336 del archivo location_system.cpp.

### 11.4.8. location_system.cpp

```
00001
00010 #include <Arduino.h>
00011 #include <HardwareSerial.h>
00012 #include <LiquidCrystal_I2C.h>
00013 #include <EEPROM.h>
00014 #include <stdio.h>
00015 #include <string.h>
00016 #include <stdlib.h>
00017 #include "location_system.h"
00018 #include "location_system_hc05.h"
00019 #include "location_system_gsm_gprs.h"
00020
00021 extern LiquidCrystal_I2C lcd;
00022
00024 #define LOCATION_SYSTEM_STRING_MAX_LENGHT       (50U)
00025
```

```
00026 #define LOCATION_SYSTEM_USER_NAME_LENGHT        (15U)
00027
00028 #define LOCATION_SYSTEM_PIN_NUMBER_LENGHT       (6U)
00029
00030 #define LOCATION_SYSTEM_USER_EMAIL_KEY_LENGHT   (20U)
00031
00032 #define LOCATION_SYSTEM_EMAIL_SERVER_LENGHT     (20U)
00033
00034 #define LOCATION_SYSTEM_EMAIL_PORT_LENGHT       (5U)
00035
00036 #define LOCATION_SYSTEM_EMAIL_RMTE_LENGHT       (30U)
00037
00038 #define LOCATION_SYSTEM_EMAILDST_NAME_LENGHT    (20U)
00039
00040 #define LOCATION_SYSTEM_EMAILDST_LENGHT         (30U)
00041
00042 #define LOCATION_SYSTEM_HOME_LAT_LENGHT         (15U)
00043
00044 #define LOCATION_SYSTEM_HOME_LONG_LENGHT        (15U)
00045
00046 #define LOCATION_SYSTEM_MOBILE_APN_NAME_LENGHT  (10U)
00047
00048 #define LOCATION_SYSTEM_MOBILE_APN_LENGHT       (20U)
00049
00050 #define LOCATION_SYSTEM_PUBLIC_IP_LENGHT        (20U)
00051
00052 #define LOCATION_SYSTEM_PHONE_NUMBER_LENGHT     (10U)
00053
00054 #define LOCATION_SYSTEM_BLUETOOTH_NAME_LENGHT   (15U)
00055
00056 #define LOCATION_SYSTEM_BLUETOOTH_PIN_LENGHT    (6U)
00057
00058 #define LOCATION_SYSTEM_DATETIME_LENGTH         (20U)
00059
00060 #define LOCATION_SYSTEM_URL_LENGHT              (32U)
00061
00062 #define LOCATION_SYSTEM_DATETIME_FORMAT "%04d-%02d-%02d-%02d-%02d-%02d"
00063
00064 #define LOCATION_SYSTEM_STRING_VARIABLES    (16U)
00065
00066 #define LOCATION_SYSTEM_EEPROM_WRITTEN    (5U)
00067
00068 #define LOCATION_SYSTEM_EEPROM_CHECK_USER_NAME_ADDRESS      (0U)
00069
00070 #define LOCATION_SYSTEM_EEPROM_CHECK_PIN_NUMBER_ADDRESS     (1U)
00071
00072 #define LOCATION_SYSTEM_EEPROM_CHECK_USER_EMAIL_KEY_ADDRESS   (2U)
00073
00074 #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_SERVER_ADDRESS   (3U)
00075
00076 #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_PORT_ADDRESS     (4U)
00077
00078 #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_RMTE_ADDRESS     (5U)
00079
00080 #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DEST_NAME_ADDRESS  (6U)
00081
00082 #define LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DST_ADDRESS      (7U)
00083
```

```
00084 #define LOCATION_SYSTEM_EEPROM_CHECK_HOME_LAT_ADDRESS     (8U)
00085
00086 #define LOCATION_SYSTEM_EEPROM_CHECK_HOME_LONG_ADDRESS      (9U)
00087
00088 #define LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_NAME_ADDRESS  (10U)
00089
00090 #define LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_ADDRESS     (11U)
00091
00092 #define LOCATION_SYSTEM_EEPROM_CHECK_PUBLIC_IP_ADDRESS      (12U)
00093
00094 #define LOCATION_SYSTEM_EEPROM_CHECK_PHONE_NUMBER_ADDRESS   (13U)
00095
00096 #define LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_NAME_ADDRESS   (14U)
00097
00098 #define LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_PIN_ADDRESS    (15U)
00099
00100 #define LOCATION_SYSTEM_EEPROM_USER_NAME_ADDRESS      (16U)
00101
00102 #define LOCATION_SYSTEM_EEPROM_PIN_NUMBER_ADDRESS     (LOCATION_SYSTEM_EEPROM_USER_NAME_ADDRESS +
      LOCATION_SYSTEM_USER_NAME_LENGHT)
00103
00104 #define LOCATION_SYSTEM_EEPROM_USER_EMAIL_KEY_ADDRESS   (LOCATION_SYSTEM_EEPROM_PIN_NUMBER_ADDRESS +
      LOCATION_SYSTEM_PIN_NUMBER_LENGHT)
00105
00106 #define LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_ADDRESS     (LOCATION_SYSTEM_EEPROM_USER_EMAIL_KEY_ADDRES
      LOCATION_SYSTEM_USER_EMAIL_KEY_LENGHT)
00107
00108 #define LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_PORT_ADDRESS  (LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_ADDRES
      LOCATION_SYSTEM_EMAIL_SERVER_LENGHT)
00109
00110 #define LOCATION_SYSTEM_EEPROM_EMAIL_RMTE_ADDRESS     (LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_PORT_ADDR
      LOCATION_SYSTEM_EMAIL_PORT_LENGHT)
00111
00112 #define LOCATION_SYSTEM_EEPROM_EMAIL_DEST_NAME_ADDRESS    (LOCATION_SYSTEM_EEPROM_EMAIL_RMTE_ADDRESS
      LOCATION_SYSTEM_EMAIL_RMTE_LENGHT)
00113
00114 #define LOCATION_SYSTEM_EEPROM_EMAIL_DST_ADDRESS      (LOCATION_SYSTEM_EEPROM_EMAIL_DEST_NAME_ADDRESS
      LOCATION_SYSTEM_EMAILDST_NAME_LENGHT)
00115
00116 #define LOCATION_SYSTEM_EEPROM_HOME_LAT_ADDRESS       (LOCATION_SYSTEM_EEPROM_EMAIL_DST_ADDRESS +
      LOCATION_SYSTEM_EMAILDST_LENGHT)
00117
00118 #define LOCATION_SYSTEM_EEPROM_HOME_LONG_ADDRESS      (LOCATION_SYSTEM_EEPROM_HOME_LAT_ADDRESS +
      LOCATION_SYSTEM_HOME_LAT_LENGHT)
00119
00120 #define LOCATION_SYSTEM_EEPROM_MOBILE_APN_NAME_ADDRESS    (LOCATION_SYSTEM_EEPROM_HOME_LONG_ADDRESS +
      LOCATION_SYSTEM_HOME_LONG_LENGHT)
00121
00122 #define LOCATION_SYSTEM_EEPROM_MOBILE_APN_ADDRESS     (LOCATION_SYSTEM_EEPROM_MOBILE_APN_NAME_ADDRESS
      LOCATION_SYSTEM_MOBILE_APN_NAME_LENGHT)
00123
00124 #define LOCATION_SYSTEM_EEPROM_PUBLIC_IP_ADDRESS      (LOCATION_SYSTEM_EEPROM_MOBILE_APN_ADDRESS +
      LOCATION_SYSTEM_MOBILE_APN_LENGHT)
00125
00126 #define LOCATION_SYSTEM_EEPROM_PHONE_NUMBER_ADDRESS     (LOCATION_SYSTEM_EEPROM_PUBLIC_IP_ADDRESS +
      LOCATION_SYSTEM_PUBLIC_IP_LENGHT)
00127
00128 #define LOCATION_SYSTEM_EEPROM_BLUETOOTH_NAME_ADDRESS   (LOCATION_SYSTEM_EEPROM_PHONE_NUMBER_ADDRESS
```

```
      LOCATION_SYSTEM_PHONE_NUMBER_LENGHT)
00129
00130 #define LOCATION_SYSTEM_EEPROM_BLUETOOTH_PIN_ADDRESS    (LOCATION_SYSTEM_EEPROM_BLUETOOTH_NAME_ADDRESS +
      LOCATION_SYSTEM_BLUETOOTH_NAME_LENGHT)
00131
00132 #define LOCATION_SYSTEM_GPS_2D_FIX_TYPE     (2U)
00133
00134 #define LOCATION_SYSTEM_GPS_3D_FIX_TYPE     (3U)
00135
00136 #define LOCATION_SYSTEM_GPS_OLD_BAUDRATE    (9600U)
00137
00138 #define LOCATION_SYSTEM_GPS_NEW_BAUDRATE    (57600U)
00139
00140 #define LOCATION_SYSTEM_EMAIL_LENGHT      (85U)
00141
00142 #define LOCATION_SYSTEM_EMAIL_TIMEOUT     (10U)
00143
00144 #define LOCATION_SYSTEM_TO                (20U)
00145
00146 #define  LOCATION_SYSTEM_BLUETOOTTH_BUTTON_TRESHOLD (250U)
00147
00148 #define  LOCATION_SYSTEM_EMAIL_BUTTON_TRESHOLD (250U)
00149
00150 static int32_t location_system_email_button_timestamp;
00151 /***/
00152 volatile location_system_email_state_t
      location_system_email_mode;
00153 /***/
00154 volatile byte location_system_bluetooth_state;
00155 /***/
00156 volatile location_system_bluetooth_state_t
      location_system_bluetooth_mode;
00158 static boolean location_system_flag_timer;
00160 static char location_system_url[60]={"http://api.thingspeak.com/update?api_key=ZLCUSQ0J0G681G7Z"};
00162 static char location_system_datetime[LOCATION_SYSTEM_DATETIME_LENGTH];
00164 static char location_system_response[LOCATION_SYSTEM_STRING_MAX_LENGHT];
00166 static char location_system_copy_response[LOCATION_SYSTEM_STRING_MAX_LENGHT
      ];
00168 static char location_system_user_name[LOCATION_SYSTEM_USER_NAME_LENGHT];
00170 static char location_system_pin_number[LOCATION_SYSTEM_PIN_NUMBER_LENGHT];
00172 static char location_system_email_key[LOCATION_SYSTEM_USER_EMAIL_KEY_LENGHT
      ];
00174 static char location_system_email_server[LOCATION_SYSTEM_EMAIL_SERVER_LENGHT
      ];
00176 static char location_system_email_port[LOCATION_SYSTEM_EMAIL_PORT_LENGHT];
00178 static char location_system_email_remitente[LOCATION_SYSTEM_EMAIL_RMTE_LENGHT
      ];
00180 static char location_system_email_dest_name[LOCATION_SYSTEM_EMAILDST_NAME_LENGHT
      ];
00182 static char location_system_email_destino[LOCATION_SYSTEM_EMAILDST_LENGHT];
00184 static char location_system_home_latitude[LOCATION_SYSTEM_HOME_LAT_LENGHT];
00186 static char location_system_home_longitude[LOCATION_SYSTEM_HOME_LONG_LENGHT
      ];
00188 static char location_system_mobile_APN_name[
      LOCATION_SYSTEM_MOBILE_APN_NAME_LENGHT];
00190 static char location_system_mobile_APN[LOCATION_SYSTEM_MOBILE_APN_LENGHT];
00192 static char location_system_public_ip[LOCATION_SYSTEM_PUBLIC_IP_LENGHT];
00194 static char location_system_phone_number[LOCATION_SYSTEM_PHONE_NUMBER_LENGHT
```

```
];
00196 static char location_system_bluetooth_name[LOCATION_SYSTEM_BLUETOOTH_NAME_LENGHT
];
00198 static char location_system_bluetooth_pin[LOCATION_SYSTEM_BLUETOOTH_PIN_LENGHT
];
00200 static char location_system_mail_subject[] = "INFO LOCALIZADOR";
00201
00222 static byte location_system_EEPROM_check[LOCATION_SYSTEM_STRING_VARIABLES];
00223
00225 static const char *location_system_strings[LOCATION_SYSTEM_STRING_VARIABLES
]=
00226 {
00227         "#usrname",
00228         "#usrphpin",
00229         "#usremailkey",
00230         "#emailserver",
00231         "#emailport",
00232         "#emailrmt",
00233         "#emaildstname",
00234         "#emaildst",
00235         "#usrhomelat",
00236         "#usrhomelong",
00237         "#apnname",
00238         "#usrapn",
00239         "#usrpublicip",
00240         "#usrphnumber",
00241         "#usrbtname",
00242         "#usrbtpin"
00243 };
00244
00246 static const uint8_t location_system_eeprom_check_bytes_address[
LOCATION_SYSTEM_STRING_VARIABLES]=
00247 {
00248         LOCATION_SYSTEM_EEPROM_CHECK_USER_NAME_ADDRESS,
00249         LOCATION_SYSTEM_EEPROM_CHECK_PIN_NUMBER_ADDRESS,
00250         LOCATION_SYSTEM_EEPROM_CHECK_USER_EMAIL_KEY_ADDRESS
,
00251         LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_SERVER_ADDRESS,
00252         LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_PORT_ADDRESS,
00253         LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_RMTE_ADDRESS,
00254         LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DEST_NAME_ADDRESS
,
00255         LOCATION_SYSTEM_EEPROM_CHECK_EMAIL_DST_ADDRESS,
00256         LOCATION_SYSTEM_EEPROM_CHECK_HOME_LAT_ADDRESS,
00257         LOCATION_SYSTEM_EEPROM_CHECK_HOME_LONG_ADDRESS,
00258         LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_NAME_ADDRESS
,
00259         LOCATION_SYSTEM_EEPROM_CHECK_MOBILE_APN_ADDRESS,
00260         LOCATION_SYSTEM_EEPROM_CHECK_PUBLIC_IP_ADDRESS,
00261         LOCATION_SYSTEM_EEPROM_CHECK_PHONE_NUMBER_ADDRESS,
00262         LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_NAME_ADDRESS
,
00263         LOCATION_SYSTEM_EEPROM_CHECK_BLUETOOTH_PIN_ADDRESS
00264 };
00265
00267 static const uint8_t location_system_array_lenghts[
LOCATION_SYSTEM_STRING_VARIABLES]=
00268 {
```

```
00269         LOCATION_SYSTEM_USER_NAME_LENGHT,
00270         LOCATION_SYSTEM_PIN_NUMBER_LENGHT,
00271         LOCATION_SYSTEM_USER_EMAIL_KEY_LENGHT,
00272         LOCATION_SYSTEM_EMAIL_SERVER_LENGHT,
00273         LOCATION_SYSTEM_EMAIL_PORT_LENGHT,
00274         LOCATION_SYSTEM_EMAIL_RMTE_LENGHT,
00275         LOCATION_SYSTEM_EMAILDST_NAME_LENGHT,
00276         LOCATION_SYSTEM_EMAILDST_LENGHT,
00277         LOCATION_SYSTEM_HOME_LAT_LENGHT,
00278         LOCATION_SYSTEM_HOME_LONG_LENGHT,
00279         LOCATION_SYSTEM_MOBILE_APN_NAME_LENGHT,
00280         LOCATION_SYSTEM_MOBILE_APN_LENGHT,
00281         LOCATION_SYSTEM_PUBLIC_IP_LENGHT,
00282         LOCATION_SYSTEM_PHONE_NUMBER_LENGHT,
00283         LOCATION_SYSTEM_BLUETOOTH_NAME_LENGHT,
00284         LOCATION_SYSTEM_BLUETOOTH_PIN_LENGHT
00285 };
00286
00288 static const uint16_t location_system_EEPROM_adress[
      LOCATION_SYSTEM_STRING_VARIABLES]=
00289 {
00290         LOCATION_SYSTEM_EEPROM_USER_NAME_ADDRESS,
00291         LOCATION_SYSTEM_EEPROM_PIN_NUMBER_ADDRESS,
00292         LOCATION_SYSTEM_EEPROM_USER_EMAIL_KEY_ADDRESS,
00293         LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_ADDRESS,
00294         LOCATION_SYSTEM_EEPROM_EMAIL_SERVER_PORT_ADDRESS,
00295         LOCATION_SYSTEM_EEPROM_EMAIL_RMTE_ADDRESS,
00296         LOCATION_SYSTEM_EEPROM_EMAIL_DEST_NAME_ADDRESS,
00297         LOCATION_SYSTEM_EEPROM_EMAIL_DST_ADDRESS,
00298         LOCATION_SYSTEM_EEPROM_HOME_LAT_ADDRESS,
00299         LOCATION_SYSTEM_EEPROM_HOME_LONG_ADDRESS,
00300         LOCATION_SYSTEM_EEPROM_MOBILE_APN_NAME_ADDRESS,
00301         LOCATION_SYSTEM_EEPROM_MOBILE_APN_ADDRESS,
00302         LOCATION_SYSTEM_EEPROM_PUBLIC_IP_ADDRESS,
00303         LOCATION_SYSTEM_EEPROM_PHONE_NUMBER_ADDRESS,
00304         LOCATION_SYSTEM_EEPROM_BLUETOOTH_NAME_ADDRESS,
00305         LOCATION_SYSTEM_EEPROM_BLUETOOTH_PIN_ADDRESS
00306 };
00307
00309 static char *location_system_strings_pointer[LOCATION_SYSTEM_STRING_VARIABLES
      ]=
00310 {
00311         location_system_user_name,
00312         location_system_pin_number,
00313         location_system_email_key,
00314         location_system_email_server,
00315         location_system_email_port,
00316         location_system_email_remitente,
00317         location_system_email_dest_name,
00318         location_system_email_destino,
00319         location_system_home_latitude,
00320         location_system_home_longitude,
00321         location_system_mobile_APN_name,
00322         location_system_mobile_APN,
00323         location_system_public_ip,
00324         location_system_phone_number,
00325         location_system_bluetooth_name,
00326         location_system_bluetooth_pin
```

```
00327 };
00328
00329 static char location_system_aux_str[200];
00330 static char location_system_at_gprs_buffer[250];
00331
00332 static long location_system_email_timeCounter;
00333 static long location_system_bluetooth_timeCounter;
00334
00336 const unsigned char UBX_HEADER[2] = { 0xB5, 0x62 };
00338 const unsigned char UBLOX_INIT[] PROGMEM =
00339 {
00340         // Deshabilita las tramas NMEA
00341         0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x24, // GxGGA off
00342         0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x01,0x00,0x00,0x00,0x00,0x00,0x01,0x01,0x2B, // GxGLL off
00343         0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x02,0x00,0x00,0x00,0x00,0x00,0x01,0x02,0x32, // GxGSA off
00344         0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x03,0x00,0x00,0x00,0x00,0x00,0x01,0x03,0x39, // GxGSV off
00345         0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x04,0x00,0x00,0x00,0x00,0x00,0x01,0x04,0x40, // GxRMC off
00346         0xB5,0x62,0x06,0x01,0x08,0x00,0xF0,0x05,0x00,0x00,0x00,0x00,0x00,0x01,0x05,0x47, // GxVTG off
00347
00348         // Deshabilita los mensajes NAV-POSLLH y NAV-STATUS con el protocolo UBLOX
00349         //  0xB5,0x62,0x06,0x01,0x08,0x00,0x01,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x17,0xDC, //NAV-PV
00350         0xB5,0x62,0x06,0x01,0x08,0x00,0x01,0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x12,0xB9, //NAV-POSLLH
00351         0xB5,0x62,0x06,0x01,0x08,0x00,0x01,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x13,0xC0, //NAV-STATUS
00352
00353         // Habilita el mensaje NAV-PVT con el protocolo UBLOX
00354         0xB5,0x62,0x06,0x01,0x08,0x00,0x01,0x07,0x00,0x01,0x00,0x00,0x00,0x00,0x18,0xE1, //NAV-PVT or
00355         //0xB5,0x62,0x06,0x01,0x08,0x00,0x01,0x02,0x00,0x01,0x00,0x00,0x00,0x00,0x13,0xBE, //NAV-POSI
00356         //0xB5,0x62,0x06,0x01,0x08,0x00,0x01,0x03,0x00,0x01,0x00,0x00,0x00,0x00,0x14,0xC5, //NAV-STAT
00357
00358         // Frecuencia de envio de datos del GPS
00359         //  0xB5,0x62,0x06,0x08,0x06,0x00,0x64,0x00,0x01,0x00,0x01,0x00,0x7A,0x12, //(10Hz)
00360         //  0xB5,0x62,0x06,0x08,0x06,0x00,0xC8,0x00,0x01,0x00,0x01,0x00,0xDE,0x6A, //(5Hz)
00361         0xB5,0x62,0x06,0x08,0x06,0x00,0xE8,0x03,0x01,0x00,0x01,0x00,0x01,0x39, //(1Hz)
00362
00363         // Baudrate
00364         0xB5,0x62,0x06,0x00,0x14,0x00,0x01,0x00,0x00,0x00,0xD0,0x08,0x00,0x00,0x00,0xE1,0x00,0x00,0x0
      ,0x03,0x00,0x00,0x00,0x00,0x00,0xDE,0xC9 //(57600)
00365 };
00366
00368 struct NAV_PVT
00369 {
00370     unsigned char cls;
00371     unsigned char id;
00372     unsigned short len;
00373     unsigned long iTOW;
00375     unsigned short year;
00376     unsigned char month;
00377     unsigned char day;
00378     unsigned char hour;
00379     unsigned char minute;
00380     unsigned char second;
00381     char valid;
00382     unsigned long tAcc;
00383     long nano;
00384     unsigned char fixType;
00385     char flags;
00386     unsigned char reserved1;
00387     unsigned char numSV;
```

```
00389     long lon;
00390     long lat;
00391     long height;
00392     long hMSL;
00393     unsigned long hAcc;
00394     unsigned long vAcc;
00396     long velN;
00397     long velE;
00398     long velD;
00399     long gSpeed;
00400     long heading;
00401     unsigned long sAcc;
00402     unsigned long headingAcc;
00403     unsigned short pDOP;
00404     short reserved2;
00405     unsigned long reserved3;
00406 };
00407
00408 NAV_PVT pvt;
00409
00414 static void location_system_calcChecksum(unsigned char* CK)
00415 {
00416     memset(CK, 0, 2);
00417     for (int i = 0; i < (int)sizeof(NAV_PVT); i++)
00418     {
00419         CK[0] += ((unsigned char*)(&pvt))[i];
00420         CK[1] += CK[0];
00421     }
00422 }
00423
00428 static bool location_system_processGPS(void)
00429 {
00430   static int fpos = 0;
00431   static unsigned char checksum[2];
00432   const int payloadSize = sizeof(NAV_PVT);
00433
00434   while (Serial2.available())
00435   {
00436     byte c = Serial2.read();
00437     if ( fpos < 2 )
00438     {
00439       if ( c == UBX_HEADER[fpos] )
00440         fpos++;
00441       else
00442         fpos = 0;
00443     }
00444     else
00445     {
00446       if ( (fpos-2) < payloadSize )
00447         ((unsigned char*)(&pvt))[fpos-2] = c;
00448
00449       fpos++;
00450
00451       if ( fpos == (payloadSize+2))
00452       {
00453         location_system_calcChecksum(checksum);
00454       }
00455       else if ( fpos == (payloadSize+3))
```

```
00456       {
00457         if (c != checksum[0])
00458           fpos = 0;
00459       }
00460       else if (fpos == (payloadSize+4))
00461       {
00462         fpos = 0;
00463         if ( c == checksum[1] )
00464         {
00465           return true;
00466         }
00467       }
00468       else if (fpos > (payloadSize+4))
00469       {
00470         fpos = 0;
00471       }
00472     }
00473   }
00474   return false;
00475 }
00476
00481 static void location_system_email_commands(void)
00482 {
00483     // Start by setting up the Email bearer profile identifier
00484     (void)location_system_gsm_gprs_sendATcommand((char*)"
      AT+EMAILCID=1",(char*) "OK", 2000);
00485     // Timeout del email.
00486     sprintf(location_system_at_gprs_buffer, "AT+EMAILTO=%d", LOCATION_SYSTEM_TO);
00487     (void)location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer, (char*) "OK", 2000);
00488
00489     //
00490     (void)location_system_gsm_gprs_sendATcommand((char*)"
      AT+EMAILSSL=1",(char*) "OK", 2000);
00491
00492     // Se selecciona el servidor de email y el puerto por el que se va a enviar.
00493     sprintf(location_system_at_gprs_buffer, "AT+SMTPSRV=\"%s\",%s", location_system_email_server,
      location_system_email_port);
00494     (void)location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer, (char*)"OK", 2000);
00495
00496     // Selecciona el email remitente y la clave de usuario (logging)
00497     sprintf(location_system_at_gprs_buffer, "AT+SMTPAUTH=1,\"%s\",\"%s\"", location_system_email_remi
       location_system_email_key);
00498     (void)location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer, (char*) "OK", 2000);
00499
00500     // Selecciona el email del remitente y su nombre
00501     sprintf(location_system_at_gprs_buffer, "AT+SMTPFROM=\"%s\",\"%s\"", location_system_email_remite
      location_system_user_name);
00502     (void)location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer, (char*) "OK", 2000);
00503
00504     // Selecciona la direccion del email destino y su nombre
00505     sprintf(location_system_at_gprs_buffer, "AT+SMTPRCPT=1,0,\"%s\",\"%s\"", location_system_email_de
       location_system_email_dest_name);
00506     (void)location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer,(char*) "OK", 2000);
```

```
00507
00508     // Cabecera del Email
00509     sprintf(location_system_at_gprs_buffer, "AT+SMTPSUB=\"%s\"", location_system_mail_subject);
00510     (void)location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer, (char*)"OK", 2000);
00511
00512     // Cuerpo del Email
00513     sprintf(location_system_at_gprs_buffer, "AT+SMTPBODY=%d",
      LOCATION_SYSTEM_EMAIL_LENGHT);
00514     if(location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer,(char*) "DOWNLOAD", 2000)==1)
00515     {
00516         // Espera el tiempo del timeout para que se escriba el mensaje
00517         Serial1.print("Hola, enviadme ayuda a esta direccion ");
00518         Serial1.print("http://maps.google.com/?q=");
00519         Serial1.print(pvt.lat / 10000000.0, 7);
00520         Serial1.print(",");
00521         Serial1.print(pvt.lon / 10000000.0, 7);
00522         Serial1.print(" \r\n");
00523         // Envia el E-mail y espera por la respuesta del modulo.
00524         if (location_system_gsm_gprs_sendATcommand((char*)"
      AT+SMTPSEND",(char*) "OK", 2000) == 1)
00525         {
00526             Serial.println("Email enviado!");
00527         }
00528         else
00529         {
00530             Serial.println("Error en el envio");
00531         }
00532     }
00533     else
00534     {
00535         Serial.println("Tiempo de espera agotado");
00536     }
00537 }
00538
00543 static void location_system_send_HTTP(void)
00544 {
00545     // Inicializa el servicio HTTP
00546     if (location_system_gsm_gprs_sendATcommand((char*)"AT+HTTPINIT",(
      char*) "OK", 2000) == 1)
00547     {
00548         // Activa el parametro CID
00549         if (location_system_gsm_gprs_sendATcommand((char*)"
      AT+HTTPPARA=\"CID\",1",(char*) "OK", 2000) == 1)
00550         {
00551             sprintf(location_system_aux_str, "AT+HTTPPARA=\"URL\",\"%s", location_system_url);
00552             Serial1.print(location_system_aux_str);
00553             Serial1.print("&field1=");
00554             Serial1.print(pvt.lat / 10000000.0, 7);
00555             Serial1.print("&field2=");
00556             Serial1.print(pvt.lon / 10000000.0, 7);
00557             Serial1.print("&field3=");
00558             Serial1.print(pvt.hMSL / 1000.0, 3);
00559             Serial1.print("&field4=");
00560             Serial1.print(location_system_datetime);
00561             Serial1.print("&field5=");
00562             Serial1.print(pvt.numSV);
```

```
00563             Serial1.print("&field6=");
00564             Serial1.print(pvt.gSpeed * 0.0036, 5);
00565             Serial1.print("&field7=");
00566             Serial1.print(pvt.heading / 100000.0, 5);
00567 
00568             if (location_system_gsm_gprs_sendATcommand((char*)"\"", (
      char*) "OK", 2000) == 1)
00569             {
00570                 // Realizamos la comprobacion de que hemos realizado la peticion.
00571                 if (location_system_gsm_gprs_sendATcommand((char*)"
      AT+HTTPACTION=0", (char*)"+HTTPACTION: 0,200,", 3000) == 1)
00572                 {
00573 
00574                     Serial.println(F("Peticion realizada"));
00575                 }
00576                 else
00577                 {
00578                     Serial.println(F("Error obteniendo url"));
00579                 }
00580             }
00581             else
00582             {
00583                 Serial.println(F("Error poniendo la url"));
00584             }
00585         }
00586         else
00587         {
00588             Serial.println(F("Error poniendo el CID"));
00589         }
00590     }
00591     else
00592     {
00593         Serial.println(F("Error inicializando"));
00594     }
00595     location_system_gsm_gprs_sendATcommand((char*)"AT+HTTPTERM", (
      char*)"OK", 2000);
00596 }
00597 
00602 static void location_system_EEPROM_clean(void)
00603 {
00604   for (int i = 0 ; i < 255 ; i++)
00605   {
00606     EEPROM.write(i, 0);
00607   }
00608 }
00609 
00614 static void location_system_EEPROM_read(void)
00615 {
00616     for (uint16_t i=0; i<LOCATION_SYSTEM_STRING_VARIABLES;i++)
00617     {
00618         location_system_EEPROM_check[i]=EEPROM.read(location_system_eeprom_check_bytes_address[i]);
00619 
00620         for (uint16_t j=0;j<location_system_array_lenghts[i];j++)
00621         {
00622             *(location_system_strings_pointer[i]+j)=EEPROM.read((location_system_EEPROM_adress[i]+j))
00623         }
00624     }
00625 }
```

```
00626
00632 static void location_system_received_items (void)
00633 {
00634     //
00635     char *location_system_pch;
00636     // Array de punteros de tipo char.
00637     char *location_system_array[1];
00638     uint16_t i=0;
00639     // Limpiamos el array de punteros con ceros.
00640     memset(location_system_array,'\0', sizeof(location_system_array));
00641
00642     location_system_pch = strtok (location_system_copy_response,":\r");
00643
00644     for (uint16_t x=0; x<LOCATION_SYSTEM_STRING_VARIABLES; x++)
00645     {
00646         // Compara todas las cabeceras (primer token)
00647         if (strcmp(location_system_pch, location_system_strings[x])==0)
00648         {
00649             while (location_system_pch != NULL)
00650             {
00651                 location_system_pch = strtok(NULL,":\r"); // Tokenizamos (troceamos) la cadena que tenem
      en el array TramaGPG por las comas
00652                 location_system_array[i++] = location_system_pch;
00653             }
00654
00655             strcpy(location_system_strings_pointer[x], location_system_array[0]);
00656             location_system_EEPROM_check[x]=LOCATION_SYSTEM_EEPROM_WRITTEN;
00657             EEPROM.write(location_system_eeprom_check_bytes_address[x],
     LOCATION_SYSTEM_EEPROM_WRITTEN);
00658
00659             for (uint16_t j=0; j<location_system_array_lenghts[x];j++)
00660             {
00661         EEPROM.write((location_system_EEPROM_adress[x]+j), 0);
00662                 EEPROM.write((location_system_EEPROM_adress[x]+j),*(location_system_strings_pointer[x]+j
00663             }
00664         }
00665     }
00666 }
00667
00672 void location_system_initialize(void)
00673 {
00674     Serial.begin(9600);
00675     Serial.println("Inicializando sistema...");
00676     // ############# Configuracion de E-S ################
00677     pinMode(LOCATION_SYSTEM_LED_BLUETOOTH, OUTPUT);
00678     pinMode(LOCATION_SYSTEM_LED_EMAIL, OUTPUT);
00679     pinMode(LOCATION_SYSTEM_LED_GPRS, OUTPUT);
00680     pinMode(LOCATION_SYSTEM_BLUETOOTH_ON_OFF, OUTPUT);
00681     pinMode(LOCATION_SYSTEM_GSM_GPRS_ON_OFF, OUTPUT);
00682     pinMode(LOCATION_SYSTEM_EMAIL_INTERRUPT, INPUT_PULLUP);
00683     pinMode(LOCATION_SYSTEM_BLUETOOTH_INTERRUPT, INPUT_PULLUP);
00684     attachInterrupt(digitalPinToInterrupt(LOCATION_SYSTEM_EMAIL_INTERRUPT),
00685             location_system_email_interrupt, FALLING);
00686     attachInterrupt(digitalPinToInterrupt(LOCATION_SYSTEM_BLUETOOTH_INTERRUPT
     ),
00687             location_system_bluetooth_interrupt, FALLING);
00688     // ################# Configuracion LCD ###############
00689     lcd.init();          // Inicializo el LCD.
```

```
00690     lcd.backlight();     // Brillo de la pantalla encendido.
00691     // ################## Lectura EEPROM ##################
00692     location_system_EEPROM_read();
00693     // ############### Apagar leds ######################
00694     LOCATION_SYSTEM_LED_MAIL_OFF();
00695     LOCATION_SYSTEM_LED_BLUETOOTH_OFF();
00696     LOCATION_SYSTEM_LED_GPRS_OFF();
00697     // ################ BLUETOOTH ######################
00698     location_system_Bluetooth_Reset();  // Resetea el modulo.
00699     LOCATION_SYSTEM_BLUETOOTH_OFF();   // Apaga el modulo.
00700     Serial.println("Bluetooth configurado!!");
00701     // ################## GPS ############################
00702     Serial2.begin(LOCATION_SYSTEM_GPS_OLD_BAUDRATE);
00703     // Enviamos los datos de configuracion utilizando el protocolo de Ublox.
00704     for(int i = 0; i < sizeof(UBLOX_INIT); i++)
00705     {
00706         // simulating a 38400baud pace (or less), otherwise commands are not accepted by the device.
00707         Serial2.write( pgm_read_byte(UBLOX_INIT+i));
00708         delay(5);
00709     }
00710     Serial2.begin(LOCATION_SYSTEM_GPS_NEW_BAUDRATE);
00711     Serial.println("Gps configurado!!");
00712     // ############### GPRS-GSM ##########################
00713     Serial1.begin(LOCATION_SYSTEM_GSM_GPRS_BAUDRATE);  // Inicializa el
      puerto serie.
00714     location_system_gsm_gprs_reset();
00715     // Inicializo el modulo GPRS con el pin y se conecta a la red GPRS
00716     location_system_gsm_gprs_initialize(location_system_pin_number,
00717             location_system_mobile_APN_name, location_system_mobile_APN);
00718     // ########## VARIABLES DEL SISTEMA ##################
00719     memset(location_system_response,'\0', sizeof(location_system_response));
00720     memset(location_system_copy_response,'\0', sizeof(location_system_copy_response));
00721     location_system_email_timeCounter=0;
00722     location_system_bluetooth_timeCounter=0;
00723     location_system_email_mode = EMAIL_OFF_MODE;
00724     location_system_bluetooth_mode =
     BLUETOOTH_OFF_MODE;
00725     location_system_bluetooth_state = 0;
00726     location_system_flag_timer = 0;
00727     location_system_email_button_timestamp = LOCATION_SYSTEM_EMAIL_TIMEOUT;
00728 }
00729
00735 void location_system_send_email(void)
00736 {
00737     static bool send_mail = false;
00738
00739     if (location_system_gsm_gprs_sendATcommand((char*)"AT+CPIN?", (
     char*)"+CPIN: READY", 2000) == 1)
00740     {
00741         //Verifica si ya se ha introducido el PIN de la tarjeta.
00742         Serial.println("PIN ya introducido");
00743
00744         // Verifica si la conexion esta configurada correctamente, si obtenemos una direccion IP, poc
      proceder.
00745         if (location_system_gsm_gprs_sendATcommand((char*)"
     AT+SAPBR=2,1",(char*)"+SAPBR: 1,1,", 2000) == 0)
00746         {
00747             Serial.println("Conectando a la red GPRS...");
```

```
00748             sprintf(location_system_at_gprs_buffer,"AT+SAPBR=3,1,\"%s\",\"%s\"",
      location_system_mobile_APN_name, location_system_mobile_APN);
00749             (void)location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer,(char*)"OK", 4000);
00750             // Se habilita el GPRS.
00751             (void)location_system_gsm_gprs_sendATcommand((char*)"
      AT+SAPBR=1,1",(char*)"OK", 2000);
00752         }
00753         send_mail = true;
00754     }
00755     else if (location_system_gsm_gprs_sendATcommand((char*)"AT+CPIN?"
      , (char*)"+CPIN: SIM PIN", 2000) == 1)
00756     {
00757         Serial.println("Reiniciando modulo SIM800L...");
00758         location_system_gsm_gprs_initialize(location_system_pin_number,
00759                 location_system_mobile_APN_name, location_system_mobile_APN);
00760         send_mail = true;
00761     }
00762     else
00763     {
00764         send_mail = false;
00765     }
00766
00767     if(send_mail == true)
00768     {
00769         LOCATION_SYSTEM_LED_MAIL_ON();
00770         location_system_email_commands();
00771         LOCATION_SYSTEM_LED_MAIL_OFF();
00772         send_mail = false;
00773     }
00774     else
00775     {
00776         //El modulo esta desconectado. No se envia el E-mail.
00777     }
00778 }
00779
00784 void location_system_buttons_pressed(void)
00785 {
00786     if((location_system_email_mode == EMAIL_ON_MODE) && (
      location_system_email_button_timestamp >= LOCATION_SYSTEM_EMAIL_TIMEOUT))
00787     {
00788         location_system_send_email();
00789         location_system_email_button_timestamp = 0;
00790         location_system_email_mode = EMAIL_OFF_MODE;
00791     }
00792
00793     if(location_system_bluetooth_mode ==
      BLUETOOTH_ON_MODE)
00794     {
00795         if(location_system_bluetooth_state == 255)
00796         {
00797             Serial3.begin(LOCATION_SYSTEM_HC05_SERIAL_CMD_BAUDRATE)
      ;
00798             LOCATION_SYSTEM_BLUETOOTH_ON();
00799             LOCATION_SYSTEM_LED_BLUETOOTH_ON();
00800         }
00801         else
00802         {
```

```
00803             Serial3.end();
00804             LOCATION_SYSTEM_BLUETOOTH_OFF();
00805             LOCATION_SYSTEM_LED_BLUETOOTH_OFF();
00806         }
00807         location_system_bluetooth_mode =
      BLUETOOTH_OFF_MODE;
00808     }
00809 }
00810
00815 void location_system_email_interrupt(void)
00816 {
00817     if (millis() > location_system_email_timeCounter +
      LOCATION_SYSTEM_EMAIL_BUTTON_TRESHOLD)
00818     {
00819         location_system_email_mode = EMAIL_ON_MODE;
00820         location_system_email_timeCounter = millis();
00821     }
00822 }
00823
00828 void location_system_bluetooth_interrupt(void)
00829 {
00830     if (millis() > location_system_bluetooth_timeCounter +
      LOCATION_SYSTEM_BLUETOOTTH_BUTTON_TRESHOLD)
00831     {
00832         location_system_bluetooth_state =~
       location_system_bluetooth_state;
00833         location_system_bluetooth_mode =
      BLUETOOTH_ON_MODE;
00834         location_system_bluetooth_timeCounter = millis();
00835
00836     }
00837 }
00838
00843 void location_system_bluetooth_receiver(void)
00844 {
00845     static byte bluetooth_byte=0;
00846     static uint8_t i = 0;
00847
00848     if(location_system_bluetooth_state == 255)
00849     {
00850         if (Serial3.available()>0)
00851         {
00852             bluetooth_byte = Serial3.read();
00853       location_system_response[i]=bluetooth_byte;    // y lo almacenaremos en la cadena de caracteres
       posicion que corresponda e incrementamos en uno el contador.
00854       i++;
00855             if (bluetooth_byte == '\r')
00856             {
00857                 strcpy(location_system_copy_response, location_system_response);
00858                 location_system_received_items();
00859                 memset(location_system_response,'\0', sizeof(location_system_response)); // Clean res
       buffer
00860                 Serial3.print(i); //Numero de caracteres recibidos por el bluettooth.
00861                 i=0;
00862             }
00863      else
00864      {
00865       //Caracteres recibidos no deseados.
```

```
00866      }
00867         }
00868     }
00869 }
00870
00875 void location_system_isr_timer(void)
00876 {
00877     location_system_flag_timer = true;
00878 }
00879
00884 void location_system_1_second_timing(void)
00885 {
00886     static uint8_t counter = 0;
00887
00888     if(location_system_flag_timer == true)
00889     {
00890         counter = (counter + 1) % LOCATION_SYSTEM_10_SECONDS_TIMING ;
00891
00892         location_system_email_button_timestamp >=LOCATION_SYSTEM_EMAIL_TIMEOUT
       ?
00893                 location_system_email_button_timestamp =
      LOCATION_SYSTEM_EMAIL_TIMEOUT:location_system_email_button_timestamp++;
00894
00895         if(counter == 0 && location_system_bluetooth_state == 0)
00896         {
00897             location_system_send_HTTP();
00898         }
00899
00900         lcd.setCursor(0,0);
00901         lcd.print("V(Km/h):");
00902         lcd.print(pvt.gSpeed * 0.0036, 5);
00903         lcd.setCursor(0,1);
00904         lcd.print("LON:");
00905         lcd.print(pvt.lon / 10000000.0, 7);
00906
00907         location_system_flag_timer = false;
00908     }
00909 }
00910
00915 void location_system_get_coordinates(void)
00916 {
00917     if (location_system_processGPS())
00918     {
00919         snprintf(location_system_datetime, LOCATION_SYSTEM_DATETIME_LENGTH,
      LOCATION_SYSTEM_DATETIME_FORMAT, pvt.year,
      pvt.month, pvt.day, pvt.hour, pvt.minute, pvt.
      second);
00920     }
00921 }
00922
```

### 11.4.9. Referencia del Archivo location_system.h

En este modulo se realizan todas las funciones que el programa principal necesita para un correcto funcionamiento del sistema.

```
#include <Arduino.h>
```

**defines**

- #define LOCATION_SYSTEM_BLUETOOTH_INTERRUPT (2U)
- #define LOCATION_SYSTEM_EMAIL_INTERRUPT (3U)
- #define LOCATION_SYSTEM_LED_BLUETOOTH (32U)
- #define LOCATION_SYSTEM_LED_EMAIL (36U)
- #define LOCATION_SYSTEM_LED_GPRS (40U)
- #define LOCATION_SYSTEM_LED_MAIL_ON() digitalWrite (LOCATION_SYSTEM_LED_EMAIL , HIGH)
- #define LOCATION_SYSTEM_LED_MAIL_OFF() digitalWrite (LOCATION_SYSTEM_LED_EMAIL , LOW)
- #define LOCATION_SYSTEM_LED_BLUETOOTH_ON() digitalWrite (LOCATION_SYSTEM_LED_BLUETOOT , HIGH)
- #define LOCATION_SYSTEM_LED_BLUETOOTH_OFF() digitalWrite (LOCATION_SYSTEM_LED_BLUETOO , LOW)
- #define LOCATION_SYSTEM_LED_GPRS_ON() digitalWrite (LOCATION_SYSTEM_LED_GPRS , HIGH)
- #define LOCATION_SYSTEM_LED_GPRS_OFF() digitalWrite (LOCATION_SYSTEM_LED_GPRS , LOW)
- #define LOCATION_SYSTEM_10_SECONDS_TIMING (20U)

**typedefs**

- typedef enum location_system_email_state_e location_system_email_state_t
- typedef enum location_system_bluetooth_state_e location_system_bluetooth_state_t

**Enumeraciones**

- enum location_system_email_state_e { EMAIL_ON_MODE, EMAIL_OFF_MODE }
- enum location_system_bluetooth_state_e { BLUETOOTH_ON_MODE, BLUETOOTH_OFF_MODE }

**Funciones**

- void location_system_email_interrupt (void)

  *Interrupcion del pulsador del E-mail.*
- void location_system_bluetooth_interrupt (void)

  *Interrupcion del pulsador del bluetooth.*
- void location_system_isr_timer (void)

*Cada vez que salta la interrupcion del timer se acude a esta funcion.*

- void location_system_bluetooth_receiver (void)

  *Recibe los caracteres por bluetooth.*

- void location_system_1_second_timing (void)

  *Esta funcion se encarga de llevar el tiempo.*

- void location_system_initialize (void)

  *Inicializa el sistema de localizacion.*

- void location_system_get_coordinates (void)

  *Obtiene las coordenadas GPS.*

- void location_system_buttons_pressed (void)

  *Esta funcion detecta si alguno de los botones ha sido presionado.*

#### 11.4.9.1. Descripcion detallada

En este modulo se realizan todas las funciones que el progama principal necesita para un correcto funcionamiento del sistema.

**Version**

1.0

**Fecha**

11/11/2017

**Autor**

Cristobal Garcia Camoira

Definicion en el archivo location_system.h.

#### 11.4.9.2. Documentacion de los 'defines'

#### 11.4.9.3. LOCATION_SYSTEM_10_SECONDS_TIMING

```
#define LOCATION_SYSTEM_10_SECONDS_TIMING (20U)
```

Definicion en la linea 28 del archivo location_system.h.

#### 11.4.9.4. LOCATION_SYSTEM_BLUETOOTH_INTERRUPT

```
#define LOCATION_SYSTEM_BLUETOOTH_INTERRUPT (2U)
```

Pines utilizados del arduino

Definicion en la linea 15 del archivo location_system.h.

#### 11.4.9.5. LOCATION_SYSTEM_EMAIL_INTERRUPT

```
#define LOCATION_SYSTEM_EMAIL_INTERRUPT (3U)
```

Definicion en la linea 16 del archivo location_system.h.

#### 11.4.9.6. LOCATION_SYSTEM_LED_BLUETOOTH

```
#define LOCATION_SYSTEM_LED_BLUETOOTH (32U)
```

Definicion en la linea 17 del archivo location_system.h.

#### 11.4.9.7. LOCATION_SYSTEM_LED_BLUETOOTH_OFF

```
#define LOCATION_SYSTEM_LED_BLUETOOTH_OFF( ) digitalWrite (LOCATION_SYSTEM_LED_BLUETOOTH
, LOW)
```

Definicion en la linea 24 del archivo location_system.h.

#### 11.4.9.8. LOCATION_SYSTEM_LED_BLUETOOTH_ON

```
#define LOCATION_SYSTEM_LED_BLUETOOTH_ON( ) digitalWrite (LOCATION_SYSTEM_LED_BLUETOOTH
, HIGH)
```

Definicion en la linea 23 del archivo location_system.h.

#### 11.4.9.9. LOCATION_SYSTEM_LED_EMAIL

```
#define LOCATION_SYSTEM_LED_EMAIL (36U)
```

Definicion en la linea 18 del archivo location_system.h.

#### 11.4.9.10. LOCATION_SYSTEM_LED_GPRS

```
#define LOCATION_SYSTEM_LED_GPRS (40U)
```

Definicion en la linea 19 del archivo location_system.h.

#### 11.4.9.11. LOCATION_SYSTEM_LED_GPRS_OFF

```
#define LOCATION_SYSTEM_LED_GPRS_OFF( ) digitalWrite (LOCATION_SYSTEM_LED_GPRS , L↩
OW)
```

Definicion en la linea 26 del archivo location_system.h.

#### 11.4.9.12. LOCATION_SYSTEM_LED_GPRS_ON

```
#define LOCATION_SYSTEM_LED_GPRS_ON( ) digitalWrite (LOCATION_SYSTEM_LED_GPRS , H↩
IGH)
```

Definicion en la linea 25 del archivo location_system.h.

#### 11.4.9.13. LOCATION_SYSTEM_LED_MAIL_OFF

```
#define LOCATION_SYSTEM_LED_MAIL_OFF( ) digitalWrite (LOCATION_SYSTEM_LED_EMAIL , L↩
OW)
```

Definicion en la linea 22 del archivo location_system.h.

#### 11.4.9.14. LOCATION_SYSTEM_LED_MAIL_ON

```
#define LOCATION_SYSTEM_LED_MAIL_ON( ) digitalWrite (LOCATION_SYSTEM_LED_EMAIL , H↩
IGH)
```

Definicion en la linea 21 del archivo location_system.h.

#### 11.4.9.15. Documentacion de los 'typedefs'

#### 11.4.9.16. location_system_bluetooth_state_t

```
typedef enum location_system_bluetooth_state_e location_system_bluetooth_state_t
```

#### 11.4.9.17. location_system_email_state_t

```
typedef enum location_system_email_state_e location_system_email_state_t
```

#### 11.4.9.18. Documentacion de las enumeraciones

#### 11.4.9.19. location_system_bluetooth_state_e

```
enum location_system_bluetooth_state_e
```

**Valores de enumeraciones**

| | |
|---|---|
| BLUETOOTH_ON_MODE | |
| BLUETOOTH_OFF_MODE | |

Definicion en la linea 36 del archivo location_system.h.

#### 11.4.9.20. location_system_email_state_e

```
enum location_system_email_state_e
```

**Valores de enumeraciones**

| | |
|---|---|
| EMAIL_ON_MODE | |
| EMAIL_OFF_MODE | |

Definicion en la linea 30 del archivo location_system.h.

#### 11.4.9.21. Documentacion de las funciones

#### 11.4.9.22. location_system_1_second_timing()

```
void location_system_1_second_timing (
        void  )
```

Esta funcion se encarga de llevar el tiempo.

Definicion en la linea 884 del archivo location_system.cpp.

#### 11.4.9.23. location_system_bluetooth_interrupt()

```
void location_system_bluetooth_interrupt (
        void  )
```

Interrupcion del pulsador del bluetooth.

Definicion en la linea 828 del archivo location_system.cpp.

#### 11.4.9.24. location_system_bluetooth_receiver()

```
void location_system_bluetooth_receiver (
        void  )
```

Recibe los caracteres por bluetooth.

Definicion en la linea 843 del archivo location_system.cpp.

#### 11.4.9.25. location_system_buttons_pressed()

```
void location_system_buttons_pressed (
        void  )
```

Esta funcion detecta si alguno de los botones ha sido presionado.

Definicion en la linea 784 del archivo location_system.cpp.

#### 11.4.9.26. location_system_email_interrupt()

```
void location_system_email_interrupt (
        void  )
```

Interrupcion del pulsador del E-mail.

Definicion en la linea 815 del archivo location_system.cpp.

#### 11.4.9.27. location_system_get_coordinates()

```
void location_system_get_coordinates (
        void  )
```

Obtiene las coordenadas GPS.

Definicion en la linea 915 del archivo location_system.cpp.

#### 11.4.9.28. location_system_initialize()

```
void location_system_initialize (
        void  )
```

Inicializa el sistema de localizacion.

Definicion en la linea 672 del archivo location_system.cpp.

#### 11.4.9.29. location_system_isr_timer()

```
void location_system_isr_timer (
        void  )
```

Cada vez que salta la interrupcion del timer se acude a esta funcion.

Definicion en la linea 875 del archivo location_system.cpp.

### 11.4.10. location_system.h

```
00001
00009 #ifndef LOCATION_SYSTEM_H_
00010 #define LOCATION_SYSTEM_H_
00011
00012 #include <Arduino.h>
00013
00015 #define LOCATION_SYSTEM_BLUETOOTH_INTERRUPT (2U)
00016 #define LOCATION_SYSTEM_EMAIL_INTERRUPT     (3U)
00017 #define LOCATION_SYSTEM_LED_BLUETOOTH       (32U)
00018 #define LOCATION_SYSTEM_LED_EMAIL           (36U)
00019 #define LOCATION_SYSTEM_LED_GPRS            (40U)
00020
```

```
00021 #define LOCATION_SYSTEM_LED_MAIL_ON()           digitalWrite (LOCATION_SYSTEM_LED_EMAIL , HIGH)
00022 #define LOCATION_SYSTEM_LED_MAIL_OFF()          digitalWrite (LOCATION_SYSTEM_LED_EMAIL , LOW)
00023 #define LOCATION_SYSTEM_LED_BLUETOOTH_ON()      digitalWrite (LOCATION_SYSTEM_LED_BLUETOOTH , HIGH)
00024 #define LOCATION_SYSTEM_LED_BLUETOOTH_OFF()     digitalWrite (LOCATION_SYSTEM_LED_BLUETOOTH , LOW)
00025 #define LOCATION_SYSTEM_LED_GPRS_ON()           digitalWrite (LOCATION_SYSTEM_LED_GPRS , HIGH)
00026 #define LOCATION_SYSTEM_LED_GPRS_OFF()          digitalWrite (LOCATION_SYSTEM_LED_GPRS , LOW)
00027
00028 #define LOCATION_SYSTEM_10_SECONDS_TIMING  (20U)  // 30 milisegundos de espera antes de renunciar a l
       bus I2C.
00029
00030 typedef enum location_system_email_state_e
00031 {
00032   EMAIL_ON_MODE,
00033   EMAIL_OFF_MODE
00034 }location_system_email_state_t;
00035
00036 typedef enum location_system_bluetooth_state_e
00037 {
00038   BLUETOOTH_ON_MODE,
00039   BLUETOOTH_OFF_MODE
00040 }location_system_bluetooth_state_t;
00041
00042 void location_system_email_interrupt(void);
00043 void location_system_bluetooth_interrupt(void);
00044 void location_system_isr_timer(void);
00045 void location_system_bluetooth_receiver(void);
00046 void location_system_1_second_timing(void);
00047 void location_system_initialize(void);
00048 void location_system_get_coordinates(void);
00049 void location_system_buttons_pressed(void);
00050
00051 #endif /* LOCATION_SYSTEM_H_ */
00052
```

### 11.4.11. Referencia del Archivo Location_system_arduino_megav2.cpp

```
#include <Arduino.h>
#include <LiquidCrystal_I2C.h>
#include "location_system.h"
#include "location_system_hc05.h"
#include "location_system_gsm_gprs.h"
#include <TimerOne.h>
#include <stdio.h>
#include <string.h>
#include <EEPROM.h>
```

#### Funciones

- LiquidCrystal_I2C lcd (0x27, 16, 2)
- void setup ()
- void loop ()

#### 11.4.11.1. Documentacion de las funciones

#### 11.4.11.2. lcd()

```
LiquidCrystal_I2C lcd (
        0x27 ,
        16 ,
        2  )
```

Establece la direccion de memoria 0x27 para un display de 16 caracteres y 2 lineas.

#### 11.4.11.3. loop()

```
void loop ( )
```

Definicion en la linea 23 del archivo Location_system_arduino_megav2.cpp.

#### 11.4.11.4. setup()

```
void setup ( )
```

Inicializa el timer1 con una cadencia de 1s.

attaches callback() as a timer overflow interrupt

Definicion en la linea 14 del archivo Location_system_arduino_megav2.cpp.

### 11.4.12. Location_system_arduino_megav2.cpp

```
00001 #include <Arduino.h>
00002 #include <LiquidCrystal_I2C.h>   // Para usar un display I2C
00003 #include "location_system.h"
00004 #include "location_system_hc05.h"
00005 #include "location_system_gsm_gprs.h"
00006 #include <TimerOne.h>
00007 #include <stdio.h>
00008 #include <string.h>
00009 #include <EEPROM.h>
00010
00012 LiquidCrystal_I2C lcd(0x27,16,2);
00013
00014 void setup()
00015 {
00016   location_system_initialize();
00018   Timer1.initialize(1000000);
00020   Timer1.attachInterrupt(location_system_isr_timer);
00021 }
00022
00023 void loop()
00024 {
00025
00026     location_system_get_coordinates();
00027
00028     location_system_buttons_pressed();
```

```
00029
00030     location_system_1_second_timing();
00031
00032     location_system_bluetooth_receiver();
00033 }
```

### 11.4.13. Referencia del Archivo location_system_gsm_gprs.cpp

En este modulo se implementan todas las funciones necesarias para el manejo del modem SIM800L.

```
#include <Arduino.h>
#include <HardwareSerial.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "location_system_gsm_gprs.h"
```

#### defines

- #define LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT (250U)
- #define LOCATION_SYSTEM_GSM_GPRS_BUFFER_LENGHT (250U)

#### Funciones

- void location_system_gsm_gprs_reset (void)

  *Resetea el modulo SIM800L.*
- int8_t location_system_gsm_gprs_sendATcommand (char ∗ATcommand, char ∗expected_answer, unsigned int timeout)

  *Funcion que envia el comando AT al modulo SIM800L y espera a recibir respuesta.*
- int8_t location_system_gsm_gprs_sendATcommand2 (char ∗ATcommand, char ∗expected_answer1, char ∗expected_answer2, unsigned int timeout)

  *Funcion que envia el comando AT al modulo SIM800L y espera a recibir 2 respuestas.*
- void location_system_gsm_gprs_initialize (char ∗pin_number, char ∗APN_name, char ∗APN_↩ server)

  *Inicia el modulo SIM800L introduciendole el PIN y conectandose a la red GPRS.*

#### 11.4.13.1. Descripcion detallada

En este modulo se implementan todas las funciones necesarias para el manejo del modem SIM800L.

#### Version

1.0

**Fecha**

11/11/2017

**Autor**

Cristobal Garcia Camoira

Definicion en el archivo location_system_gsm_gprs.cpp.

#### 11.4.13.2. Documentacion de los 'defines'

#### 11.4.13.3. LOCATION_SYSTEM_GSM_GPRS_BUFFER_LENGHT

```
#define LOCATION_SYSTEM_GSM_GPRS_BUFFER_LENGHT (250U)
```

Definicion en la linea 17 del archivo location_system_gsm_gprs.cpp.

#### 11.4.13.4. LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT

```
#define LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT (250U)
```

Definicion en la linea 16 del archivo location_system_gsm_gprs.cpp.

#### 11.4.13.5. Documentacion de las funciones

#### 11.4.13.6. location_system_gsm_gprs_initialize()

```
void location_system_gsm_gprs_initialize (
        char * pin_number,
        char * APN_name,
        char * APN_server )
```

Inicia el modulo SIM800L introduciendole el PIN y conectandose a la red GPRS.

Definicion en la linea 120 del archivo location_system_gsm_gprs.cpp.

#### 11.4.13.7. location_system_gsm_gprs_reset()

```
void location_system_gsm_gprs_reset (
        void  )
```

Resetea el modulo SIM800L.

Definicion en la linea 27 del archivo location_system_gsm_gprs.cpp.

#### 11.4.13.8. location_system_gsm_gprs_sendATcommand()

```
int8_t location_system_gsm_gprs_sendATcommand (
        char * ATcommand,
        char * expected_answer,
        unsigned int timeout )
```

Funcion que envia el comando AT al modulo SIM800L y espera a recibir respuesta.

Definicion en la linea 39 del archivo location_system_gsm_gprs.cpp.

#### 11.4.13.9. location_system_gsm_gprs_sendATcommand2()

```
int8_t location_system_gsm_gprs_sendATcommand2 (
        char * ATcommand,
        char * expected_answer1,
        char * expected_answer2,
        unsigned int timeout )
```

Funcion que envia el comando AT al modulo SIM800L y espera a recibir 2 respuestas.

Definicion en la linea 77 del archivo location_system_gsm_gprs.cpp.

### 11.4.14. location_system_gsm_gprs.cpp

```
00001
00009 #include <Arduino.h>
00010 #include <HardwareSerial.h>
00011 #include <stdio.h>
00012 #include <string.h>
00013 #include <stdlib.h>
00014 #include "location_system_gsm_gprs.h"
00015
00016 #define LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT  (250U)
00017 #define LOCATION_SYSTEM_GSM_GPRS_BUFFER_LENGHT    (250U)
00018
00019 static char location_system_gsm_gprs_response1[
      LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT];
00020 static char location_system_gsm_gprs_response2[
      LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT];
00021 static char location_system_at_gprs_buffer[
      LOCATION_SYSTEM_GSM_GPRS_BUFFER_LENGHT];
00022
00027 void location_system_gsm_gprs_reset(void)
00028 {
00029     LOCATION_SYSTEM_GSM_GPRS_OFF();
00030     delay(1000);
00031     LOCATION_SYSTEM_GSM_GPRS_ON();
00032     delay(4000);
00033 }
00034
00039 int8_t location_system_gsm_gprs_sendATcommand(char* ATcommand, char*
      expected_answer, unsigned int timeout)
00040 {
00041     uint8_t x=0,  answer=0;
```

```
00042     unsigned long previous;
00043     // Se inicializa el string
00044     memset(location_system_gsm_gprs_response1, '\0',
      LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT);
00045     // Retardo para asegurar que no exixtan interferencias con otros comandos
00046     delay(100);
00047     // espera a que se limpie el buffer de entrada
00048     while( Serial1.available() > 0) Serial1.read();
00049     Serial1.println(ATcommand);    // Se envia el comando AT.
00050     Serial.println(ATcommand);     // Se envia el comando AT.
00051     x=0;
00052     previous = millis();
00053
00054     // Este bucle espera por la respuesta
00055     do{
00056         // Si existen datos en el buffer de entrada, se leen y se comprueba la respuesta del modulo
00057         if(Serial1.available()>0)
00058         {
00059             location_system_gsm_gprs_response1[x] = Serial1.read();
00060             x++;
00061             // revisa si la respuesta deseada es la respuesta que hemos obtenido del modulo.
00062             if (strstr(location_system_gsm_gprs_response1, expected_answer) != NULL)
00063             {
00064                 answer = 1;
00065             }
00066         }
00067         // Se realiza una espera por la respuesta con el timeout
00068     } while((answer == 0) && ((millis() - previous) < timeout));
00069
00070     return answer;
00071 }
00072
00077 int8_t location_system_gsm_gprs_sendATcommand2(char* ATcommand, char
      * expected_answer1,char* expected_answer2, unsigned int timeout)
00078 {
00079     uint8_t x=0,  answer=0;
00080     unsigned long previous;
00081     // Se inicializa el string.
00082     memset(location_system_gsm_gprs_response2, '\0',
      LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT);
00083     // Retardo para asegurar que no exixtan interferencias con otros comandos
00084     delay(100);
00085     // Espera a que se borre el buffer de entrada.
00086     while( Serial1.available() > 0) Serial1.read();
00087     // Se envia el comando AT
00088     Serial1.println(ATcommand);
00089     x=0;
00090     previous = millis();
00091
00092     // Este bucle espera por la respuesta
00093     do{
00094         // Si existen datos en el buffer de entrada, se leen y se comprueba la respuesta del modulo.
00095         if(Serial1.available()>0)
00096         {
00097             location_system_gsm_gprs_response2[x] = Serial1.read();
00098             x++;
00099             // revisa si la respuesta deseada 1 es la respuesta que hemos obtenido del modulo.
00100             if (strstr(location_system_gsm_gprs_response2, expected_answer1) != NULL)
```

```
00101             {
00102                 answer = 1;
00103             }
00104             // revisa si la respuesta deseada 2 es la respuesta que hemos obtenido del modulo.
00105             else if (strstr(location_system_gsm_gprs_response2, expected_answer2) != NULL)
00106             {
00107                 answer = 2;
00108             }
00109         }
00110         //Se realiza una espera por la respuesta con el timeout
00111     } while((answer == 0) && ((millis() - previous) < timeout));
00112 
00113     return answer;
00114 }
00115 
00120 void location_system_gsm_gprs_initialize(char* pin_number, char*
      APN_name, char* APN_server)
00121 {
00122     // Verifica si el modulo responde.
00123     if(location_system_gsm_gprs_sendATcommand((char*)"AT",(char*)"OK"
      ,2000) == 1)
00124     {
00125         // Verifica si hay SIM en el modulo.
00126         if(location_system_gsm_gprs_sendATcommand((char*)"AT+CPIN?",(
      char*)"ERROR", 2000) == 1)
00127         {
00128             Serial.println("PIN incorrecto");
00129         }
00130         else
00131         {
00132             if(location_system_gsm_gprs_sendATcommand((char*)"
      AT+CPIN?", (char*)"+CPIN: SIM PIN", 2000) == 1)
00133             {
00134                 sprintf(location_system_at_gprs_buffer,"AT+CPIN=%s", pin_number);
00135                 (void)location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer,(char*)"OK", 2000);
00136                 delay(4000);
00137                 if(location_system_gsm_gprs_sendATcommand((char*)"
      AT+CREG?",(char*)"OK", 2000) == 1)
00138                 {
00139                     delay(2000);
00140                     // Se introduce el APN y su nombre
00141                     sprintf(location_system_at_gprs_buffer,"AT+SAPBR=3,1,\"%s\",\"%s\"", APN_name,
      APN_server);
00142                     (void)location_system_gsm_gprs_sendATcommand(
      location_system_at_gprs_buffer,(char*)"OK", 2000);
00143                     delay(2000);
00144                     // Se habilita el GPRS.
00145                     (void)location_system_gsm_gprs_sendATcommand((
      char*)"AT+SAPBR=1,1",(char*)"OK", 2000);
00146                     Serial.println("GPRS configurado!");
00147                 }
00148             }
00149             else
00150             {
00151                 Serial.println("Numero PIN ya introducido");
00152             }
00153         }
```

```
00154     }
00155     else
00156     {
00157         Serial.println("Modulo GPRS desconectado");
00158     }
00159 }
00160
```

### 11.4.15. Referencia del Archivo location_system_gsm_gprs.h

Este modulo contiene las cabeceras de las funciones y las variables para el control del modem SIM800L desde otros modulos.

```
#include <Arduino.h>
```

#### defines

- #define LOCATION_SYSTEM_GSM_GPRS_ON_OFF (7U)
- #define LOCATION_SYSTEM_GSM_GPRS_BAUDRATE (9600U)
- #define LOCATION_SYSTEM_GSM_GPRS_ON() digitalWrite (LOCATION_SYSTEM_GSM_GPRS_ON_OFF, HIGH)
- #define LOCATION_SYSTEM_GSM_GPRS_OFF() digitalWrite (LOCATION_SYSTEM_GSM_GPRS_ON_OFF, LOW)

#### Funciones

- int8_t location_system_gsm_gprs_sendATcommand (char ∗ATcommand, char ∗expected_answer, unsigned int timeout)

  *Funcion que envia el comando AT al modulo SIM800L y espera a recibir respuesta.*

- int8_t location_system_gsm_gprs_sendATcommand2 (char ∗ATcommand, char ∗expected_answer1, char ∗expected_answer2, unsigned int timeout)

  *Funcion que envia el comando AT al modulo SIM800L y espera a recibir 2 respuestas.*

- void location_system_gsm_gprs_initialize (char ∗pin_number, char ∗APN_name, char ∗APN_server)

  *Inicia el modulo SIM800L introduciendole el PIN y conectandose a la red GPRS.*

- void location_system_gsm_gprs_reset (void)

  *Resetea el modulo SIM800L.*

#### 11.4.15.1. Descripcion detallada

Este modulo contiene las cabeceras de las funciones y las variables para el control del modem SIM800L desde otros modulos.

Este modulo contiene las cabeceras de las funciones y las variables para el control del bluetooth HC-05 desde otros modulos.

**Version**

1.0

**Fecha**

11/11/2017

**Autor**

Cristobal Garcia Camoira

Definicion en el archivo location_system_gsm_gprs.h.

#### 11.4.15.2. Documentacion de los 'defines'

#### 11.4.15.3. LOCATION_SYSTEM_GSM_GPRS_BAUDRATE

```
#define LOCATION_SYSTEM_GSM_GPRS_BAUDRATE (9600U)
```

Definicion en la linea 15 del archivo location_system_gsm_gprs.h.

#### 11.4.15.4. LOCATION_SYSTEM_GSM_GPRS_OFF

```
#define LOCATION_SYSTEM_GSM_GPRS_OFF( ) digitalWrite (LOCATION_SYSTEM_GSM_GPRS_ON_OFF,
LOW)
```

Definicion en la linea 18 del archivo location_system_gsm_gprs.h.

#### 11.4.15.5. LOCATION_SYSTEM_GSM_GPRS_ON

```
#define LOCATION_SYSTEM_GSM_GPRS_ON( ) digitalWrite (LOCATION_SYSTEM_GSM_GPRS_ON_OFF,
HIGH)
```

Definicion en la linea 17 del archivo location_system_gsm_gprs.h.

#### 11.4.15.6. LOCATION_SYSTEM_GSM_GPRS_ON_OFF

```
#define LOCATION_SYSTEM_GSM_GPRS_ON_OFF (7U)
```

Definicion en la linea 14 del archivo location_system_gsm_gprs.h.

#### 11.4.15.7. Documentacion de las funciones

#### 11.4.15.8. location_system_gsm_gprs_initialize()

```
void location_system_gsm_gprs_initialize (
        char * pin_number,
        char * APN_name,
        char * APN_server )
```

Inicia el modulo SIM800L introduciendole el PIN y conectandose a la red GPRS.

Definicion en la linea 120 del archivo location_system_gsm_gprs.cpp.

#### 11.4.15.9. location_system_gsm_gprs_reset()

```
void location_system_gsm_gprs_reset (
        void  )
```

Resetea el modulo SIM800L.

Definicion en la linea 27 del archivo location_system_gsm_gprs.cpp.

#### 11.4.15.10. location_system_gsm_gprs_sendATcommand()

```
int8_t location_system_gsm_gprs_sendATcommand (
        char * ATcommand,
        char * expected_answer,
        unsigned int timeout )
```

Funcion que envia el comando AT al modulo SIM800L y espera a recibir respuesta.

Definicion en la linea 39 del archivo location_system_gsm_gprs.cpp.

#### 11.4.15.11. location_system_gsm_gprs_sendATcommand2()

```
int8_t location_system_gsm_gprs_sendATcommand2 (
        char * ATcommand,
        char * expected_answer1,
        char * expected_answer2,
        unsigned int timeout )
```

Funcion que envia el comando AT al modulo SIM800L y espera a recibir 2 respuestas.

Definicion en la linea 77 del archivo location_system_gsm_gprs.cpp.

### 11.4.16. location_system_gsm_gprs.h

```
00001
00009 #ifndef LOCATION_SYSTEM_GSM_GPRS_H_
00010 #define LOCATION_SYSTEM_GSM_GPRS_H_
00011
00012 #include <Arduino.h>
00013
```

```
00014 #define LOCATION_SYSTEM_GSM_GPRS_ON_OFF   (7U)
00015 #define LOCATION_SYSTEM_GSM_GPRS_BAUDRATE (9600U)
00016
00017 #define LOCATION_SYSTEM_GSM_GPRS_ON()   digitalWrite (LOCATION_SYSTEM_GSM_GPRS_ON_OFF, HIGH)
00018 #define LOCATION_SYSTEM_GSM_GPRS_OFF()  digitalWrite (LOCATION_SYSTEM_GSM_GPRS_ON_OFF, LOW)
00019
00020 int8_t location_system_gsm_gprs_sendATcommand(char* ATcommand, char*
      expected_answer, unsigned int timeout);
00021 int8_t location_system_gsm_gprs_sendATcommand2(char* ATcommand, char
      * expected_answer1, char* expected_answer2, unsigned int timeout);
00022 void location_system_gsm_gprs_initialize(char* pin_number, char*
      APN_name, char* APN_server);
00023 void location_system_gsm_gprs_reset(void);
00024
00025 #endif /* LOCATION_SYSTEM_GSM_GPRS_H_ */
00026
```

### 11.4.17. Referencia del Archivo location_system_hc05.cpp

En este modulo se implementan todas las funciones necesarias para el manejo del bluetooth.

```
#include <Arduino.h>
#include <HardwareSerial.h>
#include <stdio.h>
#include <string.h>
#include "location_system_hc05.h"
```

#### defines

- #define LOCATION_SYSTEM_HC05_RESPONSE (100U)

#### Funciones

- int8_t location_system_hc05_send_at_cmd (char ∗ATcommand, char ∗expected_answer, unsigned int timeout)

  *Esta funcion se utiliza para enviar comandos para la configuracion del bluetooth.*

- void location_system_Bluetooth_At (void)

  *Esta funcion se utiliza para poner en modo de recepcion de comandos bluetooth.*

- void location_system_Bluetooth_Reset (void)

  *Esta funcion se utiliza para resetear el bluetooth.*

#### 11.4.17.1. Descripcion detallada

En este modulo se implementan todas las funciones necesarias para el manejo del bluetooth.

#### Version

1.0

**Fecha**

11/02/2018

**Autor**

Cristobal Garcia Camoira

Definicion en el archivo location_system_hc05.cpp.

#### 11.4.17.2. Documentacion de los 'defines'

#### 11.4.17.3. LOCATION_SYSTEM_HC05_RESPONSE

```
#define LOCATION_SYSTEM_HC05_RESPONSE (100U)
```

Definicion en la linea 15 del archivo location_system_hc05.cpp.

#### 11.4.17.4. Documentacion de las funciones

#### 11.4.17.5. location_system_Bluetooth_At()

```
void location_system_Bluetooth_At (
        void  )
```

Esta funcion se utiliza para poner en modo de recepcion de comandos bluetooth.

Definicion en la linea 61 del archivo location_system_hc05.cpp.

#### 11.4.17.6. location_system_Bluetooth_Reset()

```
void location_system_Bluetooth_Reset (
        void  )
```

Esta funcion se utiliza para resetear el bluetooth.

Definicion en la linea 75 del archivo location_system_hc05.cpp.

#### 11.4.17.7. location_system_hc05_send_at_cmd()

```
int8_t location_system_hc05_send_at_cmd (
        char * ATcommand,
        char * expected_answer,
        unsigned int timeout )
```

Esta funcion se utiliza para enviar comandos para la configuracion del bluetooth.

Definicion en la linea 21 del archivo location_system_hc05.cpp.

### 11.4.18. location_system_hc05.cpp

```
00001
00009 #include <Arduino.h>
00010 #include <HardwareSerial.h>
00011 #include <stdio.h>
00012 #include <string.h>
00013 #include "location_system_hc05.h"
00014
00015 #define LOCATION_SYSTEM_HC05_RESPONSE  (100U)
00016
00021 int8_t location_system_hc05_send_at_cmd(char* ATcommand, char*
      expected_answer, unsigned int timeout)
00022 {
00023     uint8_t x=0,  answer=0;
00024     char response[LOCATION_SYSTEM_HC05_RESPONSE];
00025     unsigned long previous;
00026     // Se inicializa el string
00027     memset(response, '\0', LOCATION_SYSTEM_HC05_RESPONSE);
00028     // Retardo para asegurar que no exixtan interferencias con otros comandos
00029     delay(100);
00030     // Espera a que se borre el buffer de entrada.
00031     while( Serial3.available() > 0) Serial3.read();
00032     // Se envia el comando AT
00033     Serial3.println(ATcommand);
00034
00035     x = 0;
00036     previous = millis();
00037
00038     // Espera por la respuesta
00039     do{
00040         if(Serial3.available() != 0)
00041         {
00042             // Si existen datos en el buffer de entrada, se leen y se comprueba la respuesta del modu
00043             response[x] = Serial3.read();
00044             x++;
00045             // Revisa si la respuesta deseada es la respuesta que hemos obtenido del modulo.
00046             if (strstr(response, expected_answer) != NULL)
00047             {
00048                 answer = 1;
00049             }
00050         }
00051     }
00052     // Se realiza una espera por la respuesta con el timeout.
00053     while((answer == 0) && ((millis() - previous) < timeout));
00054     return answer;
00055 }
00056
00061 void location_system_Bluetooth_At(void)
00062 {
00063     Serial3.begin(LOCATION_SYSTEM_HC05_AT_CMD_BAUDRATE);
00064     LOCATION_SYSTEM_BLUETOOTH_OFF();           // Apaga el modulo
00065     delay (500) ;
00066     LOCATION_SYSTEM_BLUETOOTH_ATCMD_ENABLE();     // Pin activo para
       entrar en modo comandos.
00067     delay (500) ;                              // Espera antes de encender el modulo
00068     LOCATION_SYSTEM_BLUETOOTH_ON();            // Enciende el modulo
00069 }
00070
```

```
00075 void location_system_Bluetooth_Reset(void)
00076 {
00077     Serial3.begin(LOCATION_SYSTEM_HC05_SERIAL_CMD_BAUDRATE);
00078     LOCATION_SYSTEM_BLUETOOTH_OFF();           // Apaga el modulo
00079     delay (500) ;                               // Espera antes de encender el modulo.
00080     LOCATION_SYSTEM_BLUETOOTH_ATCMD_DISABLE(); // Pin desactivado
      para entrar en modo comandos.
00081     LOCATION_SYSTEM_BLUETOOTH_ON();            //Enciende el modulo
00082
00083 }
00084
```

### 11.4.19. Referencia del Archivo location_system_hc05.h

```
#include <Arduino.h>
```

#### defines

- #define LOCATION_SYSTEM_BLUETOOTH_ON_OFF (5U)
- #define LOCATION_SYSTEM_BLUETOOTH_AT_CMD (6U)
- #define LOCATION_SYSTEM_BLUETOOTH_ON() digitalWrite (LOCATION_SYSTEM_BLUETOOTH_ON_OFF, HIGH)
- #define LOCATION_SYSTEM_BLUETOOTH_OFF() digitalWrite (LOCATION_SYSTEM_BLUETOOTH_ON_OFF, LOW)
- #define LOCATION_SYSTEM_BLUETOOTH_ATCMD_ENABLE() digitalWrite (LOCATION_SYSTEM_BLUETOOT HIGH)
- #define LOCATION_SYSTEM_BLUETOOTH_ATCMD_DISABLE() digitalWrite (LOCATION_SYSTEM_BLUETOOT LOW)
- #define LOCATION_SYSTEM_HC05_AT_CMD_BAUDRATE (38400)
- #define LOCATION_SYSTEM_HC05_SERIAL_CMD_BAUDRATE (9600)

#### Funciones

- int8_t location_system_hc05_send_at_cmd (char ∗ATcommand, char ∗expected_answer, unsigned int timeout)

  *Esta funcion se utiliza para enviar comandos para la configuracion del bluetooth.*

- void location_system_Bluetooth_At (void)

  *Esta funcion se utiliza para poner en modo de recepcion de comandos bluetooth.*

- void location_system_Bluetooth_Reset (void)

  *Esta funcion se utiliza para resetear el bluetooth.*

#### 11.4.19.1. Documentacion de los 'defines'

#### 11.4.19.2. LOCATION_SYSTEM_BLUETOOTH_AT_CMD

```
#define LOCATION_SYSTEM_BLUETOOTH_AT_CMD (6U)
```

Definicion en la linea 14 del archivo location_system_hc05.h.

#### 11.4.19.3. LOCATION_SYSTEM_BLUETOOTH_ATCMD_DISABLE

```
#define LOCATION_SYSTEM_BLUETOOTH_ATCMD_DISABLE( ) digitalWrite (LOCATION_SYSTEM_BLUETOOTH_A
LOW)
```

Definicion en la linea 20 del archivo location_system_hc05.h.

#### 11.4.19.4. LOCATION_SYSTEM_BLUETOOTH_ATCMD_ENABLE

```
#define LOCATION_SYSTEM_BLUETOOTH_ATCMD_ENABLE( ) digitalWrite (LOCATION_SYSTEM_BLUETOOTH_AT
HIGH)
```

Definicion en la linea 19 del archivo location_system_hc05.h.

#### 11.4.19.5. LOCATION_SYSTEM_BLUETOOTH_OFF

```
#define LOCATION_SYSTEM_BLUETOOTH_OFF( ) digitalWrite (LOCATION_SYSTEM_BLUETOOTH_ON_OFF,
LOW)
```

Definicion en la linea 17 del archivo location_system_hc05.h.

#### 11.4.19.6. LOCATION_SYSTEM_BLUETOOTH_ON

```
#define LOCATION_SYSTEM_BLUETOOTH_ON( ) digitalWrite (LOCATION_SYSTEM_BLUETOOTH_ON_OFF,
HIGH)
```

Definicion en la linea 16 del archivo location_system_hc05.h.

#### 11.4.19.7. LOCATION_SYSTEM_BLUETOOTH_ON_OFF

```
#define LOCATION_SYSTEM_BLUETOOTH_ON_OFF (5U)
```

Definicion en la linea 13 del archivo location_system_hc05.h.

#### 11.4.19.8. LOCATION_SYSTEM_HC05_AT_CMD_BAUDRATE

```
#define LOCATION_SYSTEM_HC05_AT_CMD_BAUDRATE (38400)
```

Definicion en la linea 22 del archivo location_system_hc05.h.

#### 11.4.19.9. LOCATION_SYSTEM_HC05_SERIAL_CMD_BAUDRATE

```
#define LOCATION_SYSTEM_HC05_SERIAL_CMD_BAUDRATE (9600)
```

Definicion en la linea 23 del archivo location_system_hc05.h.

#### 11.4.19.10. Documentacion de las funciones

#### 11.4.19.11. location_system_Bluetooth_At()

```
void location_system_Bluetooth_At (
        void  )
```

Esta funcion se utiliza para poner en modo de recepcion de comandos bluetooth.

Definicion en la linea 61 del archivo location_system_hc05.cpp.

#### 11.4.19.12. location_system_Bluetooth_Reset()

```
void location_system_Bluetooth_Reset (
        void  )
```

Esta funcion se utiliza para resetear el bluetooth.

Definicion en la linea 75 del archivo location_system_hc05.cpp.

#### 11.4.19.13. location_system_hc05_send_at_cmd()

```
int8_t location_system_hc05_send_at_cmd (
        char * ATcommand,
        char * expected_answer,
        unsigned int timeout )
```

Esta funcion se utiliza para enviar comandos para la configuracion del bluetooth.

Definicion en la linea 21 del archivo location_system_hc05.cpp.

### 11.4.20. location_system_hc05.h

```
00001
00008 #ifndef LOCATION_SYSTEM_HC05_H_
00009 #define LOCATION_SYSTEM_HC05_H_
00010
00011 #include <Arduino.h>
00012
00013 #define LOCATION_SYSTEM_BLUETOOTH_ON_OFF  (5U)
00014 #define LOCATION_SYSTEM_BLUETOOTH_AT_CMD  (6U)
00015
00016 #define LOCATION_SYSTEM_BLUETOOTH_ON()   digitalWrite (LOCATION_SYSTEM_BLUETOOTH_ON_OFF, HIGH)
00017 #define LOCATION_SYSTEM_BLUETOOTH_OFF()  digitalWrite (LOCATION_SYSTEM_BLUETOOTH_ON_OFF, LOW)
00018
00019 #define LOCATION_SYSTEM_BLUETOOTH_ATCMD_ENABLE()      digitalWrite (LOCATION_SYSTEM_BLUETOOTH_AT_CMD, HI
```

```
00020 #define LOCATION_SYSTEM_BLUETOOTH_ATCMD_DISABLE()     digitalWrite (LOCATION_SYSTEM_BLUETOOTH_AT_CMD,
00021
00022 #define LOCATION_SYSTEM_HC05_AT_CMD_BAUDRATE    (38400)
00023 #define LOCATION_SYSTEM_HC05_SERIAL_CMD_BAUDRATE  (9600)
00024
00025 int8_t location_system_hc05_send_at_cmd(char* ATcommand, char*
      expected_answer, unsigned int timeout);
00026 void location_system_Bluetooth_At(void);
00027 void location_system_Bluetooth_Reset(void);
00028
00029 #endif /* LOCATION_SYSTEM_HC05_H_ */
00030
```

## 11.5. Lineas futuras del presente proyecto

En esta sección se indican aquellos puntos que, en un futuro se podrían implementar en este proyecto, ya sea para mejorar la funcionalidad del mismo como también la realización de otros proyectos y utilidades que nos puede ofrecer el módulo GPRS/GSM utilizado.

1. Reducción del tamaño del dispositivo optimizando la disposición de los componentes electrónicos en una o varias placas de circuito impreso.

2. Utilización de baterías de un tamaño mas reducido.

3. Evitar que durante el envío de un correo electrónico se paralice el envío del seguimiento al servidor web.

4. Realización de una aplicación para smarphone (android o ios) que trate el tema de la configuración del dispositivo.

5. Posibilidad de control total del dispositivo desde la aplicación realizada, tiempo de envío al servidor, posibilidad de realización de llamadas...

6. El módulo utilizado en el presente proyecto (SIM800L) no solo nos ofrece la posibilidad de ser utilizado para la localización de objetos puesto que prácticamente se puede utilizar como un teléfono móvil, pues el módulo, además de ser capaz de conectarse a la red GPRS y enviar E-mails, es capaz de realizar y recibir llamadas.

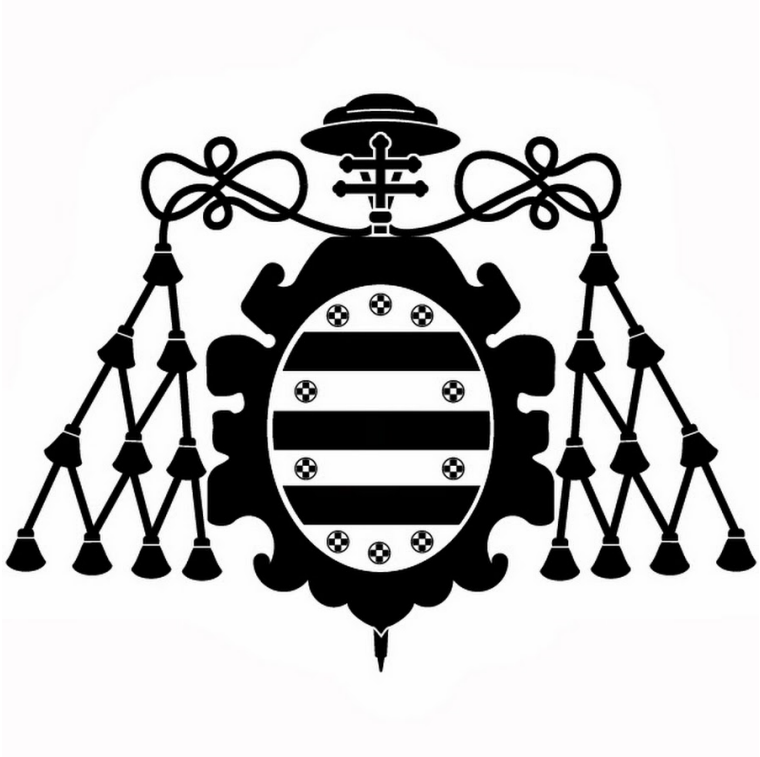
La arquitectura de este desarrollo es muy versátil, hasta tal punto que si se desea realizar cualquiera de los proyectos que se mencionan a continuación solo se debería que adaptar el software, dado que el hardware seguiría siendo el mismo.

- Sistema de seguridad de viviendas basado en Arduino y el módulo SIM800L donde el módulo seria capaz de enviarnos un SMS indicando si se ha abierto alguna puerta de nuestro domicilio, realizar una llamada en modo escucha para saber lo que puede estar ocurriendo en el interior y dar orden de sacar una foto o grabar al interior.

- Sistema de control de ganado basado en Arduino, este proyecto se realizaría de forma similar al presente, lo único que cambiaría seria que el diseño del circuito fuese mas pequeño de forma que pueda adaptarse a la ergonomía de cualquier animal de gran envergadura y ser transportado con facilidad.

- Teléfono móvil basado en Arduino.Este proyecto se aprovecharía al máximo todos los recursos que posee el módulo.En este proyecto, se incorporaría un teclado para poder marcar el numero a llamar y también para poder enviar un mensaje, un altavoz para poder escuchar las llamadas recibidas y un micrófono para poder comunicarnos.

- Servicio de telegestión de contadores. El módulo interaccionará con algún servidor enviando los datos referentes al consumo de nuestro domicilio para poder ser consultados en la red, de este modo se evitarían los inconvenientes relacionados con la estimación de las facturas.



# MANUAL DE USUARIO Y ESPECFICACIONES

**Índice del documento Manual de usuario y especificaciones**



# Índice de figuras



# Índice de tablas

# 1 Manual de usuario

En este capítulo se desarrolla el manual de usuario para la correcta utilización del dispositivo.

En primer lugar se indican las instrucciones de uso paso a paso para poder utilizar el localizador GPS y por otra parte se detallan las especificaciones técnicas del producto.

> **NOTA:** Es necesario disponer de una cuenta de Google para poder visualizar los datos del dispositivo, dado que la pagina web que se ha desarrollado para este proyecto utiliza sus servicios.

Antes de comenzar con el manual, comentar que el dispositivo dispone de los siguientes botones y leds que se muestran en la imagen 1.0.0.1.

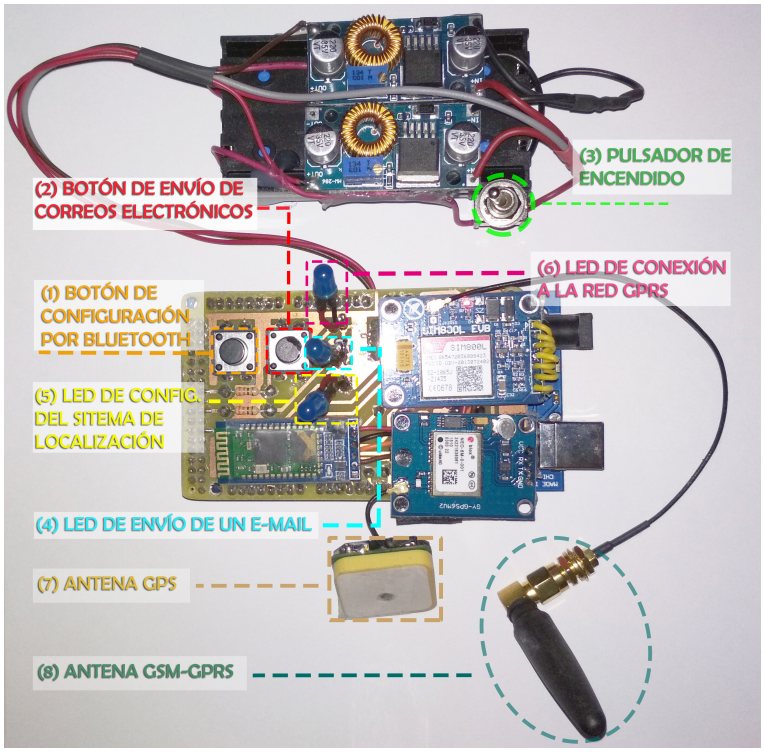



**Figura 1.0.0.1 –** Disposición de los elementos del localizador

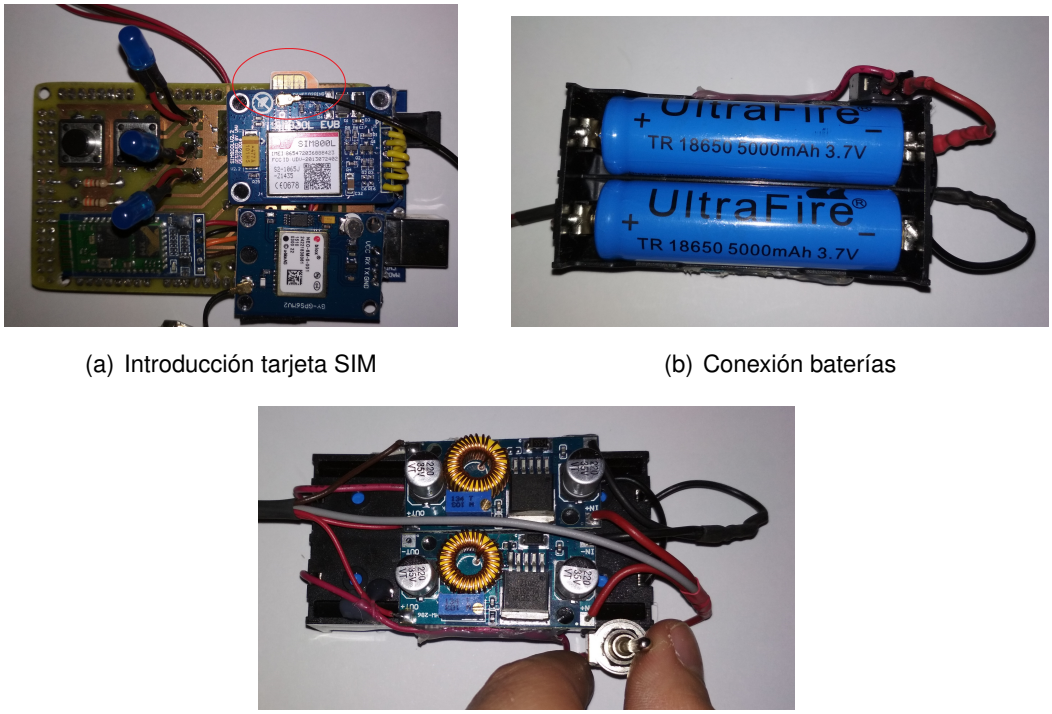
1. Botón de configuración por bluetooth.
2. Botón de envío de correos electrónicos.

3. Pulsador de encendido.
4. Led De envío de un E-mail.
5. Led de configuración del sistema de localización.
6. Led de conexión a la red GPRS.
7. Antena GPS
8. Antena GSM-GPRS

## 1.1. Puesta en marcha del equipo

Se deben seguir estas indicaciones antes de utilizar el dispositivo.

1. Colocar las 2 baterías con la polaridad indicada en el reverso soporte para las mismas.
2. Colocar la tarjeta SIM tal y como se muestra en la imagen (1.1.0.1(a)).
3. Encender el interruptor que poseen las baterías.




(a) Introducción tarjeta SIM

(b) Conexión baterías

(c) Encendido de interruptor

**Figura 1.1.0.1 –** Puesta en marcha del dispositivo localizador

Para más información acerca de la alimentación del dispositivo, se recomienda observar la sección (**??**).

## 1.2. Configuración del dispositivo mediante bluetooth

Uno de los pasos mas importantes para que el aparato funcione correctamente es la configuración inicial del mismo, ya que si se introduce en él información errónea, el aparato no podrá conectarse a la red o no enviara E-mails correctamente.

El aparato acepta una serie de mensajes que se recibirían por bluetooth, estos mensajes se representan en la siguiente tabla.

| **Comando** | **Descripción** | **Longitud máxima del <contenido> (caracteres)** | **Necesario** |
|---|---|---|---|
| **#usrname:<contenido>/r** | <contenido>: Nombre de usuario | 15 | Si |
| **#usrphpin:<contenido>/r** | <contenido>: Numero pin de la tarjeta SIM | 6 | Si |
| **#usremailkey<contenido>/r** | <contenido>: Clave que utiliza el usuario para entrar en su correo electrónico | 20 | Si |
| **#emailserver:<contenido>/r** | <contenido>: Nombre del servidor de correo electrónico utilizado | 20 | Si |
| **#emailport:<contenido>/r** | <contenido>: Puerto por el que el servidor envía el correo electrónico | 5 | Si |
| **#emailrmt:<contenido>/r** | <contenido>: E-mail del usuario | 30 | Si |
| **#emaildstname:<contenido>/r** | <contenido>: Nombre del destinatario del E-mail | 20 | Si |
| **#emaildst<contenido>/r** | <contenido>: E-mail destino | 30 | Si |
| **#usrhomelat:<contenido¿/r** | <contenido>: Latitud del punto de partida | 15 | No |
| **#usrhomelong:<contenido>/r** | <contenido>: Longitud del punto de partida | 15 | No |
| **#apnname<contenido>/r** | <contenido> Nombre del APN | 10 | Si |

**Tabla 1.2.0.1 –** Comandos bluetooth que acepta el dispositivo localizador

| **Comando** | **Descripción** | **Longitud máxima del <contenido> (caracteres)** | **Necesario** |
|---|---|---|---|
| **#usrapn:<contenido>/r** | <contenido>: APN de la tarjeta SIM | 20 | Si |
| **#usrpublicip:<contenido>/r** | <contenido>: Dirección ip publica | 20 | No |
| **#usrphnumber:<contenido>/r** | <contenido>: Numero de teléfono | 10 | No |
| **#usrbtname:<contenido>/r** | <contenido>: Nombre que le deseamos poner al dispositivo para su detección bluetooth | 15 | No |
| **#usrbtpin:<contenido>/r** | <contenido>: Pin para conexión bluetooth | 5 | No |

**Tabla 1.2.0.1 –** Comandos bluetooth que acepta el dispositivo localizador (continuación)

**NOTA:** En la tabla 1.2.0.1, la columna **"Necesario"** indica si el comando en cuestión es imprescindible para el correcto funcionamiento del dispositivo localizador. La mayor parte de los comandos marcados como no necesarios no intervienen en el funcionamiento de este dispositivo, aunque si serian necesarios en una posible nueva versión del mismo.

Algunos de los datos anteriores pueden ser un tanto difíciles de encontrar, por ello se explica a continuación como obtenerlos.

Se comienza por conocer el servidor y el puerto del del proveedor de correo electrónico que posea el usuario, dado que sin esta información no se pueden enviar correos electrónicos desde el presente dispositivo. El servidor a utilizar sera el SMTP del proveedor de correo electrónico que posea el usuario.

- Si el usuario dispone de una cuenta es de G-mail, se puede consultar el siguiente enlace https://support.google.com/a/answer/176600?hl=es.
- Si el usuario dispone de una cuenta hotmail se puede consultar este otro enlace http://www.serversmtp.com/es/smtp-hotmail-configuracion.

Para conocer el nombre del servidor SMTP de otra cuenta de correo electrónico distinta a las que se han mencionado anteriormente se recomienda indagar en internet.

Para encontrar el APN del proveedor de servicios de telefonía móvil que posea el usuario, también se puede encontrar en internet. A continuación se muestran unos ejemplos.

- Si el proveedor de servicios de telefonía del usuario es Movistar, el nombre del APN seria **Movistar** y el APN seria **telefonica.es** tal y como se muestra en https://atencionalcliente.movistar.es/pregunta-frecuente/como-configurar-el-apn-de-tu-movil/.
- Si el proveedor de servicios de telefonía del usuario es Vodafone, el nombre del APN seria **internet** y el APN seria **airtelwap.es** tal y como se muestra en https://ayudacliente.vodafone.es/particulares/moviles-tablets-y-apps/android/primeros-pasos-android/primeros-pasos-configurar-los-ajustes-de-internet-apn-de-tu-android/.

Para conocer las coordenadas del punto de partida, tanto latitud como longitud, se pueden consultar en Google maps https://www.google.es/maps haciendo click derecho en la zona que seria el punto de partida del usuario y seleccionando la opción "¿Que hay aqui?", el formato de introducirlas es el que se muestra en la siguiente imagen.

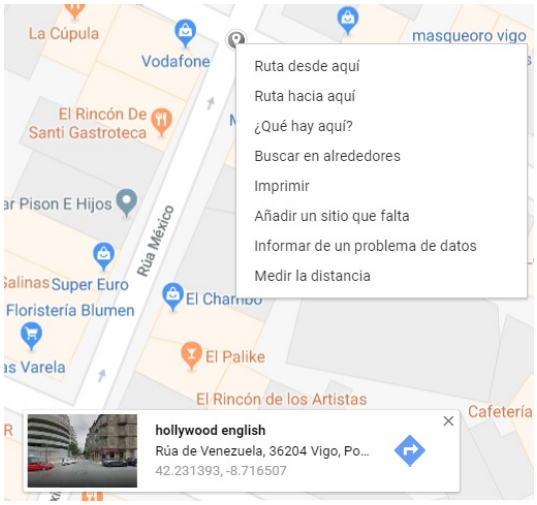



**Figura 1.2.0.1 –** Formato de las coordenadas que es necesario introducir en la aplicación

Tal y como muestra la figura 1.2.0.1, los datos que se deberían introducir en la aplicación serian:

- Latitud:42.231393
- Longitud:-8.716507

En este aspecto cabe comentar que en el caso de seguir ampliando este proyecto y se realice una aplicación para la configuración de este dispositivo (dado que no es el objeto principal de ese trabajo), la propia aplicación debería mostrar un mapa en el cual al presionar en una zona concreta del mismo, se obtendrían las coordenadas automáticamente para posteriormente enviárselas al dispositivo localizador.

Para enviar los comandos bluetooth al dispositivo, se recomienda disponer de un smartphone con bluetooth y una aplicación que permita enviar esos comandos.

Existen muchas aplicaciones que son capaces de enviar comandos a través de bluetooth, la que se ha utilizado y sobre la que se basara esta guía de configuración se llama **"Bluetooth terminal HC-05"**, para enviar los comandos anteriormente citados, es necesario realizar los siguientes pasos:

1. Una vez encendido el dispositivo localizador, se debe presionar el botón de configuración del bluetooth, una luz led nos indicara que el botón ha sido presionado. Ver (1.2.0.2).

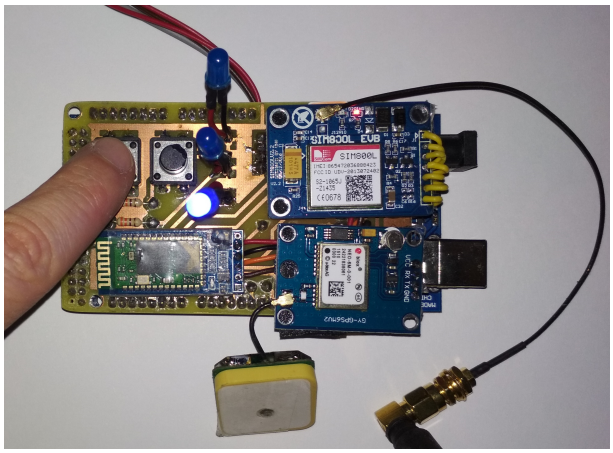
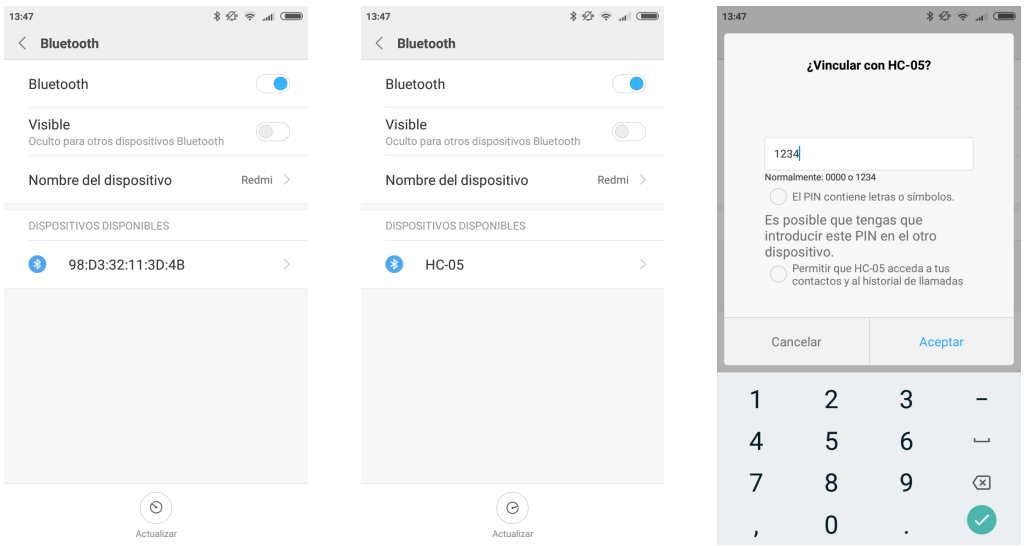


**Figura 1.2.0.2 –** El botón de configuración mediante bluetooth del dispositivo localizador se ha presionado

2. Si no se dispone de la aplicación **"Bluetooth terminal HC-05"** se recomienda su descarga.

3. Se habilita el bluetooth del smartphone. Ver (1.2.0.3).




(a) (b) (c)

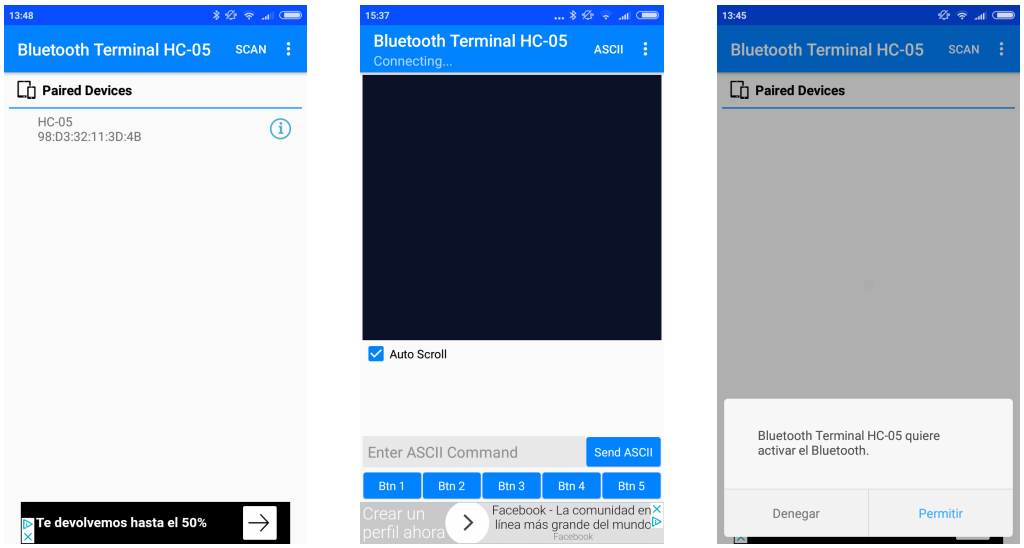
**Figura 1.2.0.3 –** Bluetooth del smartphone habilitado

4. El nombre del que dispone el dispositivo localizador por defecto es HC-05, en esta versión

del dispositivo, el nombre no se puede cambiar por los problemas que se comentan en el capitulo **??**. En este momento se empareja el Smartphone con el dispositivo localizador.

5. Se introduce la clave de emparejamiento (esta clave la pedirá la primera vez que se empareje), la clave por defecto es 1234 (ver 1.2.0.3 (c)).

6. Al abrir la aplicacion se puede observar que un dispositivo (HC-05) ya se ha emparejado con anterioridad. Se presiona sobre él. El dispositivo se emparejara de nuevo y se le podran enviar los comandos que se muestran en la tabla 1.2.0.1. Existen 2 formas de enviar comandos con esta aplicación.



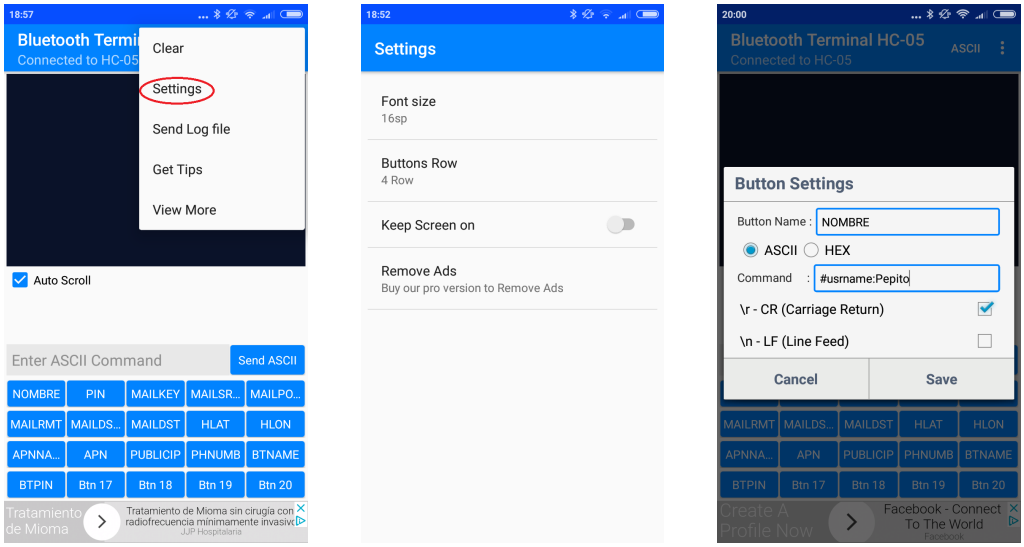

(a) (b) (c)

**Figura 1.2.0.4 –** Ejecutando la aplicación por primera vez

*a*) Forma1: Escribir el comando indicado en la tabla 1.2.0.1 y presionar el botón **“Send”**.

*b*) Forma2: Configurar los botones de la aplicación para poder enviar los comandos al presionarlos.

Para ello , al lado de la tecla ASCII ⇒ settings y se configura el entorno tal y como se muestra en la serie de imágenes siguientes.




(a) (b) (c)

**Figura 1.2.0.5 –** Configurando el entorno de la aplicación

Luego, en la ventana principal se configura uno a uno cada botón. Para configurar los botones se mantendrán presionados hasta que aparezca la ventana de configuración. En esta ventana, al lado del nombre **"Button name"** se introduce el nombre que se desea que tenga el botón de la aplicación, es necesario procurar ser breves y concisos en cuanto al nombre del botón, puesto que solo se muestran las 4 primeras letras del nombre que se le ponga, y al lado del nombre **"command"** se introduce uno de los comandos que se indican en la tabla 1.2.0.1.

Se configuraran en total 16 botones, dado que son 16 los comandos de configuración del dispositivo, de todas formas, se pueden obviar aquellos comandos que no sean necesarios, en total se tendrán entonces 10 botones configurados.

> **NOTA:** Es muy importante tener marcada la opción **"\r-cr(carriage return)"** para que se realice con éxito el envío de los comandos.

7. Se presionan cada uno de los botones que se han configurado en los pasos anteriores.

   A medida que se presionan los botones, el dispositivo envía al smartphone el numero de caracteres recibidos de cada comando, de esta forma se puede comprobar que el aparato esta recibiendo el mismo numero de caracteres que se le están enviando (Ver 1.2.0.6).

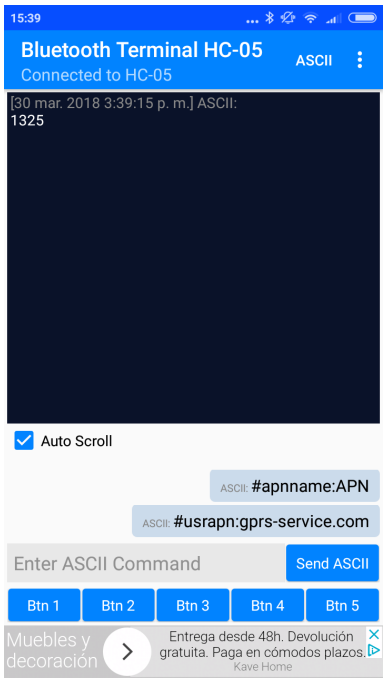



**Figura 1.2.0.6 –** Enviando comandos bluetooth a través de la aplicación del smartphone

8. Una vez configurado el dispositivo se vuelve a presionar el botón de configuración del bluetooth, la señal luminosa de configuración se apagara.

9. Se resetea el dispositivo.

## 1.3. Enviando un e-mail con el dispositivo localizador

Si se ha configurado el dispositivo localizador correctamente, esta operación se puede realizar con éxito. Tras configurar el aparato, haberlo apagado y luego, haberlo encendido de nuevo, el aparato se conectara a la red mostrando un led de los tres que dispone, encendido (habiéndolo configurado previamente de forma correcta).

El dispositivo localizador enviara cada 20 segundos la información relacionada con la posición del objeto a una pagina web, cuya url es la siguiente: `http://mapalocgps.appspot.com`.

Entonces, se puede presionar el pulsador de enviar e-mails.

En este punto pueden ocurrir dos cosas:

- Si en el momento en el que se esta enviando la información de la posición a la web se presiona el botón de enviar un e-mail, el e-mail no se enviara hasta que el envío de información a la web termine, sera en ese momento cuando el E-mail se envíe.

- Si en el momento de pulsar el botón no se esta enviando información a la web, inmediatamente se enviara el E-mail.

Durante el envío de un E-mail un led se encenderá cuando comience el envío y se apagara cuando acabe de realizarse el envío, tal y como se representa en la figura 1.3.0.1.

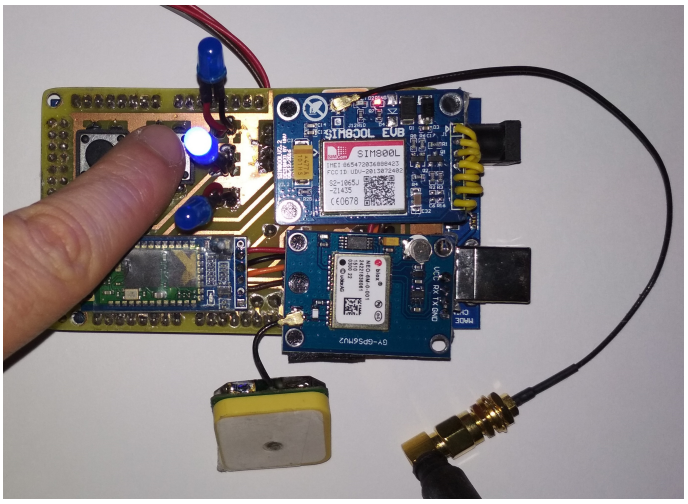


**Figura 1.3.0.1 –** El botón de de envío de e-mails del dispositivo localizador se ha presionado

El E-mail que se recibe tendrá la forma de la imagen 1.3.0.2.Dicho e-mail contiene un enlace a Google maps que indicara con un marcador el lugar en donde se encuentra nuestro dispositivo localizador.

Hay que tener en cuenta el tiempo que le lleva al GPS sincronizarse con los satélites, dado que puede ocurrir que si presionamos el botón de envío de un E-mail al poco de que el sistema de localización se conecte con la red, seguramente las coordenadas enviadas por el dispositivo sean incorrectas puesto que al GPS aun no le ha dado tiempo de sincronizarse con al menos tres satélites.

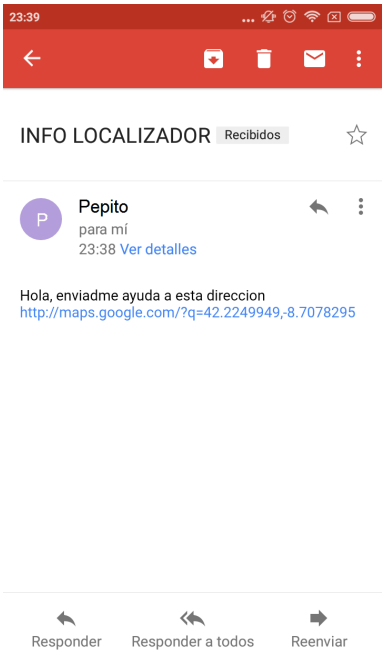


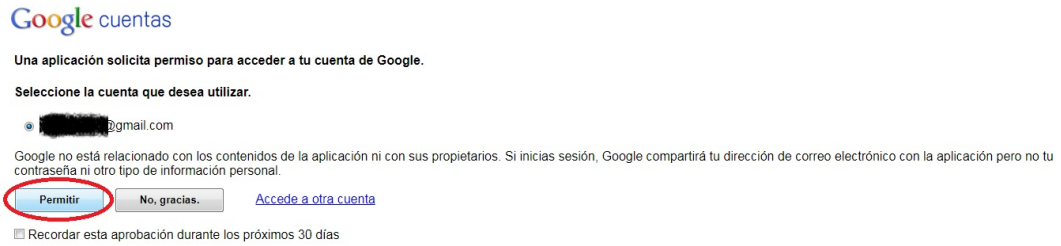

**Figura 1.3.0.2 –** E-mail recibido

## 1.4. Visualizar los datos en la pagina web

Una vez que el dispositivo localizador ha sido encendido, y se ha sincronizado con los satélites, comenzara a enviar información a la web con una cadencia de 20 segundos (en el apartado **??** se explican las razones). Los pasos para poder visualizar la posición son los siguientes:

1. En primer lugar es necesario loguearse con la cuenta de Google en la página que se ha creado. Un mensaje como el que se muestra en la figura (1.4.0.1(b)) se mostrara, se selecciona la opción **"Permitir"**.


(a) Login




(b) Mensaje tras el login

**Figura 1.4.0.1 –** Loguing en la pagina Web

2. Una vez el usuario se ha logueado, ya se puede ver la posición en la que se encuentra el dispositivo localizador. El entorno de la web es el que se muestra en la figura(1.4.0.2). En la parte suprior del mapa se muestran los datos referidos al ultimo punto enviado por el sistema de localización (latitud, longitud, altitud, fecha y hora, numero de satélites, velocidad y precisión de la altitud).


**Figura 1.4.0.2 –** Visualizar datos del localizador en la pagina Web

# 2 Especificaciones del equipo

A continuación se enumeran todas las especificaciones generales, eléctricas, térmicas y técnicas del dispositivo.

## 2.1. Especificaciones generales

1. Configuración mediante bluetooth.
2. Pulsador de emergencia para situaciones en las que se necesite ayuda.
3. Envío de correos electrónicos de forma segura utilizando protocolo SSL.
4. Visualización de la posición en la pagina web.

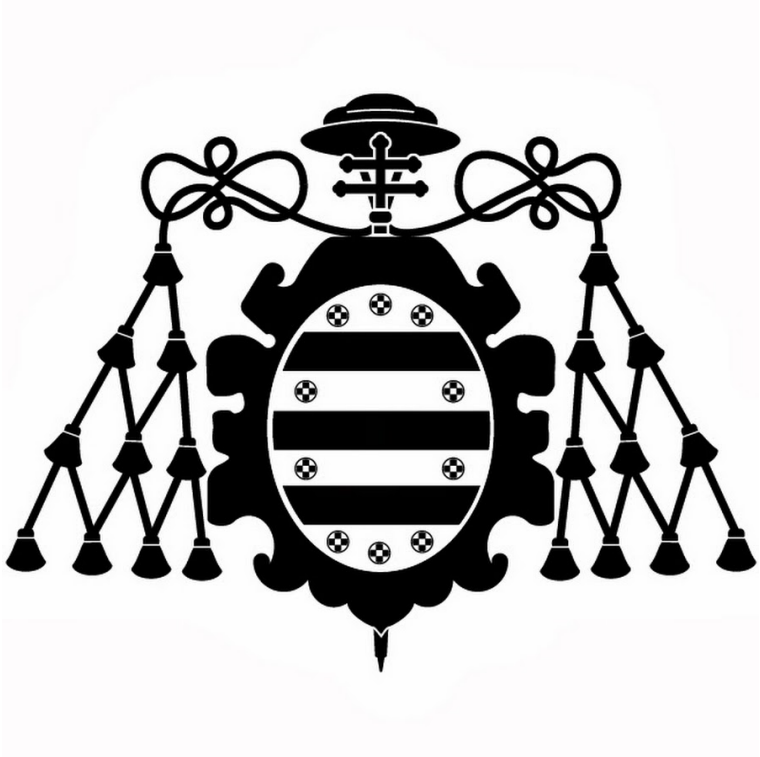
## 2.2. Especificaciones eléctricas y térmicas del equipo

1. **Alimentación:** Batería lipo 7,4V 5000mAh.
2. **Consumo medio:** Depende del uso.
3. **Rango de temperatura para su funcionamiento:** -10$^{o}$C – 50$^{o}$C.

## 2.3. Especificaciones técnicas

1. **Cadencia mínima de envío de datos a la web:** 20 segundos.
2. **Error de precisión** 10m.
3. **Banda GSM:** Cuatribanda.
4. **Voz:** No.
5. Batería interna recargable y reemplazable 5000 mAh y (7.4V).
6. **Memoria Interna:** No.
7. **Método de comunicación:** GPRS.
8. **Posición por:** Tiempo y Distancia.

9. **Ahorro de Energía:** No.
10. **Antenas:** GPS, GSM.
11. **Cubierta:** Plástica.
12. **Fabricado en:** España.



# PLANOS

# Índice del documento PLANOS



# Índice de figuras

# Índice de tablas

# 1 Documentación sobre los planos

Plano 1: Este plano nos ofrece una visión general de la conexión de los diferentes circuitos a la placa Arduino para un correcto funcionamiento del software. Siguiendo el conexionado de este plano, podremos montar el circuito sin ningún tipo de problema en una placa de prototipos (disponiendo de todos los componentes necesarios para el montaje).

Plano 2: Este plano es el diseño de una placa que se acoplara a nuestro Arduino MEGA y que se encarga de realizar las conexiones entre el Arduino MEGA y el resto de componentes para ahorrar cableado.

Plano 3: En este plano se muestran las pistas de la placa de circuito impreso citada anteriormente por la cara top (parte de arriba de la placa) y por la cara bottom (parte de abajo de la placa).

Plano 4: En este plano figura el esquema electrónico del sistema utilizado para la carga de las baterías de litio.

Plano 5: En este plano figura el esquema electrónico del modulo utilizado para la comunicación GPRS y enviar E-mails (SIM800L).

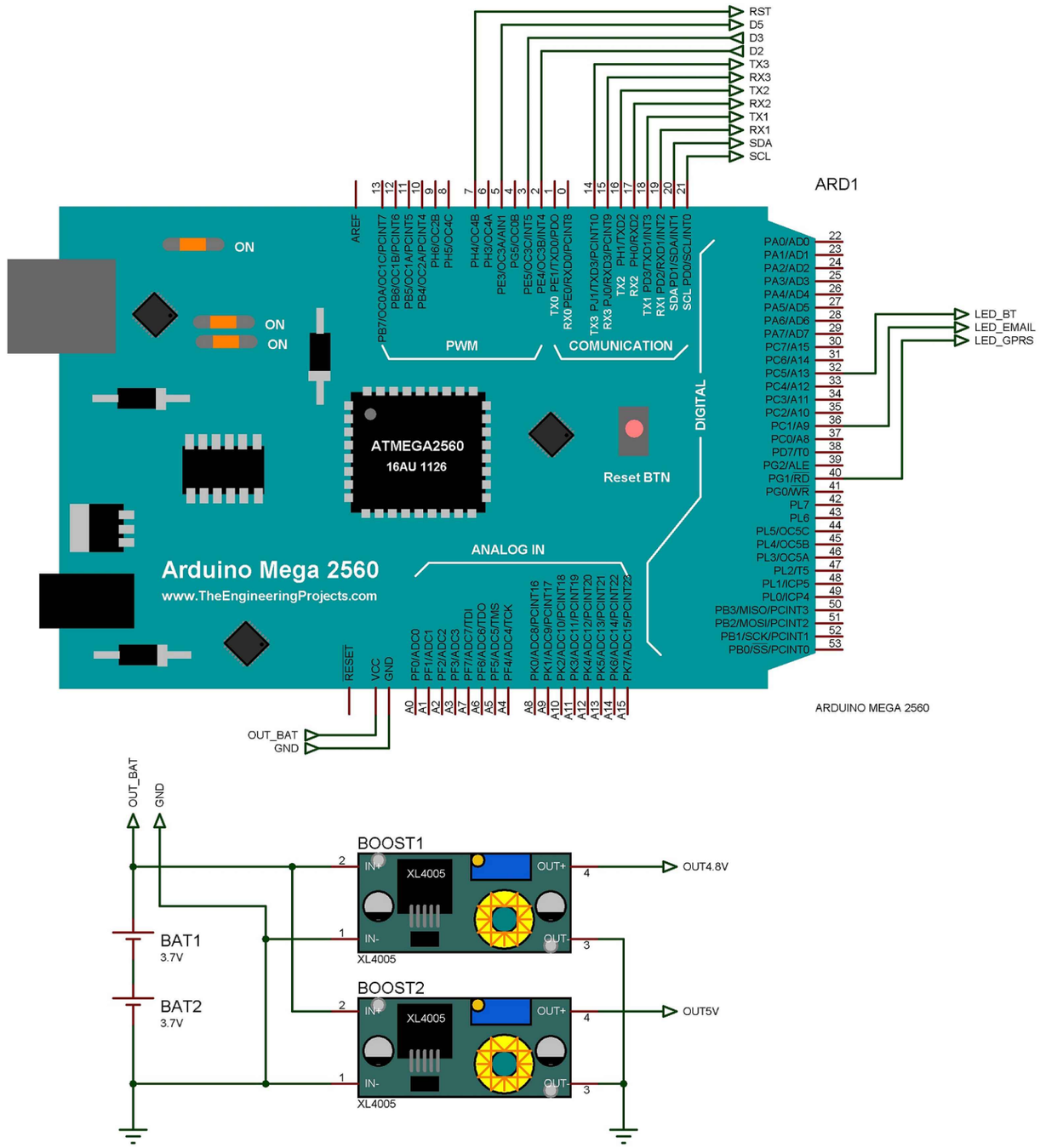
Plano 6: En este plano figura el esquema electrónico del modulo Bluetooth HC-05 utilizado en este proyecto.

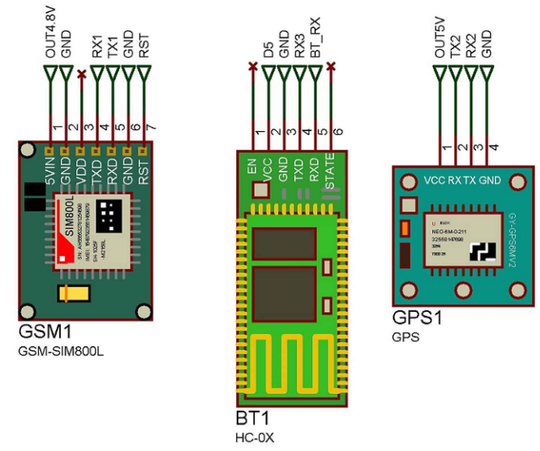
Plano 7: En este plano figura el esquema electrónico del modulo GPS UBLOX NEO-6M utilizado en este proyecto.

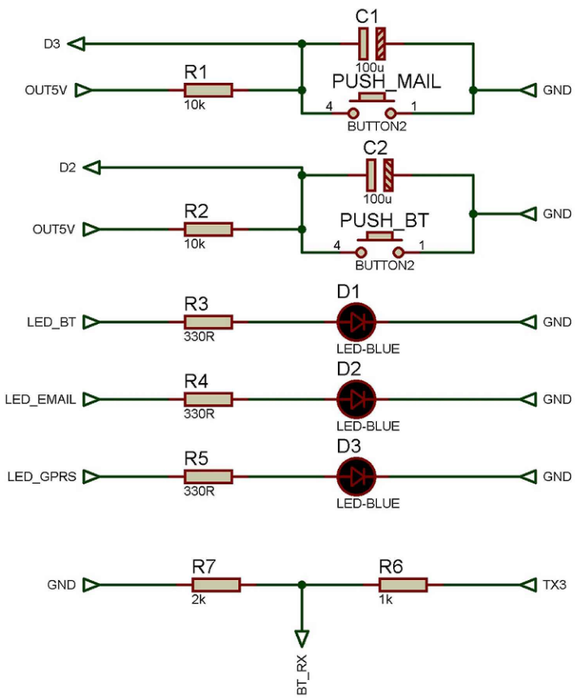
Plano 8: En este plano figura el esquema electrónico del modulo 6009 elevador de tensión utilizado para la realización de pruebas en este proyecto.

Plano 9: En este plano figura el esquema electrónico del modulo TP4056 encargado de cargar las baterías utilizado en este proyecto.

Plano 10: En este plano figura el esquema electrónico del modulo reductor de tensión XL4005 utilizado en este proyecto.

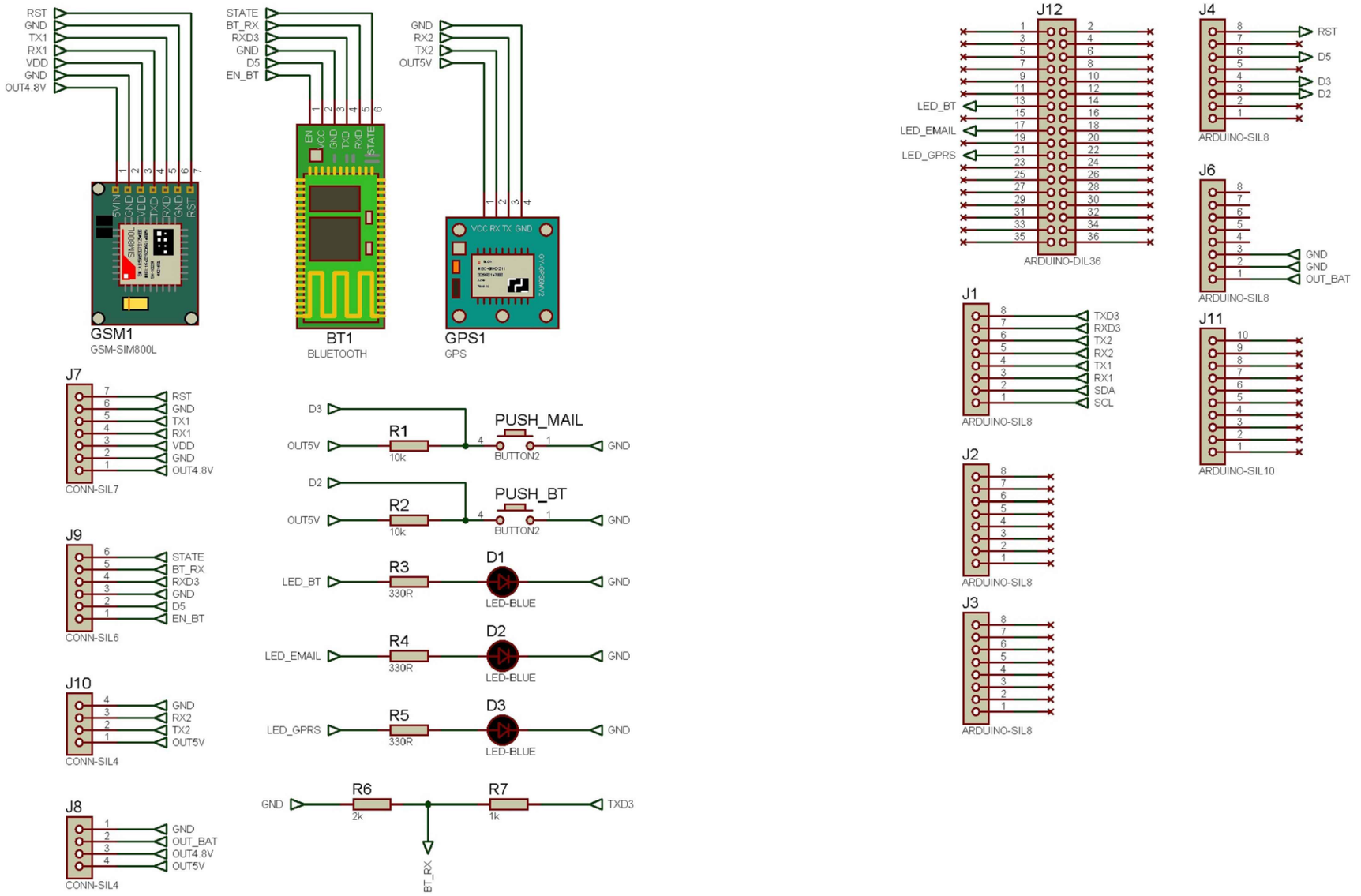



| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 10/02/2018 | | | PLANO GENERAL DE MONTAJE |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |

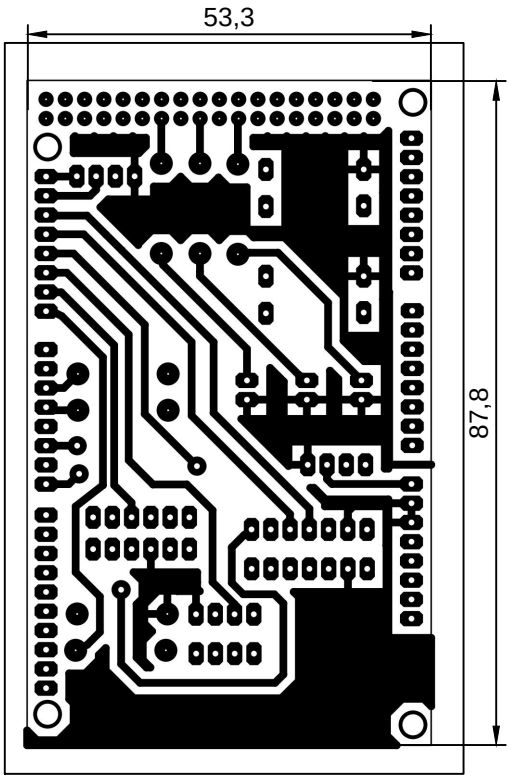
A3



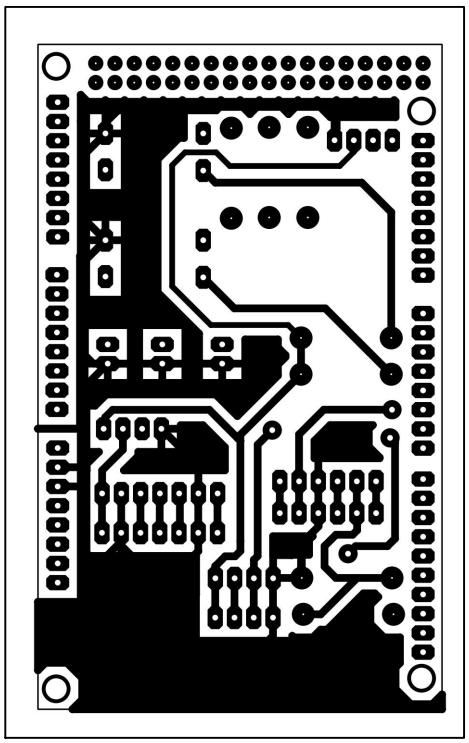

| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 25/02/2018 | | | PLANO DE LA PLACA DE ACOPLE A ARDUINO MEGA |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |

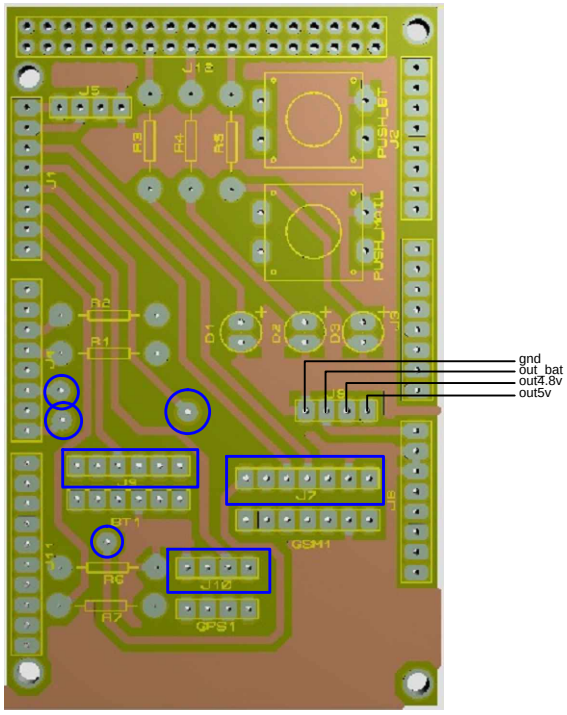
A3

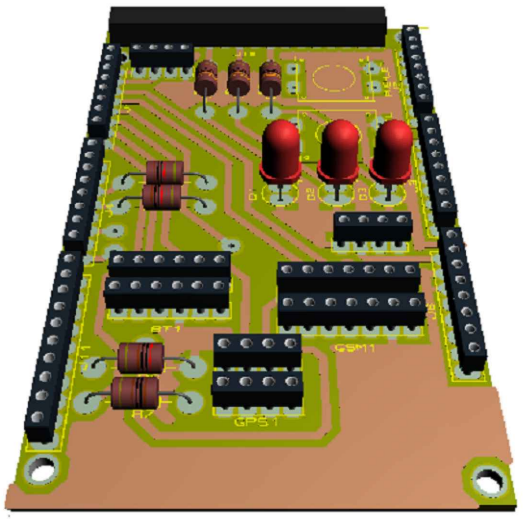
PARTE DELANTERA




PARTE TRASERA

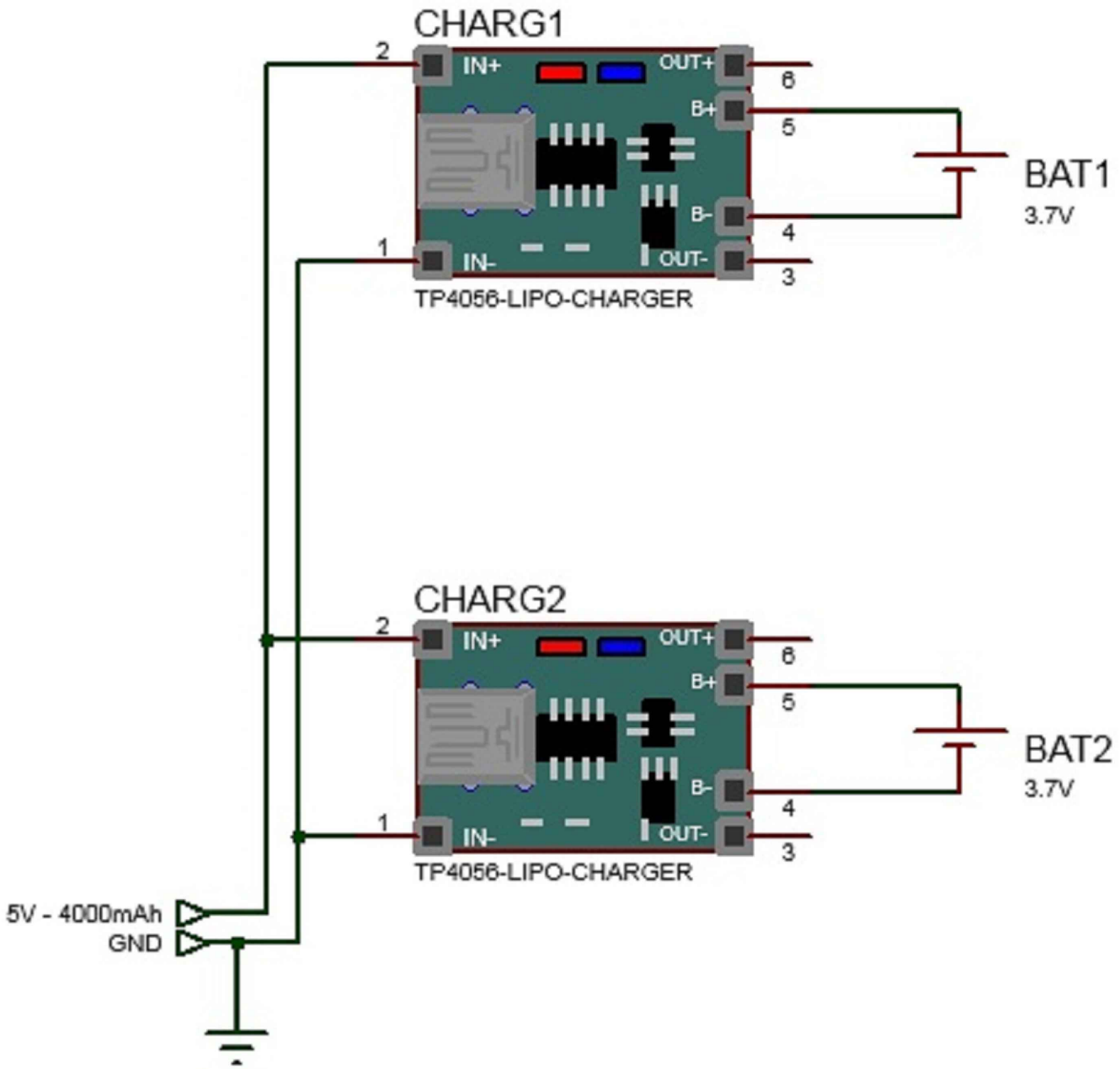




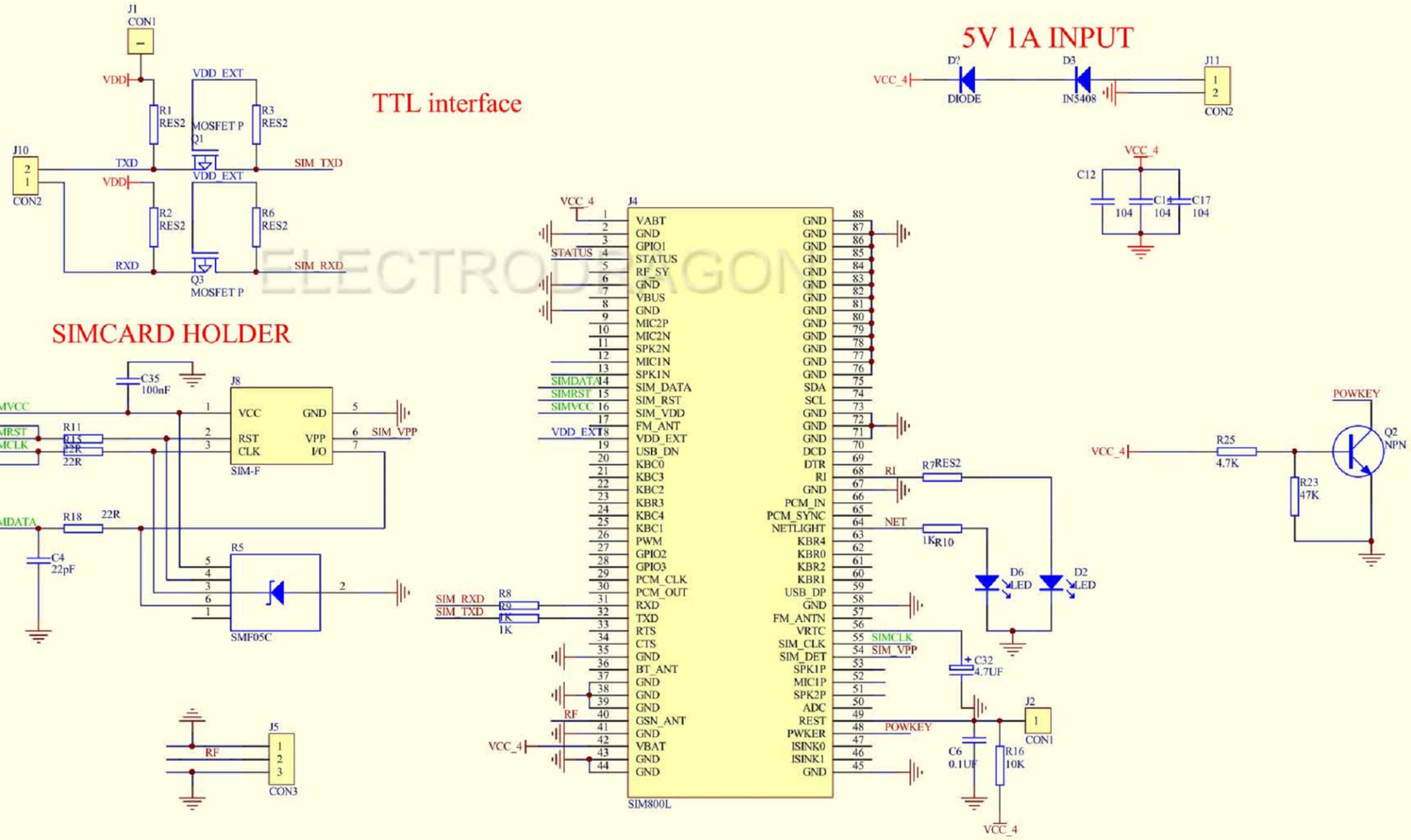



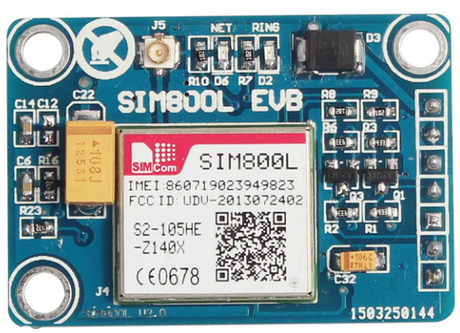
NOTA: Los conectores señalados con un recuadro/circulo azul son vías.
NOTA: Se recomienda soldar los componentes de agujero pasante por ambas caras si es posible.

| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 10/02/2018 | | | PLANO DE LAS PISTAS POR LA CARA DELANTERA Y POR LA CARA TRASERA DE LA PLACA DE ACOPLE |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |

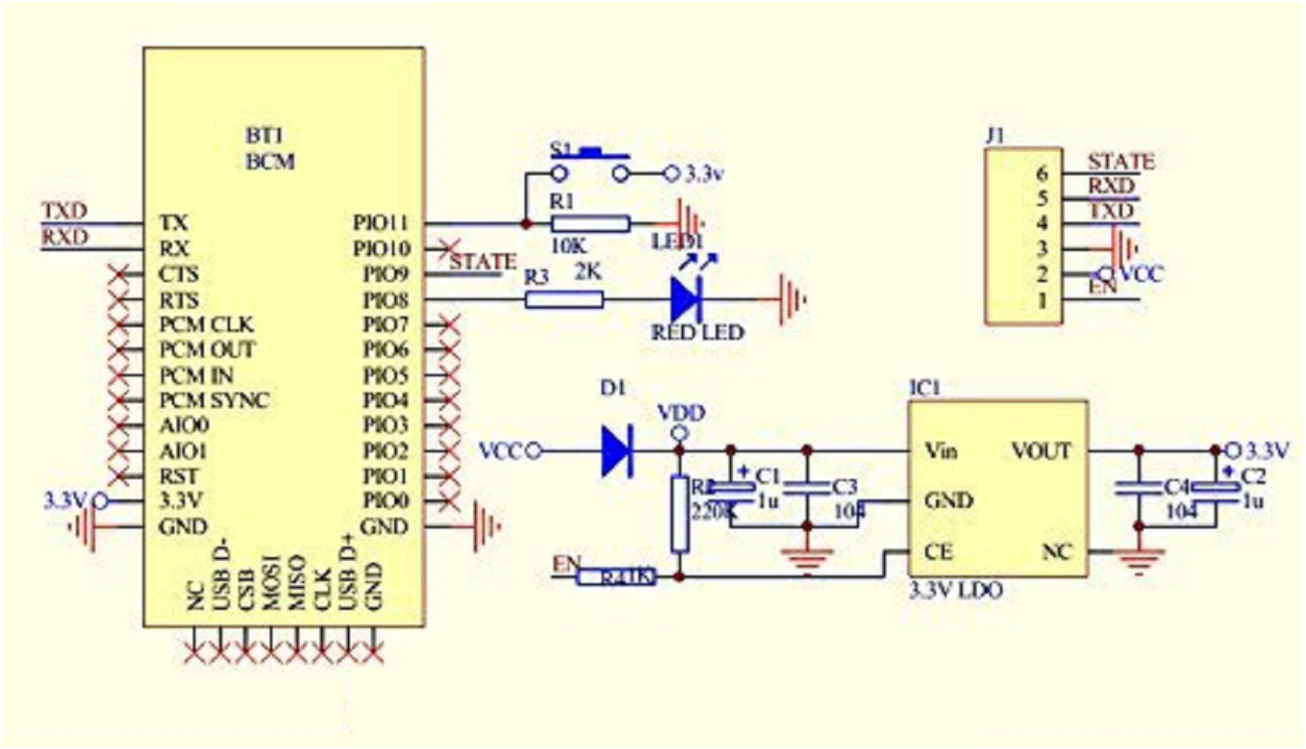


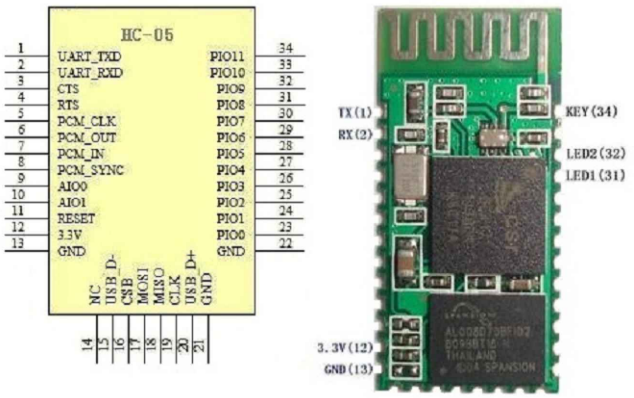

| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 18/02/2018 | | | PLANO DEL CIRCUITO DE CARGA DE LAS BATERIAS |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |



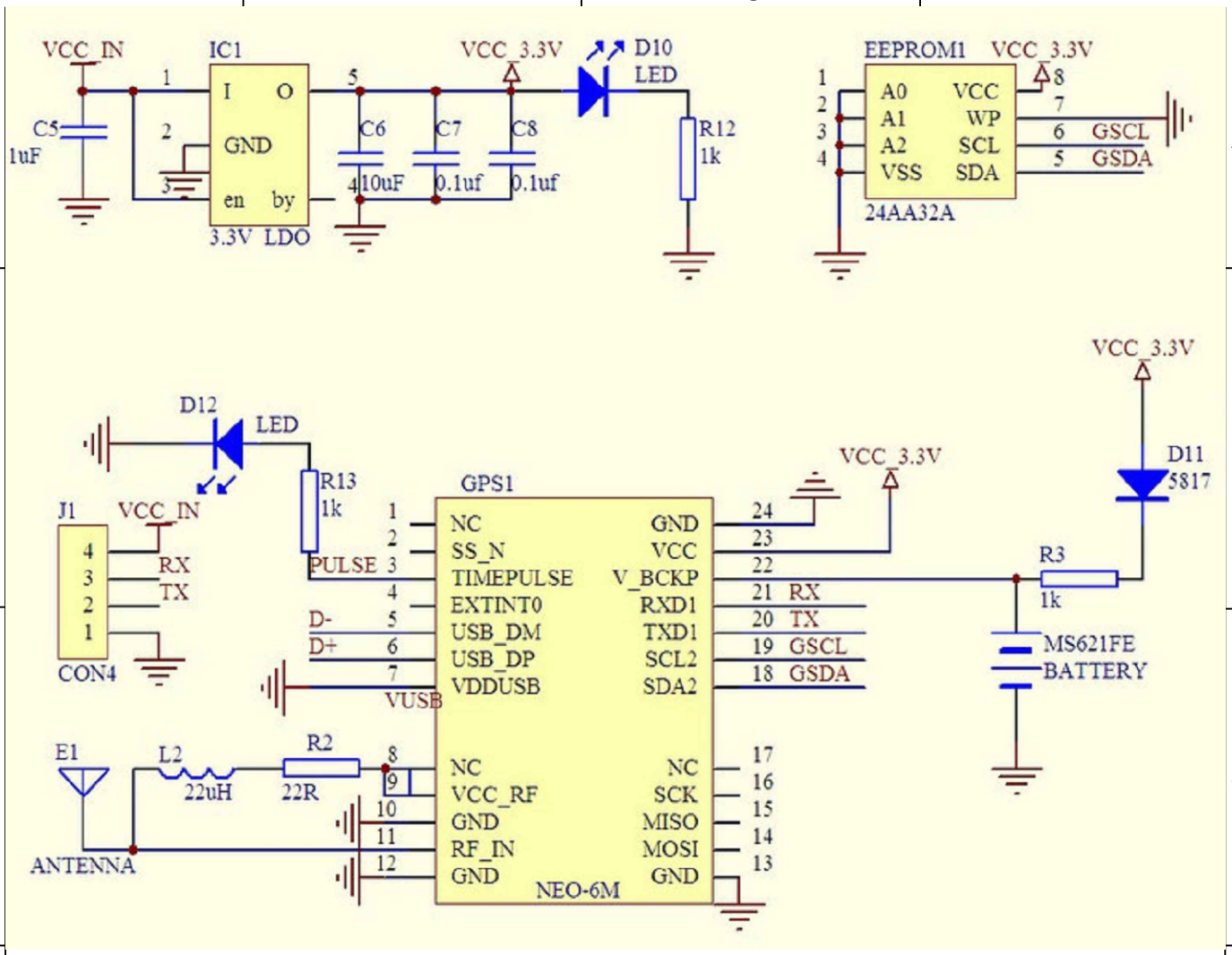




| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 10/02/2018 | | | PLANO DEL MODULO SIM800L |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |

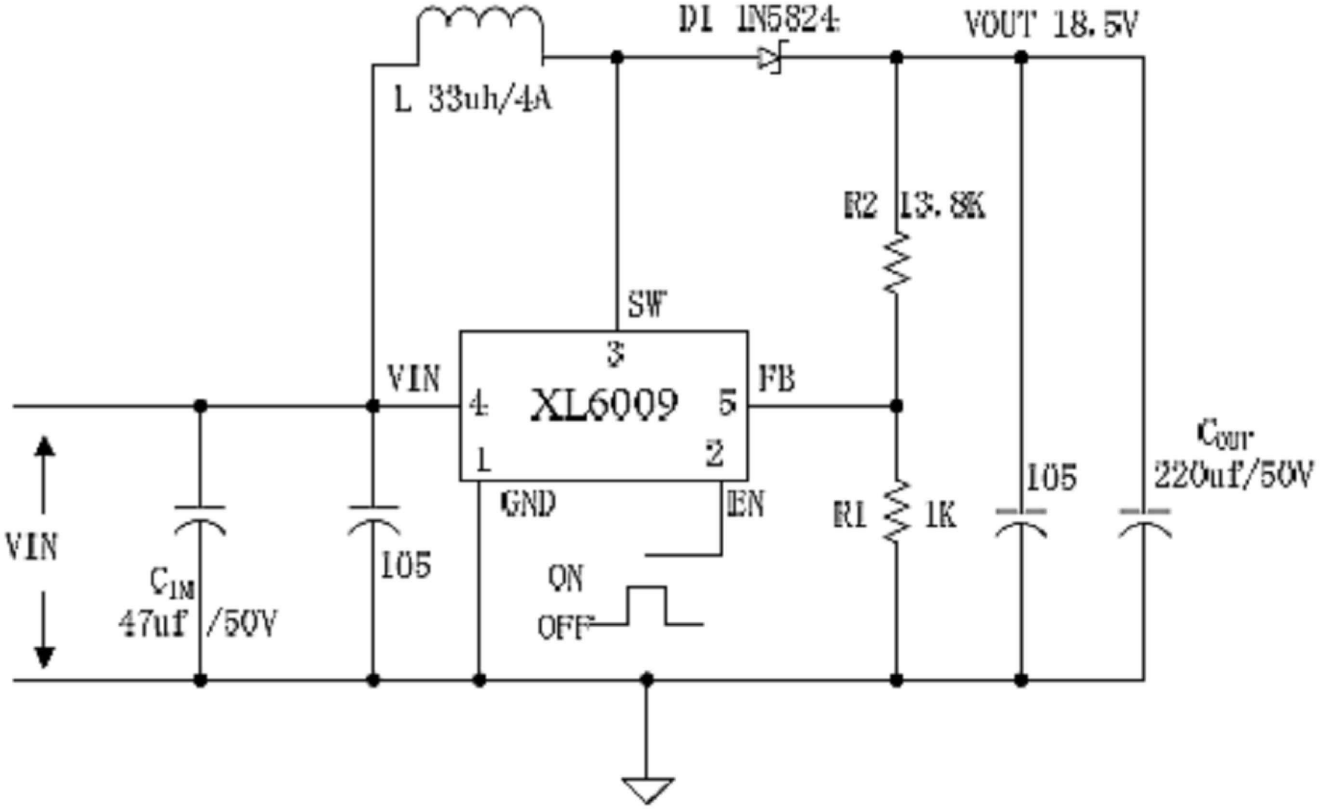


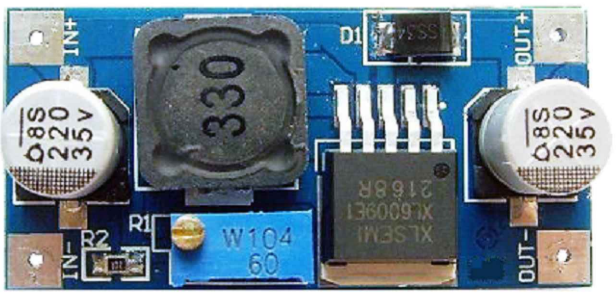




| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 10/02/2018 | | | PLANO DEL MODULO BLUETOOTH HC-05 |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |

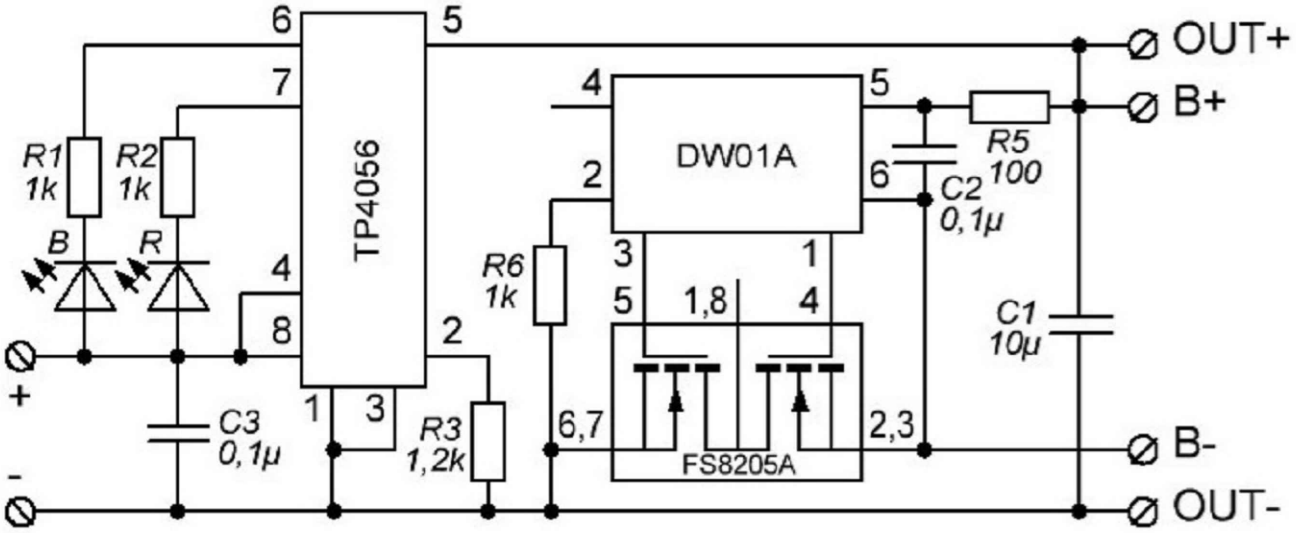


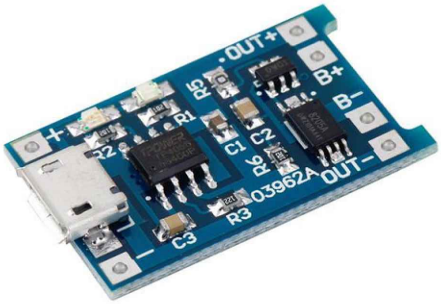

| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 10/02/2018 | | | PLANO DEL MODULO GPS UBLOX NEO-6M |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |




| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 10/02/2018 | | | PLANO DEL CONVERTIDOR BOOK-BOOST XL6009 |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |




| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 18/02/2018 | | | PLANO ELECTRÓNICO DEL MODULO TP4056 (CARGADOR DE BATERÍAS DE LITIO) |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |

## Typical System Application for 24V ~ 5V/5A

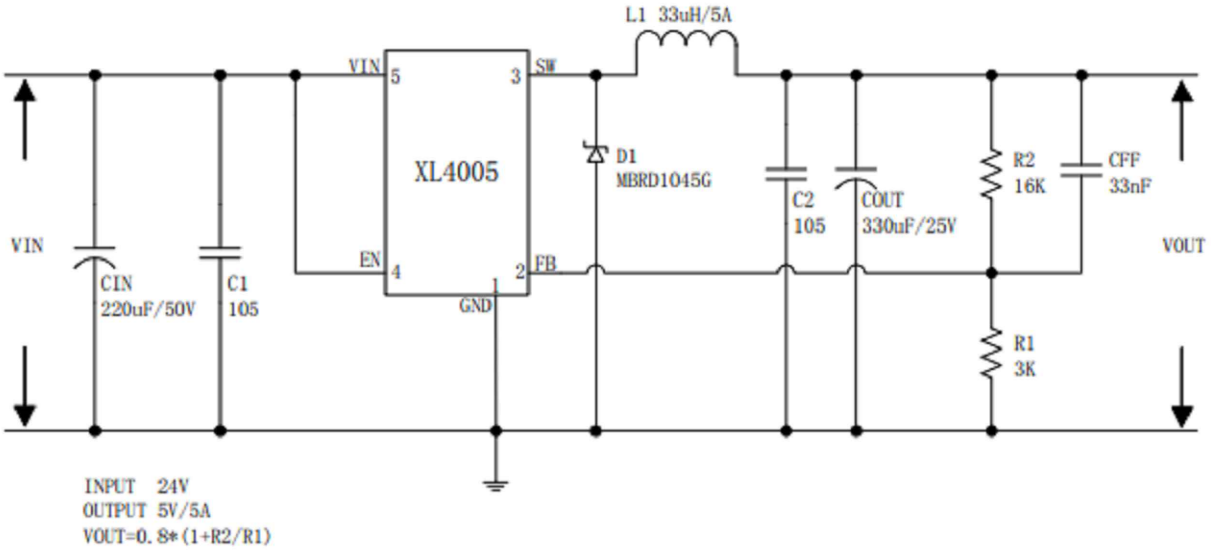



Figure7. XL4005 System Parameters Test Circuit (24V ~ 5V/5A)

| | FECHA | NOMBRE | FIRMAS | TÍTULO DEL PLANO |
|---|---|---|---|---|
| DIBUJADO | 18/03/2018 | | | PLANO ELECTRÓNICO DEL MODULO XL4005 (STEP DOWN CONVERTER) |
| MODIFICADO | | | | |
| ESCALA 1:1 | TÍTULO DEL TFM: DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE LOCALIZACIÓN EN TIEMPO REAL DE OBJETOS | | | Autor: Cristóbal García Camoira |

# 2 Lista de materiales según plano general de montaje

## 2.1. Dispositivos del equipo

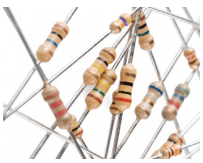
| Uds | Descripción | Imagen | Referencia a Plano[1] |
|---|---|---|---|
| 1 | Bluetooth HC-05 |  | BT1 |
| 2 | 140 PCS 10 X 14 Tipos SMD Micro Interruptor |  | PUSH_MAIL, PUSH_BT |
| 1 | Módulo SIM 800L |  | SIM1 |
| 1 | Arduino MEGA 2560 |  | ARD1 |
| 1 | Modulo GPS GY-GPS6MV2 |  | GPS1 |
| 3 | Resistencias 330Ω |  | R3, R4, R5 |

**Tabla 2.1.0.1 –** Tabla de los componentes utilizados en el dispositivo localizador

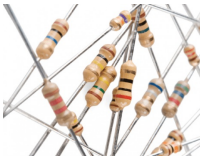
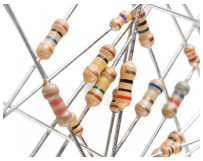
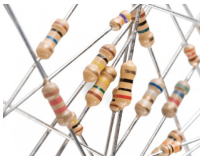
| Uds | Descripción | Imagen | Referencia a Plano[1] |
|---|---|---|---|
| 2 | Resistencias 10KΩ |  | R1, R2 |
| 2 | Resistencias 1KΩ |  | R6 |
| 2 | Resistencias 2KΩ |  | R7 |
| 3 | Leds |  | D1, D2, D3 |

**Tabla 2.1.0.1 –** Tabla de los componentes utilizados en el dispositivo localizador

## 2.2. Placa De acople a Arduino

| Uds | Descripción | Imagen | Referencia a Plano [2] |
|---|---|---|---|
| 1 | Tira de pines macho recto |  | J1, J2, J3, J4 , J11, J6, J12 |

**Tabla 2.2.0.1 –** Lista de materiales 2

## 2.3. Alimentación del dispositivo

| Uds | Descripción | Imagen | Referencia a Plano[1] |
|---|---|---|---|
| 2 | DC-DC Adjustable XL4005 Buck Step Down Power Module |  | BOOST1, BOOST2 |
| 2 | Batería Litio 3.7V 5000 mAH |  | BAT1, BAT2 |

**Tabla 2.3.0.1 –** Lista de materiales 3

## 2.4. Carga del dispositivo

| Uds | Descripción | Imagen | Referencia a Plano[4] |
|---|---|---|---|
| 2 | TP4056 + Battery protection LIPO Charger Module Board Micro USB Parts TE420 |  | CHARG1, CHARG2 |
| 2 | Batería Litio 3.7V 5000 mAH |  | BAT1, BAT2 |

**Tabla 2.4.0.1 –** Lista de materiales 4

## 2.5. Tarjeta de comunicaciones y terminal móvil

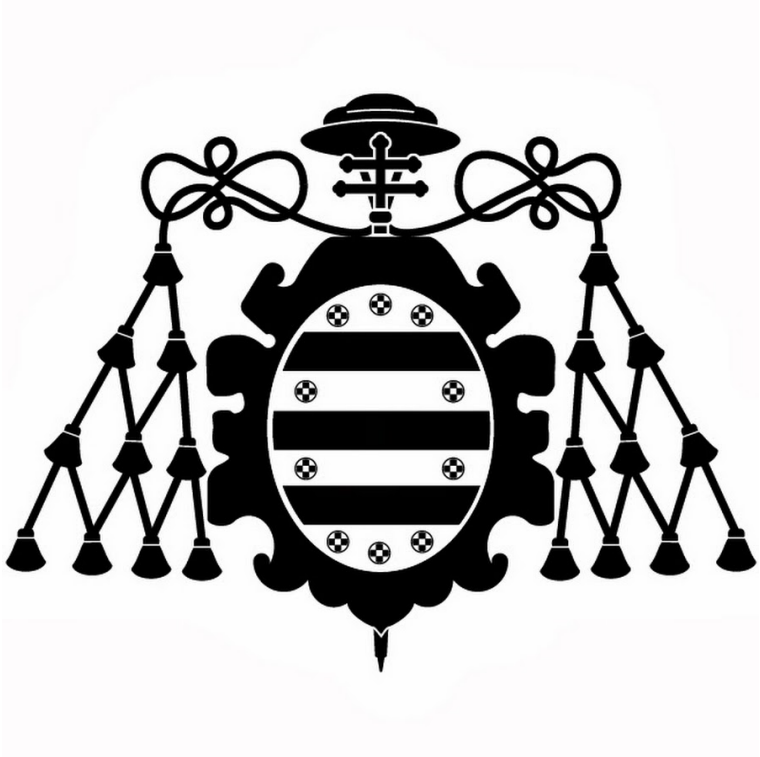
| Uds | Descripción | Imagen | Referencia a Plano |
|---|---|---|---|
| 1 | Tarjeta SIM |  | No hay referencia en planos |

**Tabla 2.5.0.1 –** Lista de materiales 5

| Uds | Descripción | Imagen | Referencia a Plano |
|---|---|---|---|
| 1 | Terminal móvil |  | No hay referencia en planos |

**Tabla 2.5.0.1 –** Lista de materiales 5

Tanto la tarjeta de comunicaciones (Tarjeta SIM),como el terminal móvil, se deja a criterio del futuro usuario la elección de los mismos.En el caso de la tarjeta de comunicaciones, se recomienda obtener información de aquella compañía que nos ofrezca la mejor tarifa de datos y SMS.



# PLIEGO DE CONDICIONES

# Índice del documento PLIEGO DE CONDICIONES

# Índice de figuras

# Índice de tablas



# Listado de códigos de programación

# 1 Especificaciones de los materiales

Los materiales y componentes utilizados en la realización del presente proyecto han sido aquellos que reunían los requisitos para ser utilizados. Además éstos componentes deben asegurar un correcto funcionamiento durante la vida útil del dispositivo, en este caso el sistema de localización en tiempo real de objetos. El subministrador debe garantizar el funcionamiento correcto de cada componente según las hojas características del mismo.

# 2 Pruebas de verificación

A continuación se detallan las pruebas de verificación necesarias para comprobar el correcto funcionamiento de cada uno de los módulos que componen el sistema de localización en tiempo real de objetos, aunque también se detallan algunas pruebas de funcionamiento de algunos dispositivos que no irán ensamblados en el prototipo final (como por ejemplo el LCD) pero que durante la fase de pruebas ha sido necesario utilizarlo (como por ejemplo para ver posición y hora enviada por el GPS en la pantalla).

## 2.1. Programa Arduino

Para la realización de las pruebas de verificación se ha utilizado la versión IDE 1.0.5 instalable de Arduino, todos los programas descritos a continuación y realizados para poder verificar el funcionamiento de los componentes están basados en esta versión y comprobados prácticamente, por tanto no se puede asegurar el funcionamiento de los mismos programas cargados a Arduino desde versiones del programa posteriores.

## 2.2. Prueba de funcionamiento del LCD

Se debe comprobar que el LCD enciende correctamente con tan sólo introducirle la tensión de alimentación. Se recomienda realizar pruebas de que funcionan las cuatro líneas disponibles.

Para ello, se conectara el LCD de la forma que se indica la imagen [2.2.0.1] a Arduino, ya sea UNO, LEONARDO o MEGA....

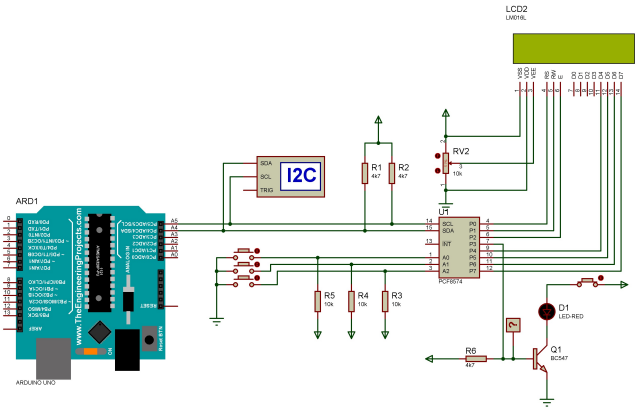



**Figura 2.2.0.1 –** Conexionado de Arduino UNO a un lcd mediante la comunicacion I2C

Se sube el siguiente programa a la placa Arduino el cual muestra en las dos líneas del LCD “Hola Mundo”.

```
//*************************************************************************
// Autor : CRISTÓBAL GARCÍA CAMOIRA
//*************************************************************************
/*
//*************************************************************************
Este programa imprime en un lcd de 16 x 2 lineas controlado por I2C
los caracteres "hola mundo"
//*************************************************************************
*/

#include <Wire.h> // La necesita la libreria LiquidCrystal_I2C.h
#include <LiquidCrystal_I2C.h>    // Para usar un display I2C
LiquidCrystal_I2C lcd(0x27,16,2); // Establece la dirección de memoria 0x27 para un
    display de 20 caracteres y 4 líneas.

void setup()
  {
    lcd.init(); // Inicializo el LCD.
    lcd.backlight(); // Brillo de la pantalla encendido.
    lcd.setCursor(0,0);
    lcd.print("Hola Mundo");
    lcd.setCursor(0,1);
    lcd.print("Hola Mundo");
  }

void loop() {}
//*************************************************************************
```

**Código 2.1:** Prueba de funcionamiento del LCD

NOTA: Si por algún error del montaje no se muestra nada en la pantalla, verifique que el conexionado este correcto, y también verifique el estado de los potenciómetros de contraste y brillo, es un error muy común.

## 2.3. Arduino

Debe comprobarse que el Arduino utilizado está en perfectas condiciones, se recomienda probar las salidas que se van a utilizar. También se debe comprobar el regulador de tensión de 5v de Arduino.

## 2.4. Pruebas de funcionamiento de pulsadores y teclados

### 2.4.1. Control de matriz de pulsadores mediante 1 pin analógico

En esta prueba de funcionamiento se muestra el conexionado a realizar para el control de un teclado matricial mediante la lectura de un único pin analógico.

Este tipo de conexionado es ideal para aquellas aplicaciones que no necesiten un tiempo de reacción elevado y una alta eficiencia en el código de programación, puesto que es necesario estar muestreando con cierta frecuencia la lectura analógica y en función del valor recibido, poder saber sobre que pulsador se esta actuando. Ademas este método impediría que el microcontrolador pueda estar en modo sleep y despertarse ante una interrupción de pulsador haciendo que el micro consuma mas, e influyendo en el gasto de batería.

Si lo que se necesita es la realización de un sistema dedicado explícitamente para la lectura de un teclado, o una serie de pulsadores de forma analógica, esta seria una buena forma de implementarlo.

Este tipo de conexionado, se sigue utilizando en automoción para la lectura de los pulsadores situados en la guantera, volante...

En el ejemplo que se explica a continuación, se efectuá la lectura del valor analógico de tensión con el conversor AD de Arduino procedente de un divisor resistivo tras la pulsación de una tecla del teclado matricial. El divisor resistivo utilizado para la lectura es el que se muestra en la imagen [2.4.1.2].

Un teclado matricial 4x4 es un dispositivo compuesto por 4x4 teclas con 4+4 lineas que conectan entre si las teclas, una linea por cada fila de teclas mas una linea por cada columna de teclas. Al ser pulsada una cualquiera de ellas une entre sí una de las lineas, la de su columna, con otra de ellas, la de su fila. Así al pulsar una tecla quedan unidas solo dos de las ocho que tiene.

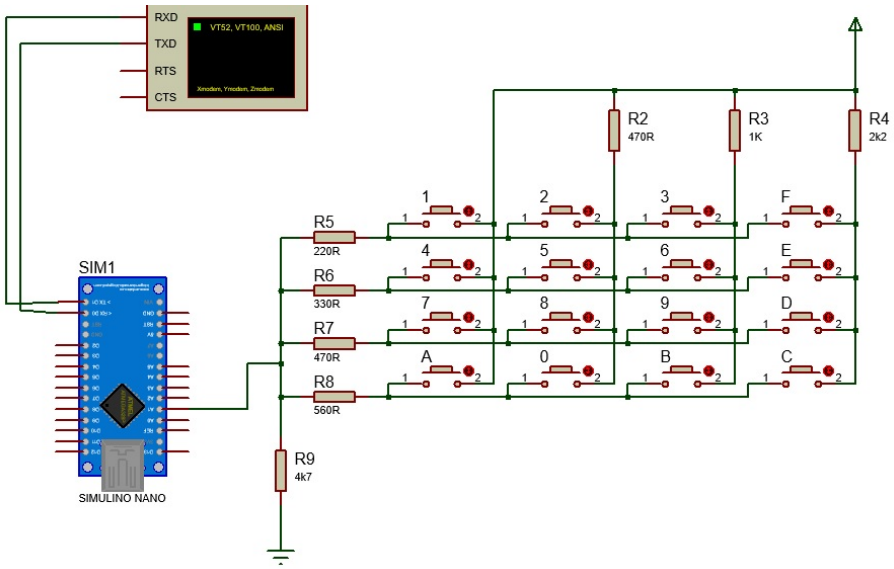



**Figura 2.4.1.1 –** Conexionado de teclado analogico a Arduino NANO

En la imagen anterior 2.4.1.1 se puede apreciar que al pulsar une una de las resistencias de R1 a R4 conectadas a VDD con otra de R5 a R8 conectadas al Arduino . Así si se presiona la tecla situada en la esquina superior izquierda se obtiene que la tension de alimentacion del circuito del teclado (VDD) llega a Arduino tras atravesar R1+R5.

Si por el contrario se presiona la tecla inferior derecha la corriente atravesara la unión entre R4+R8. Siempre que se presione una tecla cualquiera se obtiene una caída de tensión entre la suma de dos resistencias Rcolumna+Rfila. Cada tensión procede de una combinación diferente de resistencias y por lo tanto ninguna sera idéntica.

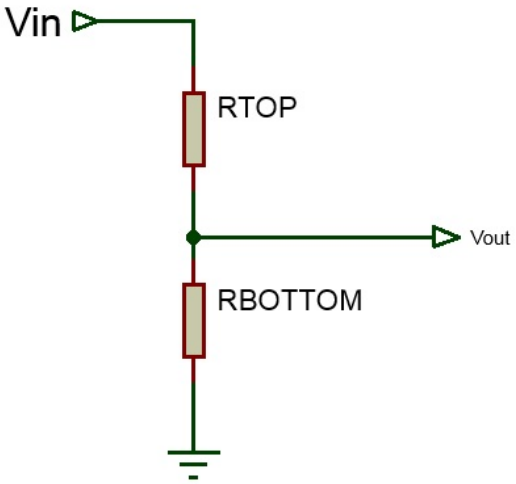



**Figura 2.4.1.2 –** Divisor de tensión implementado en para la lectura de cada pulsador

Otro detalle a tener en cuenta es que si no se presiona ninguna tecla, el pin analogico de Arduino estaría conectado a nada, la linea que une e Arduino con las resistencias R5 a R8 y tras ella el vacío. En este caso, con total seguridad, el pin analógico de Arduino recogería tensiones residuales procedentes del ruido ambiente, como si de una antena se tratase, dando

lugar a todo tipo de lecturas erroneas mientras no se presione ninguna tecla. Para evitar ese efecto, se introduce la resistencia R9 que mantendrá el pin del conversor AD conectado a masa (GND) mientras no se presione ninguna tecla.

Esta configuración se conoce como un Divisor de Tensión en la que se posee una resistencia conectada a VDD y otra a GND y se toma el valor del voltaje de la unión que hay entre ellas.

```
1  //*********************************************************************
2  // Autor : CRISTÓBAL GARCÍA CAMOIRA
3  //*********************************************************************
4  int entrada_analogica_teclado = A1;
5  int valor_analogico_teclado = 0;
6
7  void setup(){
8    Serial.begin(9600);
9  }
10
11 void loop(){
12   if(analogRead(entrada_analogica_teclado)>10){
13     valor_analogico_teclado = analogRead(entrada_analogica_teclado);
14     Serial.println(valor_analogico_teclado);
15     while(analogRead(entrada_analogica_teclado)>10) delay(1);
16   }
17 }
```

**Código 2.2:** Control de teclado mediante 1 pin analógico

Este divisor de tensión en el que se posee un (Vin) o voltaje de entrada y un (Vout) o voltaje de salida tras él, que es perfectamente calculable mediante la siguiente formula.

$Vout = (\frac{Rbottom}{Rbottom+Rtop})xVin$

Como vemos en esta configuración lo que se denomina Rtop es lo que en el teclado se ha denominado Rcolumna+Rfila, es decir, la suma de las dos resistencias correspondientes al pulsar una tecla en él. Y Rbottom es la resistencia R9 del teclado.

Sabiendo que Rtop es Rc+Rf y que VDD es 5V se concluye que Vout = R9 / R9+Rc+Rf * 5V y así se posee un valor de Vout para cada pareja de resistencias Rc+Rf.

A continuación se muestra una tabla (Tabla 2.4.1.1) en la que se ha puesto primero la tabla de resistencias de columnas y filas y las distintas sumas de cada una de ellas.

Después otra con las distintas tensiones que se generan en el divisor de tensión con cada una de las parejas anteriores.

Por último, otra tabla en la que se hace corresponder cada uno de estos voltajes con el valor de la conversión AD del Arduino con precisión de 10 bits (1024 -¿5V lo que Vout es a X)

| | 0R | 470R | 1000R | 2200R |
|---|---|---|---|---|

**Tabla 2.4.1.1 –** Valores de las resistencias para el teclado

| | | R1 Col1 | R2 Col2 | R3 Col3 | R4 Col4 |
|---|---|---|---|---|---|
| 220R | **R5 Fila1** | 220R | 690R | 1220R | 2420R |
| 330R | **R6 Fila2** | 330R | 800R | 1330R | 2530R |
| 470R | **R7 Fila3** | 470R | 940R | 1470R | 2670R |
| 560R | **R8 Fila4** | 560R | 1030R | 1560R | 2760R |
| | | **Vout** | **Vout** | **Vout** | **Vout** |
| 1200R | **R9 Pull-Down** | 4,225V | 3,175V | 2,479V | 1,657V |
| 5V | **Vin** | 3,922V | 3,000V | 2,372V | 1,609V |
| | | 3,593V | 2,804V | 2,247V | 1,550V |
| | | 3,409V | 2,691V | 2,174V | 1,515V |
| | | **CAD** | **CAD** | **CAD** | **CAD** |
| 1024 | CAD Pres. 10 bits | 865 | 650 | 508 | 339 |
| | | 803 | 614 | 486 | 329 |
| | | 736 | 574 | 460 | 318 |
| | | 698 | 551 | 445 | 310 |

| KEYS | | | |
|---|---|---|---|
| 1 | 865 | 1 | 62 |
| 4 | 803 | 2 | 67 |
| 7 | 736 | 3 | 38 |
| 0 | 698 | 4 | 48 |
| 2 | 650 | 5 | 36 |
| 5 | 614 | 6 | 40 |
| 8 | 574 | 7 | 23 |
| STOP | 551 | 8 | 43 |
| 3 | 508 | 9 | 22 |
| 6 | 486 | 10 | 25 |
| 9 | 460 | 11 | 15 |
| STOP | 445 | 12 | 106 |
| F1 | 339 | 13 | 10 |
| F2 | 329 | 14 | 12 |
| F3 | 318 | 15 | 7 |
| F4 | 310 | 16 | 310 |

**Tabla 2.4.1.1 –** Valores de las resistencias para el teclado

En la tabla anterior se muestran los valores que se van a obtener tras la conversión analógico- digital para cada tecla pulsada. Son valores razonablemente separados unos de otros y que pueden permitir leer un teclado con un único pin del Arduino (con AD) que es lo que se pretende realizar.

### 2.4.2. Control de matriz de pulsadores mediante modulo PCF8574

Este método es mas eficiente que el anterior por las razones comentadas en la sección 6.5 del documento Memoria por el simple hecho de que se activa una interrupción del modulo cada vez que cambia el estado de algún pin de entrada/salida de los que dispone. Ademas, se utilizarían muchas menos entradas y salidas ya que este modulo se comunica mediante el protocolo I2C,.

El funcionamiento de este modulo es el siguiente: Este modulo dispone de 8 salidas y 2 pines para la comunicación y otros 2 para alimentación. Una de las salidas de las que dispone (INT) va conectada a un pin de interrupción del microcontrolador. Los 7 pines restantes pueden funcionar como entradas o salidas.

El pin de interrupción dará un flanco de subida cuando una de las entradas cambie de 0 a 1 y un flanco de bajada cuando una de las entradas cambie de 1 a 0.

El conexionado para este ejemplo se realizara según la figura 2.4.2.1

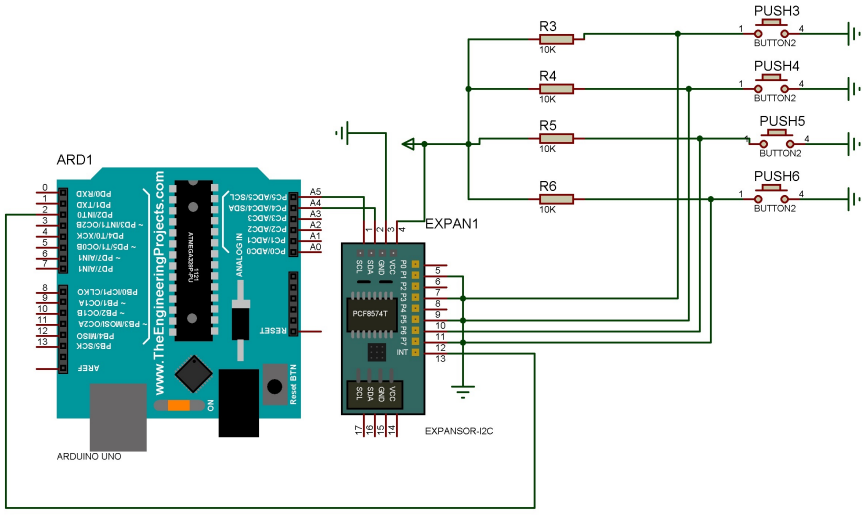



**Figura 2.4.2.1 –** Conexionado de pulsadores a Arduino UNO con el modulo PCF8574T

El siguiente programa imprime por el puerto serie la lectura del modulo I2C tras la pulsación de cualquiera de los botones.

```
1 /*
2 //**********************************************************************
3 // Autor : CRISTÓBAL GARCÍA CAMOIRA
```

```
//*****************************************************************
*/
#define DIRECCION_PCF8574AT 0x38 // Direccion del modulo
#define TIMEOUT_I2C 10 // 10 milisegundos de espera antes de renunciar a leer el bus
    I2C

#include <Wire.h>

const byte InterrupcionPin2=2;
byte lectura=0;
boolean lectura_pendiente=false;
long cronometro_timeout_i2c;
long tiempo_transcurrido;


void setup()
{
  Serial.begin(9600);
  Serial.println("PROBANDO TECLADO I2C");
  Wire.begin();
  pinMode(InterrupcionPin2,INPUT_PULLUP);
  attachInterrupt(0,recibir_PCF8574,CHANGE);
}

void loop()
{
  if(lectura_pendiente)
  {
    Wire.requestFrom(DIRECCION_PCF8574AT,1);
    cronometro_timeout_i2c=millis();
    do
    {
      tiempo_transcurrido=millis()-cronometro_timeout_i2c;
    }
    while(!Wire.available()&&tiempo_transcurrido<TIMEOUT_I2C);
    if(Wire.available())
    {
      lectura=Wire.read();
      Serial.println(lectura);
    }
    lectura_pendiente=false;
  }
}

void recibir_PCF8574()
{
  detachInterrupt(0);
  lectura_pendiente=true;
  attachInterrupt(0,recibir_PCF8574,CHANGE);
}
```

**Código 2.3:** Control de teclado mediante modulo PCF8574

## 2.5. Prueba de funcionamiento del GPS

Para la comprobación del funcionamiento del módulo GPS, es necesario poseer además de dicho módulo, un Arduino, preferiblemente Arduino UNO, MEGA o NANO.

El conexionado para esta prueba se muestra en la siguiente figura (2.5.0.1).

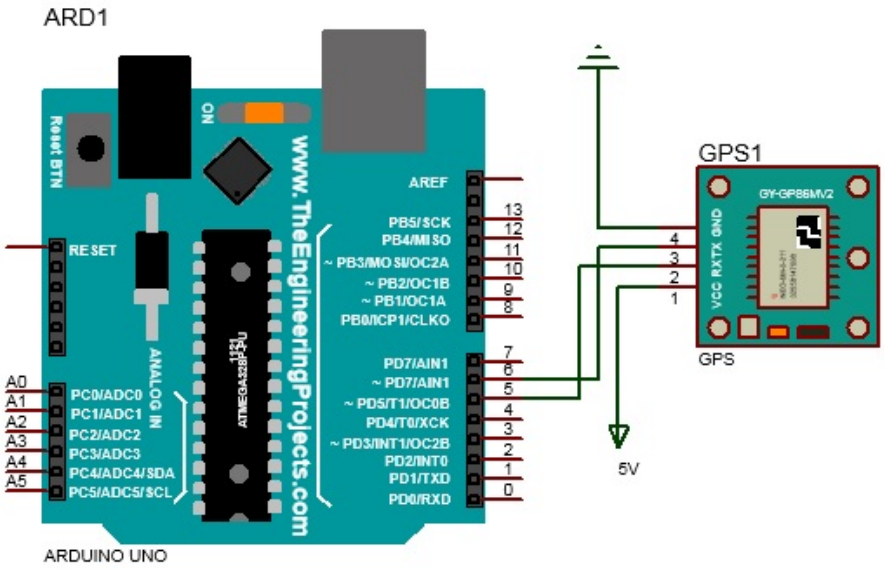



**Figura 2.5.0.1 –** Conexionado para la prueba del GPS Ublox NEO-6M

Con el siguiente programa, cargado desde la interfaz a nuestro Arduino se puede comprobar la recepción de las tramas del módulo GPS, ya sea visualizándolas con la terminal serie propia de la interfaz de Arduino, o bien mediante la utilización de un Hyperterminal, una aplicación de Windows que le permite establecer una comunicación ordenador a ordenador o a cualquier otro dispositivo a través de una conexión telefónica convencional o por el puerto serie.

```
1  //*****************************************************************
2  // GPS Neo 6-m Module
3  //*****************************************************************
4  // Autor : CRISTÓBAL GARCÍA CAMOIRA
5  //*****************************************************************
6  // # Descripción:
7  // # El presente programa sirve para probar el funcionamiento de
8  // # nuestro GPS con la placa de nuestro Arduino UNO.
9  //*****************************************************************
10 // Conexionado
```

```
11 //********************************************************************
12 // * u-blox NEO-6M - Arduino Uno
13 // * VCC - 5V
14 // * RX - 3
15 // * TX - 2
16 // * GND - GND
17
18 #include <Arduino.h>
19 #include <SoftwareSerial.h>
20
21 #define PC_BAUDRATE      9600
22 #define GPS_BAUDRATE     9600
23 #define GPS_RX           5
24 #define GPS_TX           6
25
26 SoftwareSerial gps_serial(GPS_TX, GPS_RX);
27
28 void setup(void)
29 {
30     Serial.begin(PC_BAUDRATE);
31     gps_serial.begin(GPS_BAUDRATE);
32 }
33 // Si hay algun dato del gps, se lee y se envia al PC o viceversa.
34
35 void loop(void)
36 {
37     if (gps_serial.available())
38     {
39         Serial.write(gps_serial.read());
40     }
41
42     if (Serial.available())
43     {
44         gps_serial.write(Serial.read());
45     }
46 }
47
48 //********************************************************************
```

**Código 2.4:** Lectura de datos GPS con el protocolo NMEA

## 2.6. Prueba de funcionamiento del modulo bluetooth

### 2.6.1. Cambio de configuración mediante comandos AT

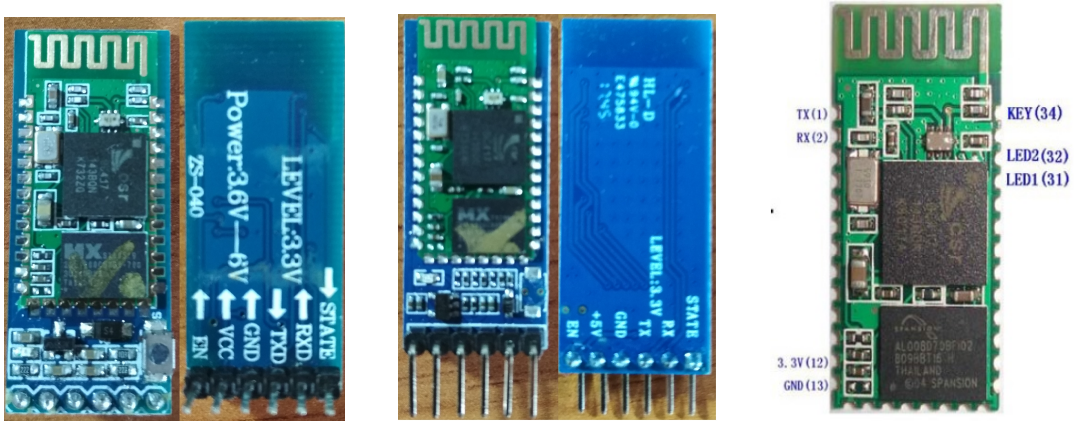
El modulo bluetooth que posee el presente proyecto, por defecto se comunica con un smartphone a una velocidad (baudrate). Pero en este modo, el modulo bluetooth solo es capaz de enviar y recibir información comunicándose con el dispositivo al cual este conectado,

es decir, si el modulo bluetooth esta conectado físicamente a un microcontrolador, ya sea de atmel, microchip, st..., y por otro lado se tiene el dispositivo móvil emparejado con el modulo bluetooth, ambos dispositivos (el smartphone y el microcontrolador) solo podrán intercambiar información entre ellos, lo que no se podría realizar en este modo es un cambio de las características del dispositivo bluetooth.

En el dispositivo de localización se desea que el nombre del modulo bluetooth (que es visible para poder conectarse a el desde nuestro smartphone) o bien la contraseña de emparejamiento, sean configurables y se puedan modificar cuando se desee.

Para ello se necesita realizar un cambio en el modo de funcionamiento del dispositivo bluetooth para que se puedan configurar los parámetros de los que se han hablado anteriormente.

Antes de continuar con los ejemplos, es necesario comentar que, existe una gran cantidad de placas que contienen este modulo y el diseño hardware va cambiando a medida que pasa el tiempo, es probable que si se adquiere un modulo de los que existen actualmente, los ejemplos que existen a continuación no funcionen, de hecho, para este proyecto se han comprado 2 módulos bluetooth, que en teoría se pensaba que eran idénticos, uno de ellos funcionaba correctamente con los programas de ejemplo que a continuación se explican y el otro no debido a esos cambios de hardware. Desgraciadamente durante las pruebas que se han realizado, una subida de tensión ha dejado el modulo inservible y se ha tenido que utilizar el otro perdiendo la funcionalidad de poder cambiar de forma automática el nombre del dispositivo y su contraseña.




(a) Topología del módulo que permite cambiar el modo de funcionamiento con el pin EN

(b) Topología del módulo utilizado en el presente proyecto

(c) Núcleo del módulo HC-05

**Figura 2.6.1.1 –** Diferentes desarrollos hardware para el mismo modulo bluetooth HC-05

La diferencia de ambos módulos era que en uno de ellos (figura 2.6.1.1(a)) disponía un divisor de tensión en la placa de circuito impreso que, aplicando una tensión de 5 voltios procedentes de una salida del microcontrolador a la entrada EN del modulo, se obtenían los 3,3 voltios necesarios para aplicarle al pin 34 del chip (figura 2.6.1.1(c)). Posteriormente se alimenta el modulo con los 5 voltios por la entrada denominada VCC y el modulo estaría configurado en modo comandos AT, esto lo podremos ver también físicamente dado que el

modulo bluetooth posee un led que, en este momento parpadeara con una frecuencia de 2Hz. La velocidad de transferencia de envío en modo comandos AT (baudrate)sera de 38400 bits por segundo según la hoja de datos técnicos del aparato.

En el otro modulo (figura 2.6.1.1(b)) (que es el que se ha utilizado en este proyecto, cuyo plano es el **[Plano 10]** que se encuentra en el documento de los planos y, para cambiar a modo comandos AT, es necesario mantener presionado el pulsador que dispone antes de alimentarlo. Una vez alimentado quedara configurado en modo comandos AT con un baudrate de 38400 bits por segundo y la frecuencia de parpadeo del led sera de 2HZ.

Para la comprobación del funcionamiento del módulo Bluetooth, es necesario poseer además de dicho módulo, un Arduino, preferiblemente Arduino UNO o MEGA. El conexionado de dicho módulo con la placa Arduino UNO se muestra en la siguiente figura (2.6.1.2) y que servirá para la realización del resto de pruebas con este modulo, teniendo en cuenta que el modulo del que se dispone para estos ejemplos es el que se muestra en la figura 2.6.1.1(a), es decir, el que posee la funcionalidad de cambiar el modo de funcionamiento del bluetooth mediante software.

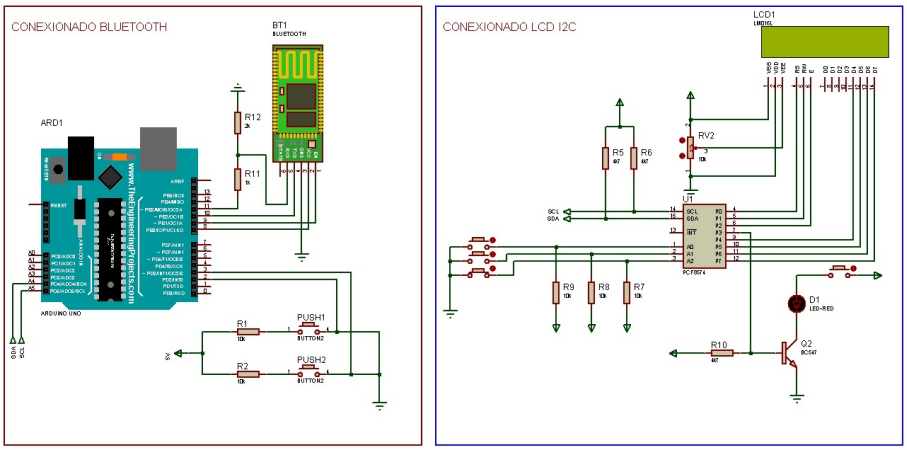



**Figura 2.6.1.2 –** Conexionado para la prueba de funcionamiento del modulo Bluetooth HC-05

La alimentación de este modelo de Bluetooth (HC-05) puede realizarse directamente a un pin de salida de Arduino, dado que su consumo es muy pequeño.

Puede darse el caso, según las especificaciones de consumo, de que el modulo se nos resetee debido a que durante el emparejamiento con cualquier dispositivo puede llegar a consumir cerca de 40mA, que es el limite de lo que nos proporciona una salida de nuestro microcontrolador. Esto se puede resolver implementando un circuito con transistores (figura2.6.1.3) que sea capaz de subministrar la potencia necesaria al modulo bluetooth.

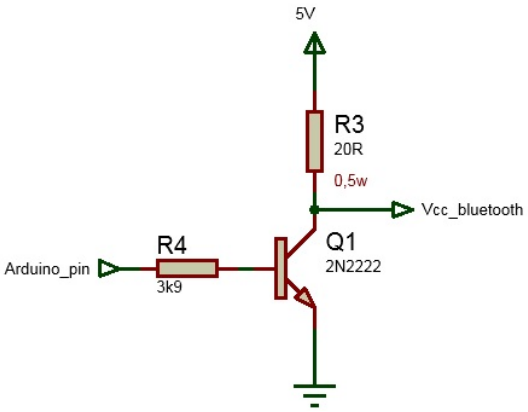



**Figura 2.6.1.3 –** Circuito con transistores que se puede implementar para el control de encendido del modulo bluetooth con Arduino

De todas formas este tipo de problema no ha ocurrido durante las pruebas, motivo por el cual en el **[Plano 1]** del documento de los planos se ha conectado directamente la alimentación del modulo al pin numero 5 de Arduino MEGA.

Con el siguiente programa, cargado desde la interfaz propia de Arduino a Arduino MEGA, se puede cambiar la configuración del modulo Bluetooth para que reciba comandos AT (si se utiliza un modulo bluetooth cuya topología sea idéntica a la que se muestra en la figura 2.6.1.1(a).

```
1  //*********************************************************************
2  //  Autor : CRISTÓBAL GARCÍA CAMOIRA
3  //*********************************************************************
4  /*
5  //*********************************************************************
6  El presente programa es capaz de enviar los comandos AT a nuestro bluetooth
7  con ellos podremos cambiar el ID del dispositivo, la velocidad de transmision
8  configurarlo en modo maestro o esclavo (esto solo se puede realizar con el
9  modulo HC-05.
10 Es necesario comprobar que la velocidad a la que transmite el arduino y la que
11 esta configurada en el bluetooth es la misma, si no, no se establecera una
12 comunicacion correcta o bien los caracteres recibidos seran irreconocibles.
13
14 NOTA: Acordarse de tener marcada la opcion de CR-LF en la esquina infreior
15 derecha del monitor serial del programa Arduino para poder ver la respuesta
16 de nuestro modulo bluetooth.
17 //*********************************************************************
18 */
19
20 #include <SoftwareSerial.h>
21
22 SoftwareSerial BT1(10, 11); // RX | TX
23 void setup()
24    { pinMode(8, OUTPUT);          // Al poner en HIGH forzaremos el modo AT
25      pinMode(9, OUTPUT);          // cuando se alimente de aqui
```

```
26      digitalWrite(9, HIGH);
27      delay (500) ;                // Espera antes de encender el modulo
28      Serial.begin(9600);
29      Serial.println("Modo comandos del moduo HC-05");
30      digitalWrite (8, HIGH);      //Enciende el modulo
31      Serial.println("Esperando comandos AT:");
32      BT1.begin(38400);
33    }
34
35  void loop()
36    {
37
38      if (BT1.available())
39              Serial.write(BT1.read());
40       if (Serial.available())
41              BT1.write(Serial.read());
42    }
```

**Código 2.5:** Cambio de la configuración del modulo Bluetooth mediante comandos AT

### 2.6.2. Cambio del modo de funcionamiento (modo comandos y modo conexión)

Como se ha venido advirtiendo a lo largo de este apartado, para que este ejemplo funcione, es necesario poseer un modulo bluetooth cuya topología sea idéntica a la que se muestra en la figura 2.6.1.1(a).

Con el programa que a continuación se describe, comprobado con Arduino UNO, se pretende cambiar el modo de funcionamiento del bluetooth presionando 2 pulsadores, se recurda que el modulo Bluetooth posee 2 modos básicos de funcionamiento:

1. Modo Comandos: En el cual el modulo sera indetectable por cualquier otro dispositivo bluetooth y de esta forma se pueden enviar los comandos AT desde un ordenador cualquiera para cambiar aspectos como la velocidad de transferencia, el nombre del dispositivo o el pin, aspectos que ya se han comentado con anterioridad.

2. Modo de funcionamiento normal: Cualquier otro dispositivo (un smartphone por ejemplo), se puede conectar al dispositivo Bluetooth y enviar y recibir información a través del mismo.

Presionando uno de los pulsadores el dispositivo Bluetooth se volverá indetectable por el resto de dispositivos bluetooth y será en ese momento cuando sea capaz de aceptar comandos AT que procederán a cambiar la configuración del dispositivo.

Presionando el otro pulsador, se sale del modo anterior volviéndose detectable por el resto de dispositivos bluetooth pudiendo establecer una comunicación satisfactoria.

El montaje de esta prueba se puede realizar teniendo en cuenta el circuito que se muestra en la figura (2.6.1.2).

El siguiente programa cambia el modo de funcionamiento del bluetooth presionando 2 pulsadores asociados a interrupciones externas del microcontrolador, para ello se ejecuta una rutina de inicialización del bluetooth diferente que sucederá tras presionar cada botón, de esta forma se puede acceder a la configuración del modulo bluetooth mediante software.

```
//*********************************************************************
//  Autor : CRISTÓBAL GARCÍA CAMOIRA
//*********************************************************************
/*
//*********************************************************************
El presente programa controla el estado del bluetooth mediante 2 botones
utilizando las interrupciones 0 y 1 del Arduino.
Presionando el boton designado como pulsador1 asociando al pin2 del
Arduino el modulo Buletooth entraria en modo comandos AT donde podriamos
variar el nombre de dispositivo bluetooth, la velocidad de transmision y
recepcion, consutar la verson de su firmware la contraseña que tiene asociada
y muchos otros comandos.El bluetooth permanecera en modo comandos hasta que
presionemos el otro pulsador, y mientras tanto el dispositivo bluetooth no
podra ser detectado por otro tipo de dispositivos, smatphones, ordenadores...

Presionando el boton designado como pulsador2 asociando al pin3 del Arduino el
modulo Buletooth sale del modo comandos, es entonces cuando nos podremos conectar
a el a traves de nuestro smatrphone, hay que tener en cuenta que nuestro
smartphone nos pedira por defecto una contraseña, nuestro dispositivo bluetooth
es el modelo HC-05 y por defecto sus contraseñas suelen ser "0000" o "1234.
Hay que tener cuidado y no olvidar la contraseña en el caso de haberla cambiado
dado que no nos podremos volver a conectar al dispositivo, en ese caso se
recomienda entrar en modo comandos presionando el pulsador asociado al pin 2 del
Arduino e introducir el comando AT+RESET.

NOTA: Acordarse de tener marcada la opcion de CR-LF en la esquina infreior
derecha del monitor serial del programa Arduino para poder ver la respuesta
de nuestro modulo bluetooth.
//*********************************************************************
*/

#include <SoftwareSerial.h>
 const byte Bt_alimentacion=8;
 const byte Btpin_Atcomando=9;
 const byte Pulsador1=2;
 const byte Pulsador2=3;

 #define Encender_Bluetooth()    digitalWrite (8, HIGH)
 #define Apagar_Bluetooth()      digitalWrite (8, LOW)

 #define Habilitar_Bt_AtCmd()        digitalWrite (9, HIGH)
 #define Deshabilitar_Bt_AtCmd()     digitalWrite (9, LOW)

SoftwareSerial BT1(10, 11); // RX | TX
```

```
45
46 boolean C_set = false;
47 volatile int PosBoton2=0;                    // Contador para el dial.
48 static boolean rotating=false;               // Eliminacion de rebotes.
49 volatile unsigned int lastReportedPos = 0; // Gestion para el cambio.
50 volatile unsigned int PosBoton1 = 0;         // Contador para el pulsador.
51 volatile unsigned int lastPosicion = 0;     // Gestion para el cambio.
52
53 void setup()
54    {
55      pinMode(Pulsador1, INPUT_PULLUP); // Nuevo metodo para habilitar las
    resistencias de pullup.
56      attachInterrupt( 0, ServicioBoton1, CHANGE);// Pin 2 interripcion 0
57      pinMode(Pulsador2, INPUT_PULLUP); // Nuevo metodo para habilitar las
    resistencias de pullup.
58      attachInterrupt( 1, ServicioBoton2, CHANGE);// pin3 interrucion 1
59      pinMode( Bt_alimentacion, OUTPUT);
60      pinMode( Btpin_Atcomando, OUTPUT);
61      Serial.begin(9600);
62      BT1.begin(38400);
63      Encender_Bluetooth();      //Enciende el modulo
64    }
65
66 void loop()
67 {
68       rotating = true;  // Reestablecer el circuito antirrebote
69      if (BT1.available())
70            Serial.write(BT1.read());
71       if (Serial.available())
72            BT1.write(Serial.read());
73
74      if (lastPosicion != PosBoton1)
75      {
76      lastPosicion  = PosBoton1;
77      Bluetooth_At();// Modo comandos del bluetooth.
78      Serial.println(PosBoton1,DEC);
79      }
80
81      if (lastReportedPos != PosBoton2)
82      {
83      lastReportedPos = PosBoton2;
84      Bluetooth_Reset();// Modo de deteccion del bluetooth.
85      Serial.println(PosBoton2,DEC);
86      }
87    }
88
89 void Bluetooth_At()
90 {
91   Apagar_Bluetooth();// Apaga el modulo
92   delay (500) ;
```

```
93    Habilitar_Bt_AtCmd();// Pin activo para entrar en modo comandos.
94    delay (500) ;                // Espera antes de encender el modulo
95    Serial.println("Levantando el modulo HC-05");
96    Encender_Bluetooth();     //Enciende el modulo
97    Serial.println("Esperando comandos AT:");
98  }
99
100 void Bluetooth_Reset()
101 {
102   Serial.println("Reseteando el modulo HC-05");
103   Apagar_Bluetooth();// Apaga el modulo
104   delay (500) ;                // Espera antes de encender el modulo.
105   Deshabilitar_Bt_AtCmd();// Pin desactivado para entrar en modo comandos.
106   Encender_Bluetooth();     //Enciende el modulo
107 }
108
109 void ServicioBoton1()
110    {
111       if ( rotating ) {delay (20);}
112       if( digitalRead(Pulsador1)!= C_set   )
113       {
114       C_set = !C_set;
115       if( C_set   )
116       {
117       PosBoton1 += 1;
118       rotating = false;
119       }
120       }
121    }
122 void ServicioBoton2()
123    {
124       if ( rotating ) {delay (20);}
125       if( digitalRead(Pulsador2)!= C_set   )
126       {
127       C_set = !C_set;
128       if( C_set   )
129       {
130       PosBoton2 += 1;
131       rotating = false;
132       }
133       }
134    }
```

**Código 2.6:** Cambio del modo de funcionamiento del modulo bluetooth

### 2.6.3. Captura de datos recibidos a través de Bluetooth

El presente programa recibe por Bluetooth una trama de datos que sera almacenado en un string y lo visualiza en un lcd de 16x2 lineas y también lo imprime por el puerto serie. Esta

parte es muy útil, puesto que puede ser utilizada para que Arduino realice acciones en función de los datos recibidos por Bluetooth (controlar Arduino mediante Bluetooth).

El montaje de esta prueba se puede realizar teniendo en cuenta el circuito que se muestra en la figura (2.6.1.2).

En el caso de tener un módulo cuya topología sea similar a la que se muestra en la figura 2.6.1.1(b), se recomienda no conectar la entrada EN del módulo.

```
1  //************************************************************************
2  // Autor : CRISTÓBAL GARCÍA CAMOIRA
3  //************************************************************************
4  /*
5  El presente programa recibe por bluetooth una trama de datos que sera
6  almacenado en un string y lo visualiza en un lcd de 16x2 lineas y tambien
7  lo imprime por el puerto serie.
8
9  NOTA: Acordarse de tener marcada la opcion de CR-LF en la esquina infreior
10 derecha del monitor serial del programa Arduino para poder ver la respuesta
11 de nuestro modulo bluetooth.
12 */
13
14 //Incluimos librerias
15  #include <Wire.h>                 // La necesita la libreria LiquidCrystal_I2C.h
16  #include <LiquidCrystal_I2C.h>    // Para usar un display I2C
17  #include <SoftwareSerial.h>
18  LiquidCrystal_I2C lcd(0x27,16,2); // Establece la dirección de memoria 0x27 para un
        display de 20 caracteres y 4 líneas.
19
20  const byte Bt_alimentacion=8;
21  const byte Btpin_Atcomando=9;
22
23  #define Encender_Bluetooth()    digitalWrite (8, HIGH)
24  #define Apagar_Bluetooth()      digitalWrite (8, LOW)
25
26  #define Habilitar_Bt_AtCmd()       digitalWrite (9, HIGH)
27  #define Deshabilitar_Bt_AtCmd()    digitalWrite (9, LOW)
28
29  SoftwareSerial BT1(10, 11); // RX | TX
30
31 void setup()
32    {
33      lcd.init();                      // Inicializo el LCD.
34      lcd.backlight();                 // Brillo de la pantalla encendido.
35      pinMode( Bt_alimentacion, OUTPUT);
36      pinMode( Btpin_Atcomando, OUTPUT);
37      lcd.setCursor(0,2);
38      lcd.print("---BLUETOOTH---");
39      Serial.begin(9600);
40      BT1.begin(9600);
```

```
41      Encender_Bluetooth();      //Enciende el modulo
42    }
43
44  void loop()
45    {
46      int i=0;
47      char commandbuffer[100];
48
49      if(BT1.available())
50      {
51      delay(100);
52      while( BT1.available() && i< 99)
53        {
54        commandbuffer[i++] = BT1.read();
55        }
56      commandbuffer[i++]='\0';
57      }
58
59      if(i>0)
60
61      Serial.println((char*)commandbuffer);
62      lcd.print((char*)commandbuffer);
63      delay(1000);
64      lcd.clear();
65    }
```

**Código 2.7:** Captura de datos recibidos a través de Bluetooth

## 2.7. Enviar SMS por comandos AT

Antes de comenzar esta prueba, es necesario recordar que la tarjeta SIM es necesario introducirla antes de encender el modulo SIM800L, de no ser así, el modulo no podrá conectarse a la red con éxito y por lo tanto no funcionara.

Lo mas critico de este modulo es la alimentación, para ello se debe revisar el apartado (4.5).

Cualquier variación en la tensión de alimentación del modulo repercute en su funcionamiento, ademas aunque el modulo en si es diminuto, durante el envío de un SMS o una conexión a la red GSM el modulo posee picos de consumo de hasta 2 amperios.

En caso de producirse un reset del modulo al introducirle el comando del Pin, o bien durante el envío de un SMS, realización de una llamada o conexión a la red GPRS, las causas pueden ser:

1. La tarjeta Sim No se ha introducido.

2. La tarjeta Sim se ha introducido después de haber encendido el modulo.

3. Alimentación insuficiente. Este caso es bastante común para la gente que no esta acostumbrada a este tipo de sistemas. Durante la conexión a la red se producen picos de corriente de hasta 2A en este tipo de módulos, ocurre lo mismo durante la ejecución de los comandos para realizar una llamada o el envío de un SMS, por ello es necesario disponer de una alimentación adecuada (con una fuente de alimentación de 5V y 2 amperios es mas que suficiente).

4. Incoherencia entre las masas del sistema. El sistema que envía los datos y el modulo deben tener el mismo punto de referencia (masa), es decir, la masa del modulo SIM800L debe ser la misma que la del sistema que le envíe por serie los datos (ya sea un Arduino, un ordenador ...)

El programa a continuación descrito solo sirve para configurar los pines de control del módulo GPRS/GSM SIM800L con Arduino MEGA o UNO, se recomienda seguir los pasos que se indican en el mismo para la correcta utilización del módulo.

El montaje para la realización de este apartado se muestra en la figura (2.7.0.1)

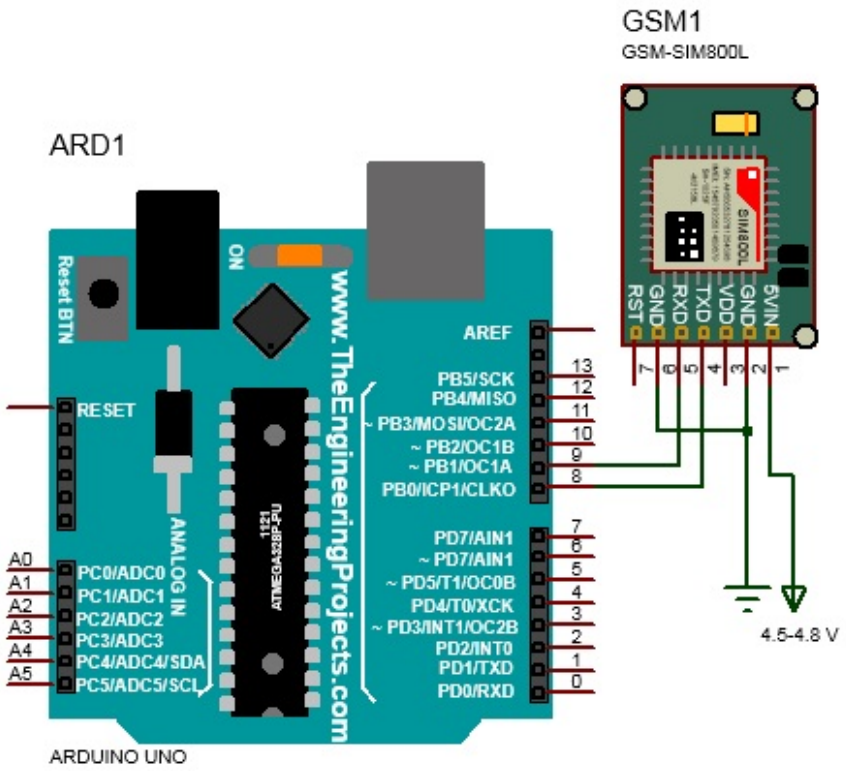



**Figura 2.7.0.1 –** Conexionado para la prueba de funcionamiento del modulo SIM800L

Este programa ha sido probado tanto en Arduino UNO como en Arduino MEGA.

```
/*
//************************************************************************
//GPS/GPRS/SIM 800L MODULE V2.0
//************************************************************************
```

```
// Autor:  CRISTOBAL GARCIA CAMOIRA
//*****************************************************************
// Descripcion:
// El presente programa sirve para probar el funcionamiento de
// nuestro Modulo GPS GPRS GSM V3.0 mediante el envio de comandos AT.
// Pasos:
// # 1.Montar el circuito presente en el plano XXXX.
// # 2.Cargar el presente codigo en Arduino UNO o MEGA.
// # 3.Seguir los pasos que se indican a continuacion.
// # 4.Comprobar que hemos recibido el SMS en nuestro terminal
*/

#include <SoftwareSerial.h>

SoftwareSerial SIM800L(8,9); // RX | TX
void setup()
   {
     Serial.begin(9600);
     SIM800L.begin(9600);
   }

void loop()
   {  if (SIM800L.available())
            Serial.write(SIM800L.read());
      if (Serial.available())
            SIM800L.write(Serial.read());
   }
```

**Código 2.8:** Enviar SMS mediante comandos AT con Arduino y modulo SIM800L

Una vez que se ha subido el código a la placa Arduino, se recomienda la descarga del programa coolterm del siguiente enlace (http://freeware.the-meiers.org/).

El programa citado anteriormente es un terminal serie que funciona de manera muy similar al Hyperterminal, no se ha utilizado Hyperterminal, ni el monitor serial de Arduino, porque no poseen la opción de enviar caracteres hexadecimales. En este caso concreto se necesita enviar el terminador de SMS (“1A”) en hexadecimal.

Pasos para la configuración del terminal Coolterm:

1. Se ejecuta Coolterm y se selecciona “Options”, tal y como se muestra en la figura 2.7.0.2.

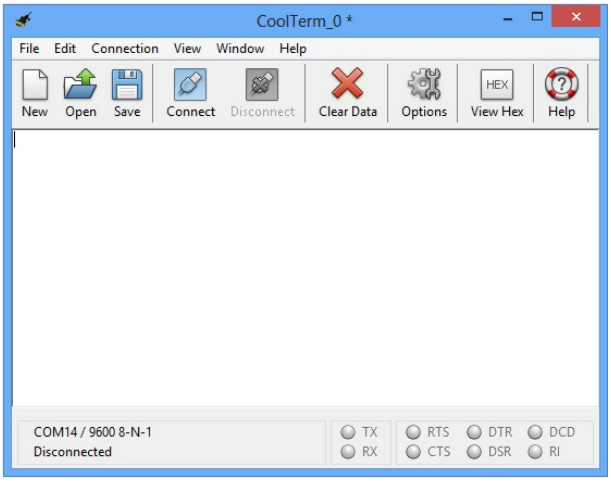



**Figura 2.7.0.2 –** Vista del terminal Coolterm

2. Se selecciona el puerto serie por el que se ha conectado Arduino. Si no se conoce la ubicación del puerto serie en el cual se ha conectado nuestro Arduino, se puede verificar haciendo click derecho en mi pc o computer, administrar, administrador de dispositivos, puertos COM Y LPT.

3. Se debe asegurar de que la velocidad de transmisión se establece en 9600. El número de bits de los datos es 8. Se ajusta paridad a “ninguno”. Se ajusta el número de bits de parada en 1.Este tercer paso se observa en la imagen que muestra la figura 2.7.0.3.

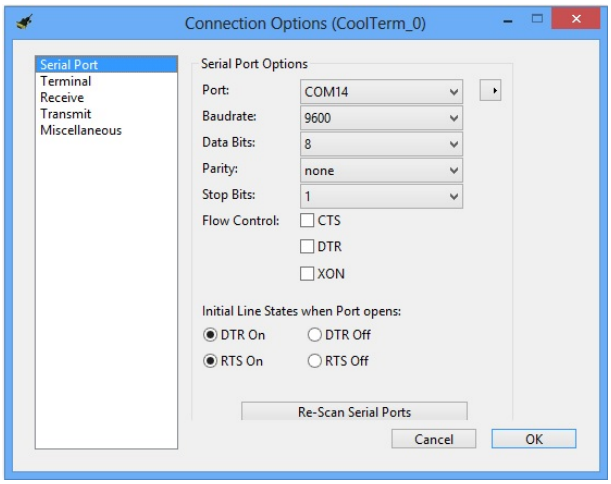



**Figura 2.7.0.3 –** Configuración del terminal Coolterm 1

4. En la lista de la parte izquierda de la ventana de opciones, haga click en “Terminal”. Asegúrese de que la opción “Local Echo” está activada. Esto le permitirá ver lo que esta escribiendo en el terminal. Esto se muestra en la figura 2.7.0.4.

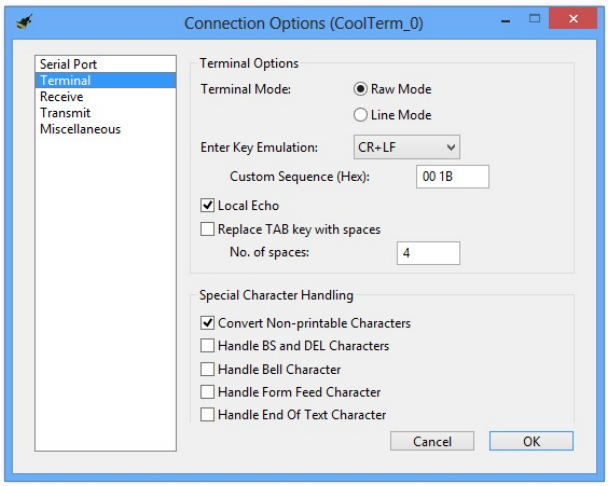



**Figura 2.7.0.4 –** Configuración del terminal Coolterm 2

5. Se presiona “OK” para grabar la configuración y se cierra la ventana de Opciones.

6. Se presiona “Connect ”en la barra de menús.

7. En este momento se puede verificar el estado “Connected ” en la barra de estado.

A continuación se deben enviar los siguientes comandos por orden. Las funciones de dichos comandos han sido explicadas en la sección 10.2 del documento MEMORIA. Dichos comandos se detallan a continuación:

```
AT
OK
AT+CPIN=XXXX
OK
AT+CREG?
+CREG: 0,1
AT+COPS?
+COPS: 0,0,"Orange"
OK
AT+CMGF=1
OK
AT+CMGS="6XXXXXXXX"
> Enviando SMS comandos AT...
> 0x1A(hex)
```

**Código 2.9:** Envío de comandos AT a través de la terminal

El carácter hexadecimal “0x1A” (el último comando enviado al módulo GPRS-GSM), indica el fin del mensaje. Inmediatamente después, el módulo procederá a su envío, y transcurridos unos segundos el SMS enviado llegará al smartphone, con el texto “Enviando SMS comandos AT...”.

Todavía no se ha explicado la forma de enviar este carácter hexadecimal desde Coolterm, es muy sencillo, solo se deben seguir los siguientes pasos:

1. Se presiona en la pestaña “Connection” situada en la barra superior y se selecciona la opción “Send string”, esta opción abrirá una ventana donde se muestran 2 opciones.
2. Se selecciona la opción Hex, entonces se escribe el texto hexadecimal que deseamos enviar, en este caso “1A” y presionamos el botón “Send”
3. El entorno de envío del carácter hexadecimal se muestra en la figura2.7.0.5.

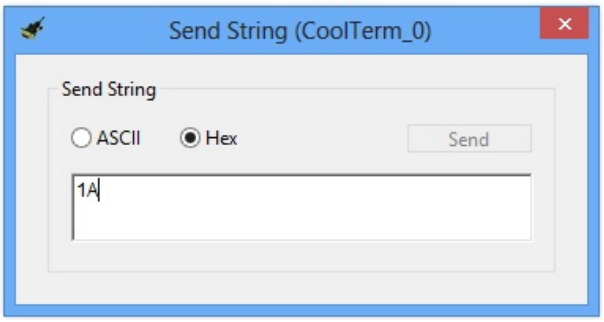



**Figura 2.7.0.5 –** Envío de un carácter hexadecimal utilizando el terminal Coolterm

Realizando los pasos previos, se debería ver reflejado en el smartphone, el SMS que se ha escrito en el código que se ha presentado anteriormente. Si no se recibe el SMS en el Smartphone vuelva a repetir los pasos previos, verifique el montaje, asegúrese de que la tarjeta SIM esta bien colocada y que funciona correctamente en un smartphone.

Ahora se vuelve a realizar prueba del envío de un SMS pero esta vez, a través de Arduino, sin necesidad de teclear manualmente los comandos AT.

## 2.8. Enviar SMS mediante Arduino

Se debe introducir en el programa el pin de la tarjeta SIM que se posea para que todo funcione correctamente. En el caso de que hubiese algún error en la introducción del PIN, tras tres errores obtenidos tras la introducción de un PIN incorrecto, la tarjeta se bloqueará.

**NOTA IMPORTANTE:** Para desbloquearla será necesario extraerla del módulo e introducirla en un smartphone, el smartphone mostrará en la pantalla que la tarjeta SIM está bloqueada y pedirá la introducción del código PUK (se poseen 10 intentos antes de que se bloquee la tarjeta definitivamente), se introduce el código PUK, y luego se vuelve a introducir el código PIN, ya sea uno nuevo o el anterior, y la tarjeta SIM estará desbloqueada.

El presente programa envía el mensaje “Hola soy Arduino” almacenado en el array “mensaje” al número de teléfono almacenado en el array “numero_Tlf” con el pin de la tarjeta SIM

almacenado en el array “pin”. Los comandos AT enviados al módulo se encuentran en la función “establecer_numero_tlf()”.

El montaje para la realización de este apartado es el mismo que se indica en la figura (2.7.0.1).

```
//*********************************************************************
// GPRS/GSM Module SIM800L
//*********************************************************************
// Autor : CRISTÓBAL GARCÍA CAMOIRA
//*********************************************************************
//  Descripción:
//  El presente programa envía el mensaje "Hola soy Arduino"
//  almacenado en el array mensaje al número de teléfono almacenado
//  en el array numero_Tlf. Los comandos AT enviados al módulo se
//  encuentran en la función establecer_numero_tlf().


char mensaje[35] ="Hola soy Arduino";
char numero_Tlf[12]="XXXXXXXXX";
char pin[5]={"XXXX"};
char numero_buf[25];

void setup()
{
  Serial.begin(9600); // Establece la velocidad de transmisión del p.serie.
}

void loop()
{
  establecer_numero_tlf (numero_Tlf, pin);
  envia_mensaje_gsm (mensaje);
  finaliza_envio_gsm ();
  while(1);
}

// Establece el número de teléfono y el pin.
void establecer_numero_tlf (char *numero, char *pin )
{
  Serial.println ("AT");
  delay (2000);
  Serial.println ("AT");
  delay (2000);
  sprintf (numero_buf, "AT+CPIN=%s", pin);
  Serial.println (numero_buf);
  delay(2000);
  Serial.println("AT+COPS?");
  delay(2000);
  Serial.println ("AT+CMGF=1");
  delay (1000);
```

```
45    sprintf (numero_buf, "AT+CMGS=\"%s\"", numero);
46    Serial.println (numero_buf);
47    delay (1000);
48  }
49  // Envía el mensaje al teléfono móvil.
50  void envia_mensaje_gsm (char *mensaje)
51  {
52    Serial.println (mensaje);
53  }
54  // Envia el comando hexadecimal.
55  void finaliza_envio_gsm()
56  {
57    delay (1000);
58    Serial.write (26);
59    delay (2000);
60  }
61  //*********************************************************************
```

**Código 2.10:** Enviar SMS desde Arduino directamente con modulo SIM800L

## 2.9. Enviar un email mediante comandos AT

Enviar un e-mail con un modulo del fabricante SIMCOM (familias sim900-sim800) es una tarea un tanto compleja si no se tienen en cuenta los siguientes detalles que son necesarios para un envío de un e-mail con éxito.

Existen gran cantidad de tutoriales que muestran la forma de enviar los comandos AT necesarios al modulo GSM/GPRS para conseguir el envío de un e-mail, incluyendo códigos de ejemplo para Arduino, pero hay que tener en cuenta que existen protocolos de seguridad que encriptan los mensajes durante el proceso de envío-recepción, alguno de estos protocolos son: ssl, pop e imap ...

El modulo que se utiliza en el presente proyecto solo soporta el protocolo SSL, por lo tanto se pueden utilizar los servidores gratuitos SMTP de Yahoo, Gmail o Hotmail a través del citado protocolo de seguridad SSL. El protocolo de seguridad realiza una encriptacion del mensaje antes de entrar en la bandeja de salida de correo, pues durante el proceso de envío de un e-mail, se comprueba que el contenido del mensaje esta encriptado, de no estarlo, el servidor no enviara ese mensaje y dará un aviso de error que llegara en forma de correo electrónico.

Existen dos formas de enviar un correo electrónico, la primera que se va a explicar es la de enviar el correo como texto plano (sin encriptar).Es un poco mas complicada de realizar si no tenemos una cuenta de correo electrónico creada con un dominio propio, y ademas los mensajes que recibamos entraran como spam. La segunda forma es la mas adecuada y correcta, siempre que nuestro modulo soporte encriptacion SSL.

1. Envío de correo electrónico como texto plano.

- Se ha creado una cuenta de correo nueva (en nuestro caso de yahoo), que se utilizara como remitente del correo electrónico.

- Se ha accedido a la web https://www.smtp2go.com y se ha creado una nueva cuenta gratuita que permite enviar hasta 1000 emails/mes, mas que suficiente para el uso que se le quiere dar a este servicio. Al registrarse en este portal web lo que se esta haciendo es realizando una ”pasarela virtual”, algo como una especie de bypass, mediante el cual los emails pueden ser enviados sin ser encriptados.Para el registro en este portal se ha utilizado la cuenta de Yahoo previamente creada, de esta forma se podra enviar texto plano con esa cuenta utilizando los servidores y los puertos que nos ofrece el portal smtp2go.

Después de haber realizado lo anteriormente citado, si se sube el programa que se indica en (2.9) a Arduino y se introducen los comandos AT necesarios para el envío de un e-mail (2.9) utilizando la terminal serie Coolterm, es muy probable que se obtenga una respuesta satisfactoria del modulo ”$+SMTPSEND:1'$, no obstante, a la hora de revisar el correo y el servidor snmtp2go, ocurrirá lo que se indica en las figuras (2.9.0.1) y (2.9.0.2).

Estos errores se presentan al enviar el e-mail en texto plano a cuentas de correo que tienen como destinatarios cuentas de correo de Yahoo, Gmail, Hotmail. El servidor interpreta que el e-mail ha violado todos los protocolos de seguridad, de esta forma ya ni siquiera lo acepta como correo malicioso (SPAM).

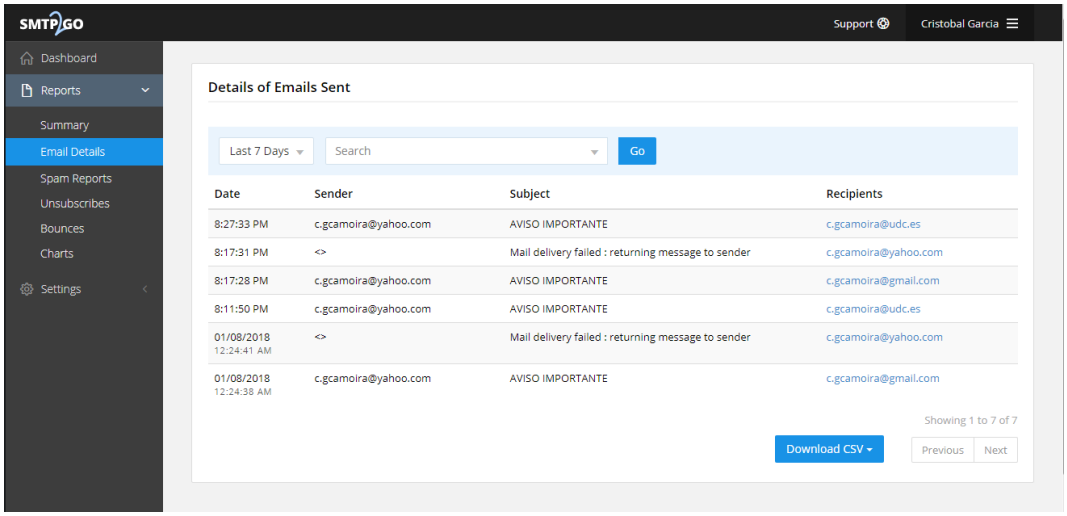



**Figura 2.9.0.1 –** Error percibido en el servidor SMTP2GO tras el envío de un e-mail a una dirección de Gmail

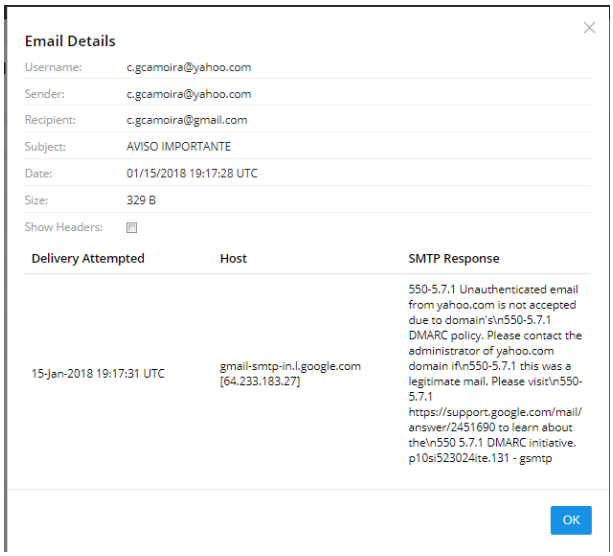



**Figura 2.9.0.2 –** Error detallado tras el envío de un e-mail utilizando el servidor SMTP2GO

Existen servidores de correo electrónico que no poseen tanta seguridad puesto que son de código abierto, el usuario seria el encargado de implementar sus propios protocolos de seguridad, añadir sus propios certificados etc. Un ejemplo de estos servidores son los que se utilizan en `https://mail.zimbra.com`.

En la figura 2.9.0.3 el mensaje se ha enviado de forma exitosa a una dirección de correo electrónico existente en la plataforma citada anteriormente, el problema es que se ha recibido como correo malicioso, que ocurra este caso no es deseable. Debido a este motivo, es necesario enviar el correo electrónico con encriptación, y se aplicara el protocolo SSL.

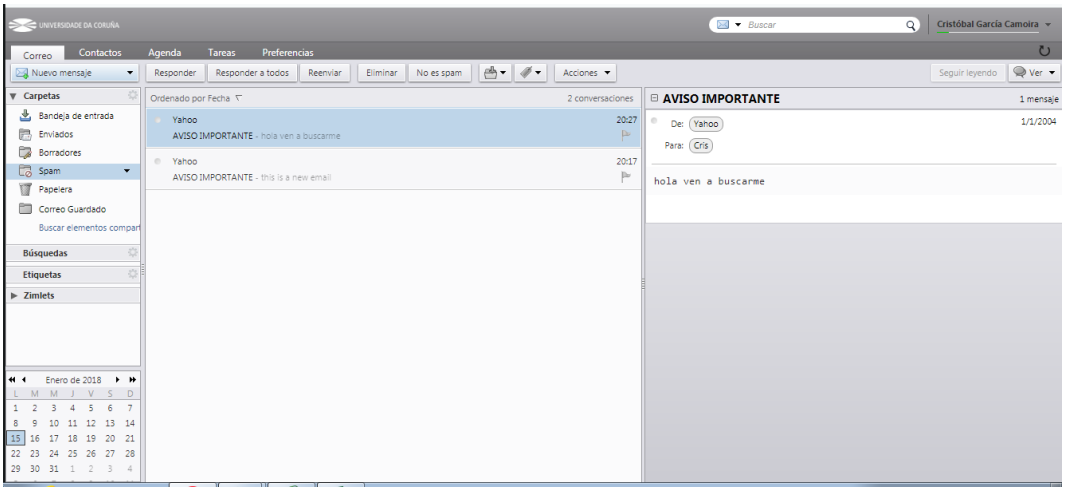



**Figura 2.9.0.3 –** E-mail recibido como Spam en una cuenta de correo perteneciente a zimbra

2. Envío de correo electrónico con encriptación SSL.

   Este es el método mas sencillo y mas seguro que se puede realizar con el modulo SIM800L, utilizando la encriptación (SSL). Este método sirve para realizar el envío de

correos electrónicos desde cuentas de Gmail, Yahoo, Hotmail. Los pasos se muestran a continuación:

- Se realiza el montaje que se indica en la figura (2.7.0.1).
- Se sube el programa (2.9) a la placa Arduino UNO, o MEGA, o NANO.
- Se ejecuta consola serie Coolterm, cuya configuración ya ha sido mencionada en los apartados anteriores.
- Se ejecutan los comandos que se indican en el apartado (2.9) y al cabo de unos segundos se recibirán en la bandeja de entrada del servidor de correo electrónico destinatario el correo que se acaba de enviar, tal y como se muestra en la siguiente imagen (2.9.0.4).

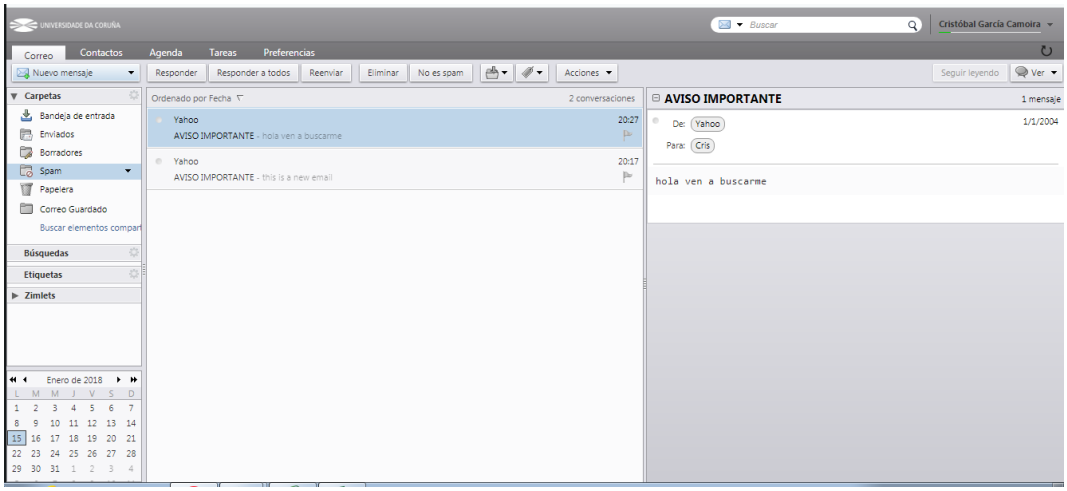



**Figura 2.9.0.4 –** E-mail recibido en la cuenta de correo de Gmail

> **NOTA:** En el caso de utilizar Gmail se utilizara el servidor smtp.gmail.com cuyo puerto es el 465.

Una vez que se ve que el ultimo comando recibido por el modulo tras esta secuencia es $+SMTPSEND:1$ el e-mail se habrá enviado el e-mail con éxito.

Este programa ha sido probado en Arduino UNO, Arduino MEGA y en Arduino NANO.

```
/*
//************************************************************************
//GPS/GPRS/SIM 800L MODULE V2.0
//************************************************************************
// Autor:  CRISTOBAL GARCIA CAMOIRA
//************************************************************************
// Descripcion:
// El presente programa sirve para probar el funcionamiento de
// nuestro Modulo GPS GPRS GSM V3.0 mediante el envio de comandos AT.
```

```
// Pasos:
// # 1.Montar el circuito presente en el plano XXXX.
// # 2.Cargar el presente codigo en Arduino UNO o MEGA.
// # 3.Seguir los pasos que se indican a continuacion.
// # 4.Comprobar que hemos recibido el SMS en nuestro terminal
*/

#include <SoftwareSerial.h>

SoftwareSerial SIM800L(8,9); // RX | TX
void setup()
    {
      Serial.begin(9600);
      SIM800L.begin(9600);
    }

void loop()
    {   if (SIM800L.available())
              Serial.write(SIM800L.read());
        if (Serial.available())
              SIM800L.write(Serial.read());
    }
```

**Código 2.11:** Enviar Emails mediante comandos AT con Arduino y modulo SIM800L

Luego enviaremos uno a uno los comandos que siguen a continuación:

```

AT
OK
AT+CPIN=XXXX
OK
+CPIN: READY
Call Ready
SMS Ready
AT+CREG?
+CREG: 0,5
OK
AT+SAPBR=3,1,"APN","Nombre del APN"
OK
AT+SAPBR =1,1
OK
AT+SAPBR=2,1
+SAPBR: 1,1,"XXX.XXX.XXX.XXX"
OK
AT+EMAILCID=1
OK
AT+EMAILTO=120
OK
```

```
AT+SMTPSRV="Servidor del email","Puerto"
OK
AT+SMTPAUTH=1,"Correo remitente","contrasena del correo"
OK
AT+SMTPFROM="Correo remitente","Nombre del remitente"
OK
AT+SMTPRCPT=0,0,"Correo destinatario","Nombre del destinatario"
OK
AT+SMTPSUB="Asunto del correo"
OK
AT+SMTPBODY=19
DOWNLOAD
This is a new email // debe tener una ongitud de 19 caracteres
OK
AT+SMTPSENDAT+SMTPSEND
OK
+SMTPSEND: 1
```

**Código 2.12:** Lista de comandos para el modulo SIM800L para el envio de un e-mail sin encriptación SSL (texto plano)

```

AT
OK
AT+CPIN=XXXX
OK
+CPIN: READY
Call Ready
SMS Ready
AT+CREG?
+CREG: 0,5
OK
AT+SAPBR=3,1,"APN","Nombre del APN"
OK
AT+SAPBR =1,1
OK
AT+SAPBR=2,1
+SAPBR: 1,1,"XXX.XXX.XXX.XXX"
OK
AT+EMAILCID=1
OK
AT+EMAILTO=120
OK
AT+SMTPSRV="Servidor del email","Puerto"
OK
AT+SMTPAUTH=1,"Correo remitente","contrasena del correo"
OK
AT+SMTPFROM="Correo remitente","Nombre del remitente"
OK
```

```
AT+SMTPRCPT=0,0,"Correo destinatario","Nombre del destinatario"
OK
AT+SMTPSUB="Asunto del correo"
OK
AT+SMTPBODY=19
DOWNLOAD
This is a new email // debe tener una ongitud de 19 caracteres
OK
AT+SMTPSENDAT+SMTPSEND
OK
+SMTPSEND: 1
```

**Código 2.13:** Lista de comandos para enviar un e-mail mediante el protocolo SSL con el modulo SIM800L

## 2.10. Conexión de Arduino a un servidor web

Dado que es necesario poder ver la posición del dispositivo localizador en la web, de alguna forma se debe conectar con ella y enviarle parámetros, pero para ello primero tiene que existir esa pagina web.

La web a la que accede el dispositvo SIM800L mediante peticiones, se ha realizado con los servicios gratuitos que ofrece Google, pero se puede realizar con una pagina web que el propio usuario haya desarrollado para tal efecto. En la sección 4.2 se muestran los pasos que se deben seguir para crear con éxito una pagina web.

En primer lugar, para el desarrollo de este ejemplo, se debe conectar el modulo SIM800L a Arduino MEGA de la forma que indica el esquema de la imagen (2.10.0.1).

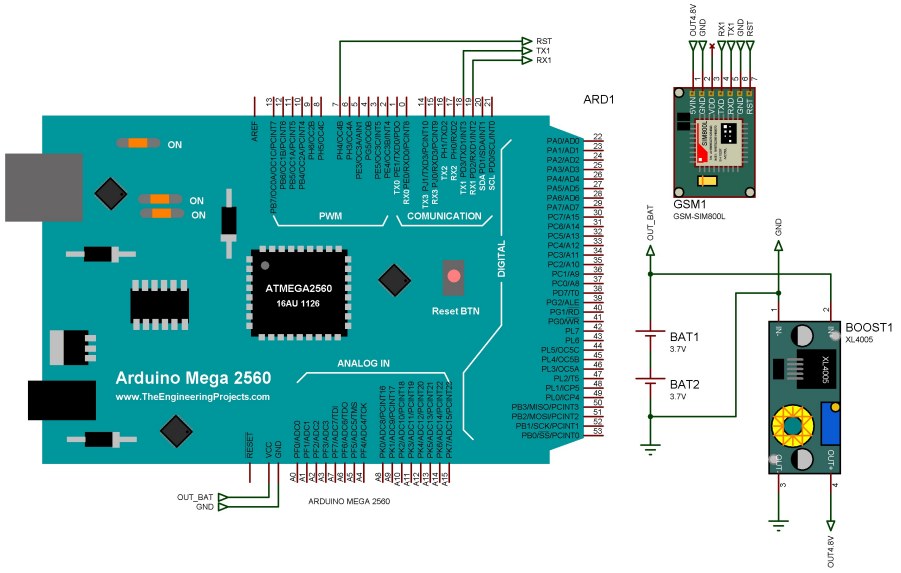



**Figura 2.10.0.1 –** Esquema de ejemplo de conexión del modulo SIM800L a Arduino MEGA para la realización de una petición HTTP

Luego, se debe subir a la placa Arduino MEGA el programa 2.10. En el código fuente es

necesario cambiar algunas variables, como el PIN, el nombre del APN de nuestro operador de telefonía y el propio APN del operador de telefonía del que se disponga.

```
1  //*********************************************************************
2  // Autor : CRISTÓBAL GARCÍA CAMOIRA
3  //*********************************************************************
4  //*********************************************************************
5  // CONEXIONES
6  // ARDUINO MEGA RX1   --> TX SIM800L
7  // ARDUINO MEGA TX1   --> RX SIM800L
8  // ARDUINO MEGA PIN 7 --> RST SIM800L
9  //*********************************************************************
10 /**
11  * Codigo de ejemplo para el manejo de un modulo SIM800L.
12  * A traves del modulo SIM800L, Arduino se conecta a la
13  * red GPRS y hace una peticion a google.
14  */
15 #include <Arduino.h>
16 #include <TimerOne.h>
17 #include <stdio.h>
18 #include <string.h>
19
20 /** Pin de reset del modulo SIM800L */
21 #define LOCATION_SYSTEM_GSM_GPRS_ON_OFF    (7U)
22 /** Habilita el pin de reset del modulo SIM800L */
23 #define LOCATION_SYSTEM_GSM_GPRS_ON()    digitalWrite(
    LOCATION_SYSTEM_GSM_GPRS_ON_OFF, HIGH)
24 /** Deshabilita el pin de reset del modulo SIM800L */
25 #define LOCATION_SYSTEM_GSM_GPRS_OFF()   digitalWrite(
    LOCATION_SYSTEM_GSM_GPRS_ON_OFF, LOW)
26 /** Longitud del buffer de respuesta del Modulo SIM800L */
27 #define LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT   (100U)
28 /** Longitud del array que contiene el comando AT para enviar al modulo SIM800L */
29 #define LOCATION_SYSTEM_GSM_GPRS_BUFFER_LENGHT     (100U)
30 /** Cadencia de acceso a la URL (segundos)*/
31 #define LOCATION_SYSTEM_20_SECONDS_TIMING   (20U)
32
33 /** Flag del timer */
34 static boolean location_system_flag_timer;
35 /** Url a la cual se suben las variables del sistema */
36 static char location_system_url[]={"http://www.google.es\""};
37 /** Array donde se almacena el numero pin de la tarjeta sim insertada en el modulo
    */
38 static char location_system_pin_number[]={"numero_PIN"};
39 /** Array donde se almacena el nombre que se le quiere dar a la conexion GPRS para
    el modem */
40 static char location_system_mobile_APN_name[]={"nombre_APN"};
41 /** Array donde se almacena el APN para la conexion a internet del modulo 2G */
42 static char location_system_mobile_APN[]={"telefonica.es"};
```

```
43 /** Array donde se almacena la respuesta del modulo SIM800L */
44 static char location_system_gsm_gprs_response1[
       LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT];
45 /** Array donde se escriben los comandos para enviar al moduo SIM800L*/
46 static char location_system_at_gprs_buffer[LOCATION_SYSTEM_GSM_GPRS_BUFFER_LENGHT];
47
48 void setup(void)
49 {
50   Serial.begin(9600);   // Inicializa el puerto serie.
51   Serial1.begin(9600);  // Inicializa el puerto serie 1.
52   pinMode(LOCATION_SYSTEM_GSM_GPRS_ON_OFF, OUTPUT);
53   Serial.println("Inicializando sistema...");
54   /** ############### GPRS-GSM ########################### **/
55   LOCATION_SYSTEM_GSM_GPRS_OFF();
56   delay(1000);
57   LOCATION_SYSTEM_GSM_GPRS_ON();
58   delay(4000);
59   /** Inicializo el modulo GPRS con el pin y se conecta a la red GPRS**/
60   sprintf(location_system_at_gprs_buffer,"AT+CPIN=%s", location_system_pin_number);
61   (void)location_system_gsm_gprs_sendATcommand(location_system_at_gprs_buffer,(char
       *)"OK", 2000);
62   delay(4000);
63   (void)location_system_gsm_gprs_sendATcommand((char*)"AT+CREG?",(char*)"OK", 2000);
64   delay(4000);
65   // Se introduce el APN y su nombre
66   sprintf(location_system_at_gprs_buffer,"AT+SAPBR=3,1,\"%s\",\"%s\"",
       location_system_mobile_APN_name, location_system_mobile_APN);
67   (void)location_system_gsm_gprs_sendATcommand(location_system_at_gprs_buffer,(char
       *)"OK", 2000);
68   delay(2000);
69   // Se habilita el GPRS.
70   (void)location_system_gsm_gprs_sendATcommand((char*)"AT+SAPBR=1,1",(char*)"OK",
       2000);
71   Serial.println("GPRS configurado!");
72   /** ########## VARIABLES DEL SISTEMA ################## **/
73   location_system_flag_timer = 0;
74   /** Inicializa el timer1 con una cadencia de 1s.**/
75   Timer1.initialize(1000000);
76   /** Realiza un callback (puntero a funcion) cuando el timer se desborda */
77   Timer1.attachInterrupt(location_system_isr_timer);
78 }
79
80 void loop(void)
81 {
82     location_system_1_second_timing();
83 }
84
85 /**
86  * \brief Envia los datos a la pagina web.
87  */
```

```
88  void location_system_send_HTTP(void)
89  {
90    // Inicializa el servicio HTTP
91    if (location_system_gsm_gprs_sendATcommand((char*)"AT+HTTPINIT",(char*) "OK",
        2000) == 1)
92    {
93      // Se inicializa el parametro CID
94      if (location_system_gsm_gprs_sendATcommand((char*)"AT+HTTPPARA=\"CID\",1",(char
        *) "OK", 2000) == 1)
95      {
96        sprintf(location_system_at_gprs_buffer, "AT+HTTPPARA=\"URL\",\"%s",
        location_system_url);
97
98        if (location_system_gsm_gprs_sendATcommand(location_system_at_gprs_buffer, (
        char*) "OK", 2000) == 1)
99        {
100         // Hacemos un GET de la pagina
101         if (location_system_gsm_gprs_sendATcommand((char*)"AT+HTTPACTION=0", (char*)
        "+HTTPACTION: 0,200,", 5000) == 1)
102         {
103
104           Serial.println(F("Peticion realizada"));
105         }
106         else
107         {
108           Serial.println(F("Error obteniendo url"));
109         }
110       }
111       else
112       {
113         Serial.println(F("Error poniendo la url"));
114       }
115     }
116     else
117     {
118       Serial.println(F("Error poniendo el CID"));
119     }
120   }
121   else
122   {
123     Serial.println(F("Error inicializando"));
124   }
125   location_system_gsm_gprs_sendATcommand((char*)"AT+HTTPTERM", (char*)"OK", 2000);
126 }
127
128 /**
129  * \brief Cada vez que salta la interrupcion del timer se acude a esta funcion.
130  */
131 void location_system_isr_timer(void)
132 {
```

```
133   location_system_flag_timer = true;
134 }
135
136 /**
137  * \brief Esta funcion se encarga de llevar el tiempo
138  */
139 void location_system_1_second_timing(void)
140 {
141   static uint8_t counter = 0;
142
143   if(location_system_flag_timer == true)
144   {
145     counter = (counter + 1) %LOCATION_SYSTEM_20_SECONDS_TIMING ;
146
147     if(counter == 0 )
148     {
149       location_system_send_HTTP();
150     }
151
152     location_system_flag_timer = false;
153   }
154 }
155
156 /**
157  * \brief Funcion que envia el comando AT al modulo SIM800L y espera a recibir
      respuesta.
158  */
159 int8_t location_system_gsm_gprs_sendATcommand(char* ATcommand, char* expected_answer
      , unsigned int timeout)
160 {
161   uint8_t x=0,  answer=0;
162   unsigned long previous;
163
164   memset(location_system_gsm_gprs_response1, '\0',
      LOCATION_SYSTEM_GSM_GPRS_RESPONSE_LENGHT); // Se limpia el buffer
165
166   delay(100); // Retardo para asegurar que no exixtan interferencias con otros
      comandos
167   while( Serial1.available() > 0) Serial1.read();    // Espera para borrar el buffer
       de entrada.
168   Serial1.println(ATcommand);    // Se envia el comando AT.
169   Serial.println(ATcommand);    // Se envia el comando AT.
170   x=0;
171   previous = millis();
172
173   // Este bucle espera por la respuesta
174   do{
175     // Si existen datos en el buffer de entrada, se leen y se comprueba la respuesta
       del modulo
176     if(Serial1.available()>0)
```

```
177     {
178       location_system_gsm_gprs_response1[x] = Serial1.read();
179       x++;
180       // revisa si la respuesta deseada es la respuesta que hemos obtenido del
      modulo.
181       if (strstr(location_system_gsm_gprs_response1, expected_answer) != NULL)
182       {
183         answer = 1;
184       }
185     }
186     // Se realiza una espera por la respuesta con el timeout
187   } while((answer == 0) && ((millis() - previous) < timeout));
188
189   return answer;
190 }
```

**Código 2.14:** Código de ejemplo para la realización de una petición a una pagina web desde Arduino MEGA con el modulo SIM800L

Si se conecta el cable USB desde Arduino al ordenador, se puede ver el resultado de la comunicación entre el Arduino y el modulo SIM800L.Aproximadamente cada 20 segundos, Arduino realizara una petición a Google.

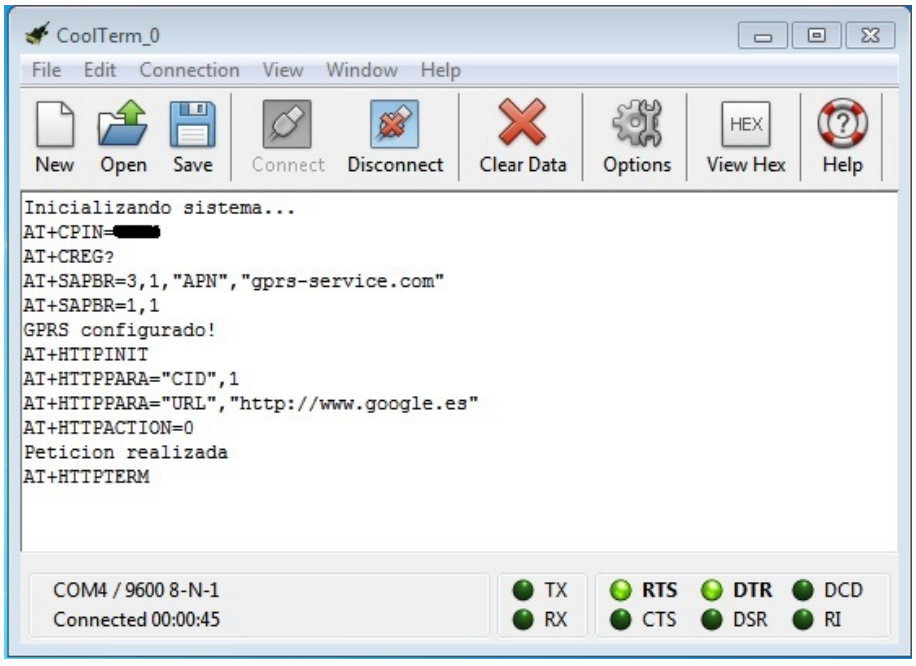



**Figura 2.10.0.2 –** Arduino realizando una petición web a Google

# 3 Condiciones de almacenamiento

- El dispositivo debe estar protegido de la luz solar para evitar un sobrecalentamiento de los circuitos.
- La temperatura de almacenamiento debe ser inferior a 50ºC.
- El ambiente debe ser seco y con ausencia de polvo.
- Proteger especialmente de la humedad y del agua.
- Mantenerlo alejado de superficies magnéticas o imantadas.

# 4 Guía de implementación

En esta última parte del pliego se elaboran las condiciones de implementación del equipo. Cuando se pretenda realizar el montaje del mismo se deberá recurrir a esta guía para asegurar que el montaje de los dispositivos sea el correcto.

## 4.1. Primeros pasos

Antes de comenzar a programar nuestro Arduino se deben instalar los drivers del mismo y el software de programación adecuados. Esto se puede descargar desde la página oficial de Arduino, en el siguiente enlace:

- http://arduino.cc/en/Main/OldSoftwareReleases

En este enlace hay que seleccionar el paquete disponible acorde al sistema operativo que se tenga instalado en el ordenador donde se va a usar el Arduino. Se recomienda descargar la versión de Arduino IDE 1.0.5 , para ello, hacemos click donde se muestra Windows Installer, tal y como muestra la figura 4.1.0.1. Este archivo es un instalador automático del compilador de Arduino, solo hay que seguir los pasos que se indican a continuación, aunque son bastante intuitivos, se explican para que no queden dudas. De esta forma se asegura la compatibilidad de todas las librerías utilizadas durante la fase de realización del proyecto.

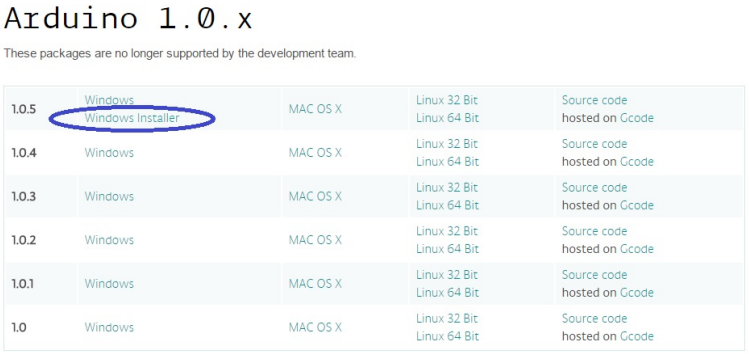



**Figura 4.1.0.1 –** Instalación del IDE Arduino 1

Una vez descargado el fichero ejecutable del enlace anterior se presiona en el recuadro donde se muestra **I agree** tal y como se muestra en la figura 4.1.0.2:


**Figura 4.1.0.2 –** Instalación del IDE Arduino 2

Se seleccionan todos los componentes que se indican, como se muestra en la figura 4.1.0.3 y presionamos **Next**.

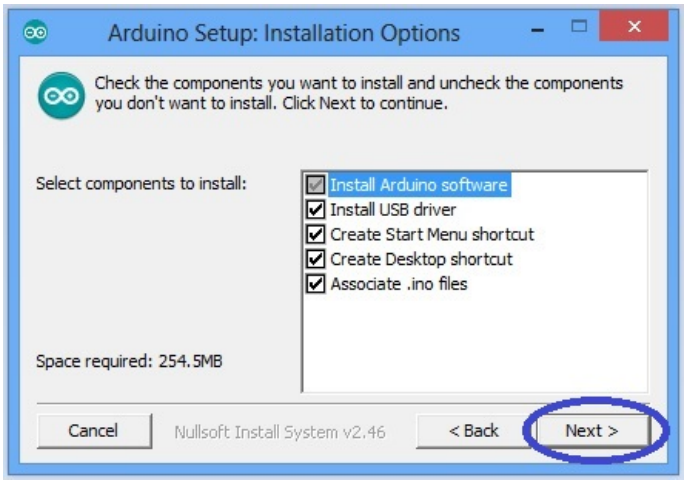



**Figura 4.1.0.3 –** Instalación del IDE Arduino 3

El siguiente paso es indicarle la ruta de instalación de Arduino, ruta que por defecto es la siguiente: `C:\Program Files (x86)\Arduino`

El sistema operativo utilizado durante la instalación ha sido Windows 7, puede que en Windows Vista, 8, o XP venga determinada por defecto una ruta diferente.

Al cabo de unos minutos el compilador de Arduino habrá finalizado su instalación.

Los drivers que controlan las diferentes placas de Arduino se instalaran automáticamente en Windows 7, solo se tiene que conectar Arduino al PC y en la pantalla aparecerá un ventana como la que que se muestra en la figura 4.1.0.4.

El sistema operativo se encargará del resto. En esta parte de la instalación es necesario tener un poco mas de paciencia.

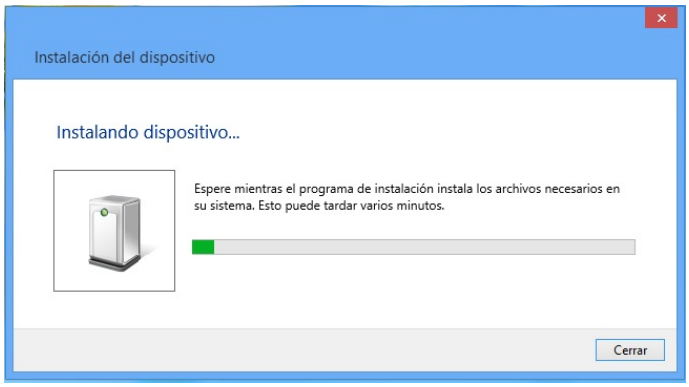



**Figura 4.1.0.4 –** Instalación del IDE Arduino 4

Una vez instalado el programa y los drivers, y con el Arduino ya conectado, se necesita saber qué puerto COM le ha asignado el ordenador, para ello se hace click derecho en Equipo, se hace click en Administrar – Administrador de dispositivos – Puertos (COM y LPT) y se vera qué puerto le ha asignado el ordenador. Esto se muestra en la figura 4.1.0.5.

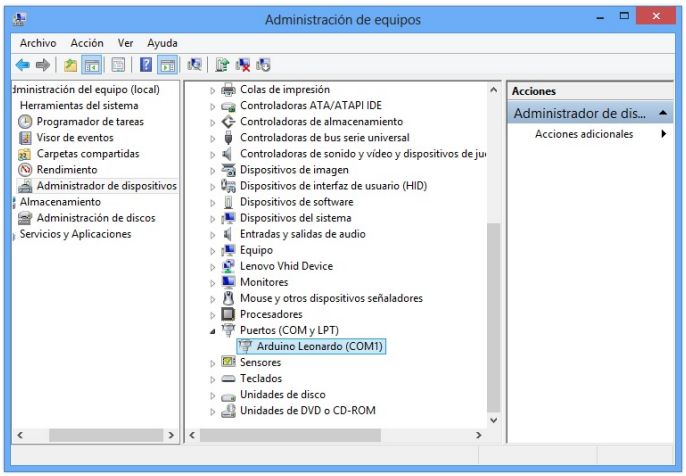



**Figura 4.1.0.5 –** Instalación del IDE Arduino 5

En el caso de que se disponga de otro sistema operativo tal como windows XP y los drivers no se instalen automáticamente, se pueden seleccionar de la siguiente carpeta situada en el directorio en el cual se ha instalado el programa de Arduino.

Esta carpeta se muestra en la figura 4.1.0.6.

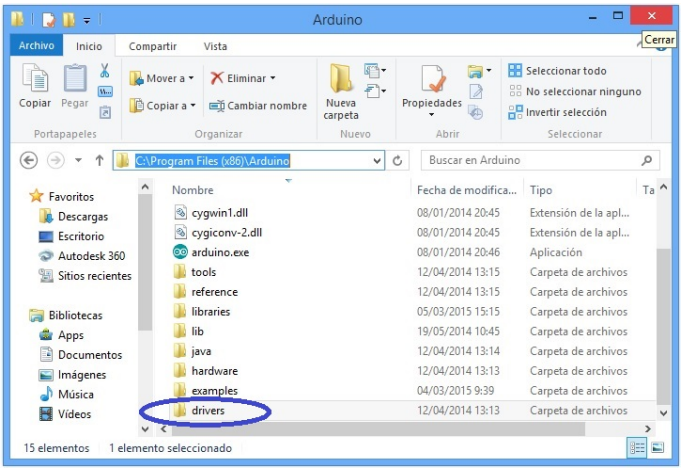



**Figura 4.1.0.6 –** Instalación del IDE Arduino 6

Una vez en este punto , es necesario descargar e instalar las siguientes librerías (en el caso de que no se dispongan), estas son:

- LiquidCrystal_I2C: Para poder realizar pruebas con una pantalla LCD de 16x2 líneas.
- TimerOne: Para poder establecer una base de tiempos.

Estas librerías se encuentran fácilmente en la web y para poder utilizarlas basta con depositar el fichero descomprimido en la carpeta libraries del directorio en el cual se ha instalado el IDE de Arduino, que por defecto es `C:\Program Files (x86)\Arduino`.

Llegados a este punto ya se puede programar la placa Arduino con el código fuente que se indica en la seccion 10.4 del documento MEMORIA.

Una vez realizados todos los pasos anteriores el hardware del dispositivo de localización esta preparado para ser programado de forma que los periféricos conectados funcionen de forma correcta.

Lo que se debe hacer a continuación es compilar y cargar el archivo fuente con extensión `.ino` con el Arduino Mega.

## 4.2. Creación de un servidor web gratuito

En este apartado se pretende explicar de forma breve la forma mas sencilla que se ha descubierto para poder almacenar los datos que se reciban, en este caso del dispositivo localización, pero puede ser de cualquier otro dispositivo, y que estos datos permanezcan almacenados.

Por otro lado también se pretende mostrar la forma en como estos datos se portan a una pagina web en la cual ha de ser necesario loguearse para poder visualizar la información.

En primer lugar es necesario conocer thingspeak, thingspeak una Plataforma Open Source para conectar productos y servicios tal y como se comenta en http://iot-spain.com/?p=188, tan solo se necesita registrarse en esta pagina y ya se puede comenzar a enviar datos desde el dispositivo que se ha desarrollado en el presente trabajo.

Los pasos para la configuración de thingspeak son los siguientes:

1. Registrarse en la pagina thingspeak.

2. Cumplimentar en la pestaña que indica la figura 4.2.0.1 los campos de los datos que se deseen enviar desde el dispositivo localizador.

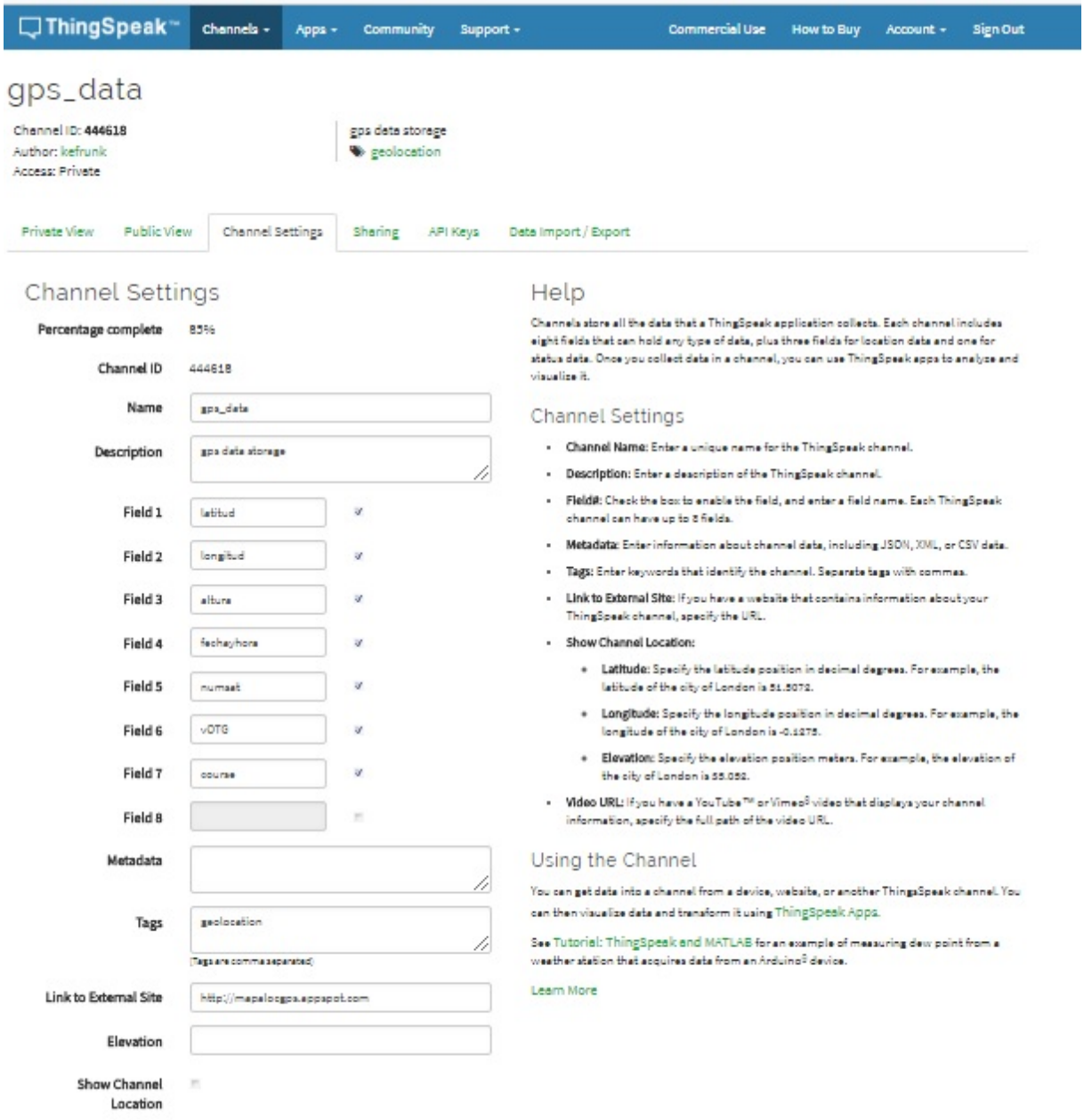



**Figura 4.2.0.1 –** Campos de la base de datos de Thingspeak

3. En la pestaña de la siguiente figura 4.2.0.2 se puede ver la URL mediante la cual se pueden enviar datos desde un dispositivo conectado a internet a esta base de datos. La url tiene el aspecto siguiente:

   - `http://api.thingspeak.com/update?api_key=XXXXXXXXXXXX&field1=var1`

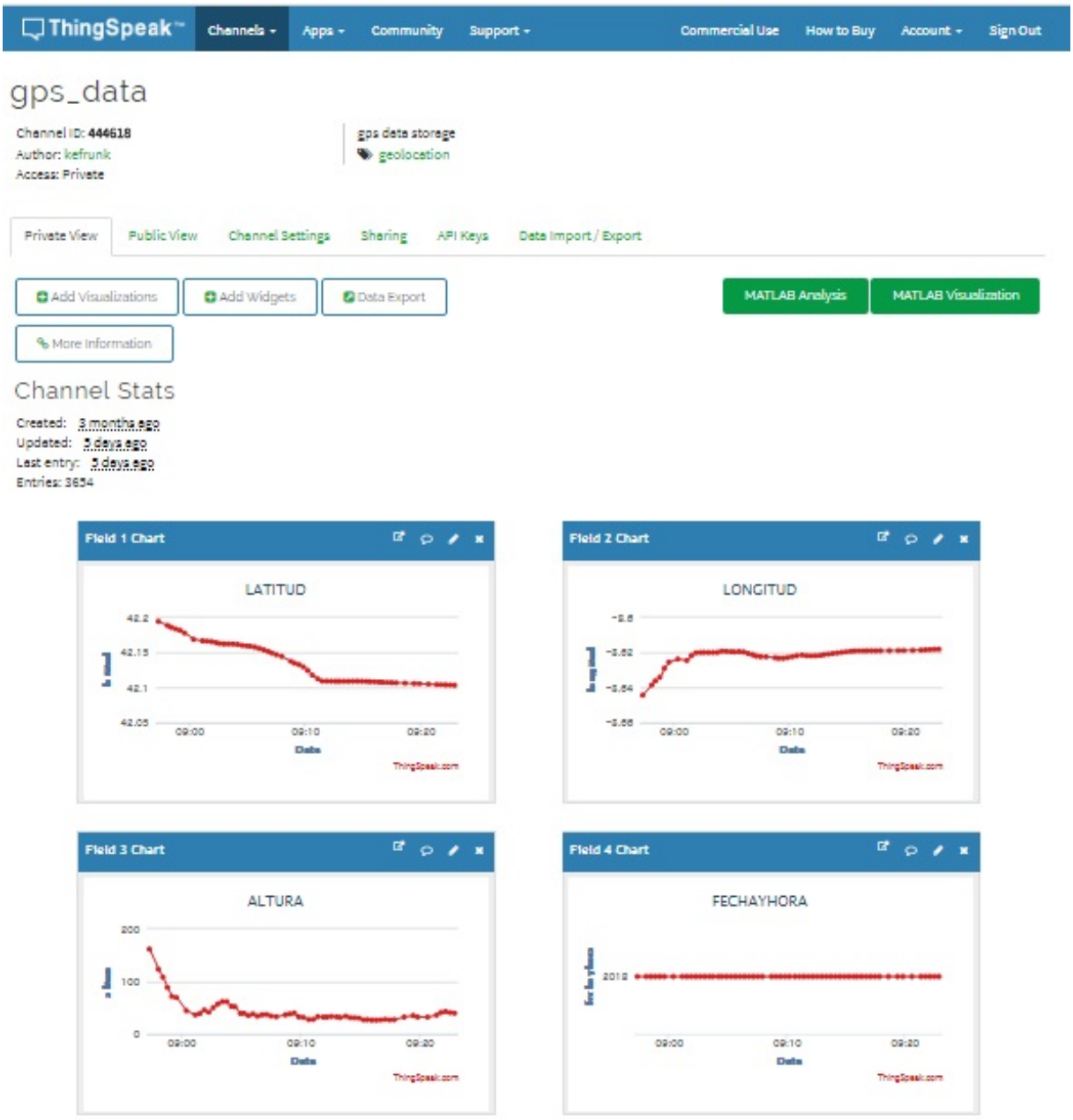



**Figura 4.2.0.2 –** Url a través de la cual se envían los datos desde el dispositivo a la base de datos de Thingspeak

Se enviaran tantas variables «var»como campos se tengan agregados en dicha base de datos.

4. Se puede comprobar en la pagina web que los datos enviados son los que se están almacenando en la base de datos, tal y como se mustra en la figura 4.2.0.3.

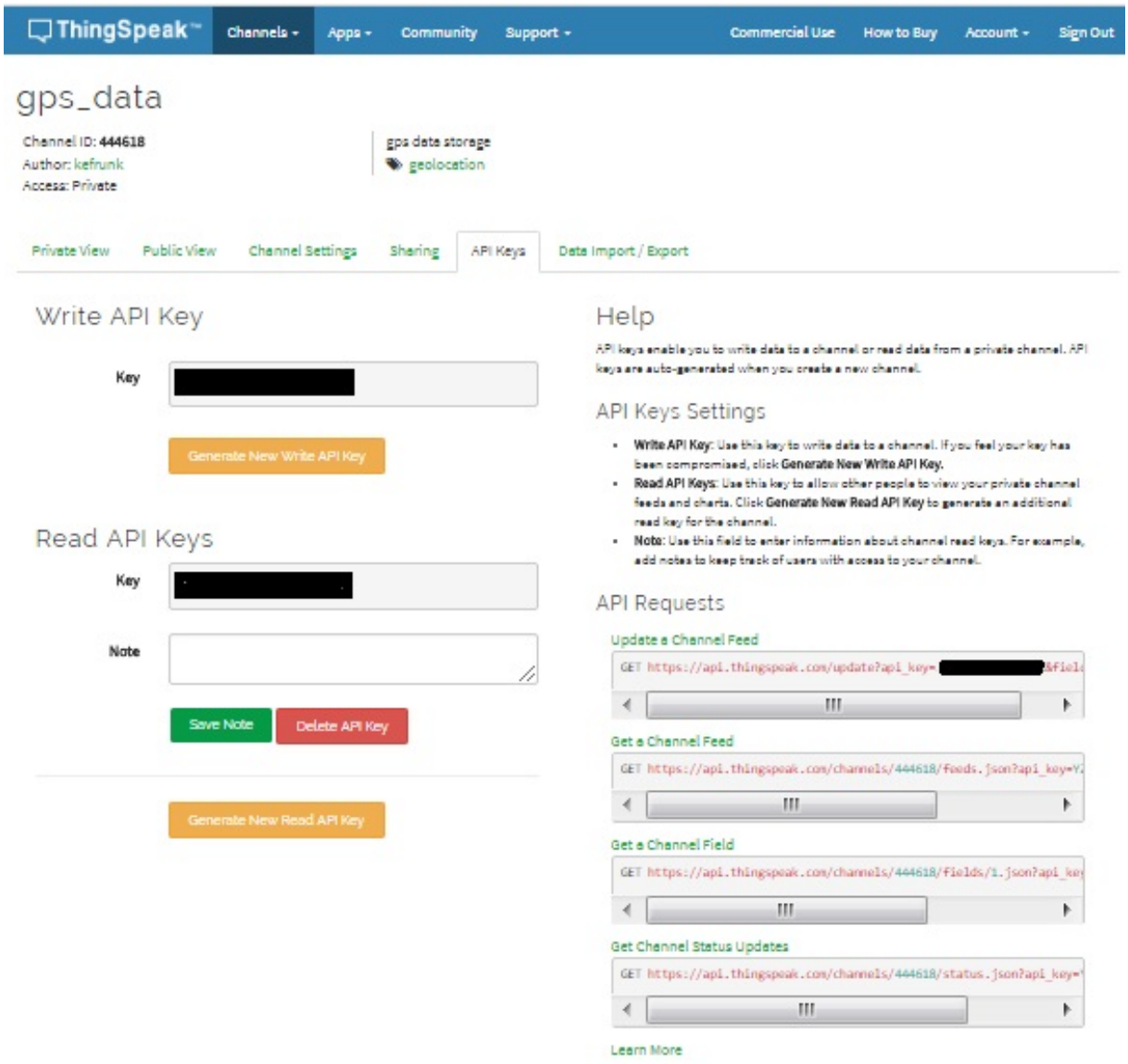



**Figura 4.2.0.3 –** Gráficas de datos recibidos en el servidor Thingspeak

La única contra que tiene el utilizar thingspeak como base de datos de forma gratuita es que no permite guardar datos con una cadencia inferior a 15 segundos, esto puede ser un problema si se fuese a viajar con el dispositivo de localización a una velocidad muy alta, pero aun así, si se obtiene un punto cada 15 segundos, se podría intuir la ruta seguida.

La creación de una pagina web es una tarea sencilla si se compra un dominio, si no, se tiene que recurrir a servicios gratuitos como por ejemplo los de Google, que son los que se han utilizado para el desarrollo de este proyecto. En este aspecto no se va a entrar mucho en detalle ya que son necesarios conocimientos muy técnicos acerca de lenguajes como java y HTML. Los pasos a seguir son los siguientes:

1. Instalar Google appengine en el ordenador en ella se pueden crear aplicaciones en la

nube de todo tipo, en el caso que ocupa a este proyecto, sera una aplicación que recogerá los datos almacenados en la cuenta de thingspeak creada y los pintara en un mapa (también proporcionado por google).

2. Para realizar la aplicación mas básica es necesario seguir el ejemplo que se indica en la pagina web siguiente https://cloud.google.com/appengine/docs/, y, a partir de ahí es necesario ir evolucionando.

3. Insertar un mapa de Google en una aplicación no es una tarea muy complicada, ellos mismos ya preparan el código para que sea sencillo de realizar, una mezcla de java y HTML, a cambio obtendrán lo que ellos desean, los datos de las personas, en este caso los relativos a la posición, velocidad...

4. No hay que olvidar que para utilizar las apis de Google (como por ejemplo googgle maps) es necesario obtener unas claves personales «api keys», el siguiente enlace explica como realizarlo https://developers.google.com/maps/documentation/geocoding/get-api-key.Sin estas claves se estaria colocando un recuadro vacío en el que no se vería nada.

## 4.3. Preparación de los diferentes circuitos

### 4.3.1. Preparación del modulo SIM900A- SIM800L

Una vez instalado el software de Arduino, es el momento de configurar y comprobar el funcionamiento de los diferentes módulos que son necesarios para llevar a cabo el presente proyecto, se comienza por el modulo GSM.

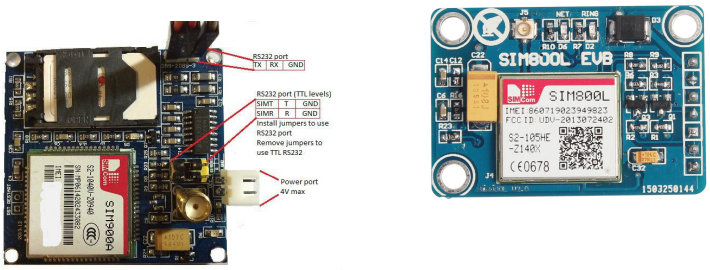
El modulo GSM utilizado en el presente proyecto es el mas barato existente en el mercado actual pudiéndolo encontrar por menos de 10 euros en tiendas online como Ebay o Amazon.

Hay que tener en cuenta, y debido a su bajo precio que el modulo adquirido es un dispositivo 2G. En España cualquiera de las compañías de telefonía que operan funcionaran con este dispositivo incluyendo las compañías virtuales tras los cambios que se realizaran en el firmware posteriormente. Se recomienda adquirir (siempre que la economía lo permita) un módulo 3G en su lugar.

La intención de estas lineas es cotejar la información para que alguien utilice o pruebe un módulo nuevo Sim900A - SIM800L GPRS/GSM, conectado a un PC, o microcontrolador.

Toda la información que se pretende mostrar en esta sección no es nueva, ha sido recopilada mediante diferentes portales web que se irán mencionando a lo largo de esta sección. El uso principal que se le va a dar a este dispositivo es poder conectarlo a la web y transmitir la información desde cualquier punto de España (aunque en teoría puede servir para cualquier región Europea).

Los módulos SIM900A y SIM800L (4.3.1.1)son similares al módulo SIM900, y se conocen como “mini” tarjetas de desarrollo. A ser posible y para evitar complicaciones como las que han surgido a lo largo del desarrollo de este proyecto, se recomienda el uso de un modulo que contenga el chip SIM900, ya que es un dispositivo cuatribanda que funciona en todas partes.




(a) Modulo SIM900A utilizado durante la fase de pruebas en el proyecto

(b) Sim800L evb utilizado en el proyecto

**Figura 4.3.1.1 –** Representación de los módulos SIM900A y SIM800L

A continuación y a lo largo de estas lineas se hablara de la configuración y peculiaridades que ofrece el modulo SIM900A debido a que este modulo ya se disponía de inicio. Se han realizado numerosas pruebas con este modulo, y, en base a su respuesta de funcionamiento, se ha decidido realizar la compra de un modulo algo mas versátil (SIM800L) y que diese menos problemas de configuración que el modulo SIM900A, ademas el SIM800L posee un tamaño mas compacto y, lo que es mas importante, cumple uno de los requisitos mas importantes, que es el de poder enviar correos electrónicos, algo que el módulo SIM900A no soporta en su versión de firmware original.

El Sim900A es un dispositivo de 2 bandas, en 900-1800MHz y está limitada a nivel regional según el siguiente enlace http://www.blog.zapro.dk/?p=368 que enumera las regiones en las que el módulo va a funcionar, y proporciona acceso al documento SIMCOM que describe las restricciones. Las restricciones sólo se aplican después de la versión de firmware "1137B06SIM900A64_ST_ENHNCE".

La mayoría de las tiendas de que venden módulos como el SIM900A no indican que el chip SIM900A posee un uso restringido, es decir, solo funciona en determinadas regiones. Esto puede ser por temor a no ser vendidos o por lo complicado que puede resultar explicarle a alguien sin conocimiento como reflashear el firmware.

Existen unos pocos sitios web que hablan de volver a grabar el firmware para que funcione en otras partes del mundo con éxito. Se recomienda echar un vistazo a los siguientes enlaces para obtener más información.

- http://amichalec.net/2012/05/andida/
- http://pixelatedpic.blogspot.fr/2013/08/simcom-sim900a-fixed.html

La serie de versiones de firmware, así como los programas dedicados a flashear cada uno de los módulos están disponibles en el siguiente enlace:

- http://dostmuhammad.com/sim900-firmware-update-tutorials-appnotes/

El módulo SIM900A ha sido adquirido a través de una tienda de origen asiático. Este se muestra en la imagen 4.3.1.1 y poseía la versión de firmware 1137B05SIM900AM64_ST_ENHANCE. Al traer una versión de firmware anterior a la 1137B06SIM900A64_ST_ENHNCE, tras la carga del nuevo firmware ha podido funcionar sin problemas.

Cabe comentar que la red 2G no está cifrada durante la transmisión y no es "segura". De acuerdo con Steve Gibson en GRC.com 3G y 4G si están codificadas en el aire, pero no una vez recibidas por la estación base.

#### 4.3.1.1. Alimentación del módulo SIM900A y SIM800L

El manual de hardware de diseño del módulo SIM900A la página web de SIMCOM SIM900ASIMCOMhttp://wm.sim.com/upfile/20121129151745f.pdf (es necesario registrarse primero para poder descargarselo, pero también buscando por Google lo podemos encontrar) proporciona un diseño de referencia de la fuente de alimentación necesaria para este dispositivo, también afirma que esta fuente de alimentación debe ser capaz de subministrar picos de corriente de hasta 2A, (La alimentación de 5V producida por el puerto USB es insuficiente, ya que no puede suministrar la corriente necesaria. El módulo consumirá estos picos de corriente durante el envío de un SMS, para el resto de servicios tales como realizar llamadas, conectarse como cliente a un servidor, funcionar como servidor... la alimentación que proporciona el puerto USB de los ordenadores existentes hoy día en el mercado es mas que suficiente).

Con lo cual y debido a lo anteriormente citado, se debe utilizar una fuente de alimentación externa que sea capaz de subministrar 3.7V hasta 4.2V, y 2 amperios de corriente (para el modulo SIM900A).

Algunos vendedores dicen que se puede alimentar el módulo a 5V, aunque mas seguro es utilizar un regulador de tensión que reduzca esta tensión a la recomendada.

La tensión máxima de alimentación indicada por el fabricante es de 4,8V.

Para las pruebas se ha utilizado una fuente de alimentación de 12V 2A y un convertidor DC-DC Buck-Boost disponible en eBay y funciona de maravilla, aunque se recomienda ademas conectar un condensador electrolítico grande a la salida para suministrar la corriente de pico (en el caso de que en nuestra aplicación se necesiten enviar SMSs).

Si el led de nuestra placa denominado "Netlight" (LED D6) parpadea lentamente y de manera constante todo funciona correctamente, de lo contrario si se obtienen 4 caracteres que se repiten en el PC y nada más, particularmente si el indicador del anillo (RI, LED D5) también de vez en cuando parpadea, la fuente de alimentación se cortocircuita durante esos altos impulsos de corriente y el módulo se reseteará.

#### 4.3.1.2. Conexionado para la comunicación

El módulo SIM900A posee 2 puertos serie de comunicación RS232. Uno es para las comunicaciones mas comunes (para comunicar un PC con el módulo), y el otro es el puerto de comunicación denominado de "servicio", utilizado para subir o cambiar el firmware que dispone.

El presente módulo se comunica con niveles de voltaje TTL (0 al voltaje de alimentación -0.1V), ademas el módulo posee un chip MAX232 en el puerto de comunicaciones RS232 sólo para traducir los niveles de voltaje TTL a los niveles de voltaje estándar (-12 V a 12 V) y pueda conectarse directamente a un PC. Se puede acceder al puerto de comunicaciones a niveles TTL extrayendo los puentes existentes cerca del conector de la antena. (Para ser estrictamente correcto, el MAX232 produce niveles de voltaje de entre -7V y + 7V).

El puerto de depuración solo esta disponible para niveles TTL y para poder usarlo es necesario soldar unos cables al módulo como figura en la imagen 4.3.1.2 aunque no vamos a utilizar esta forma de conexionado para realizar la comunicación.

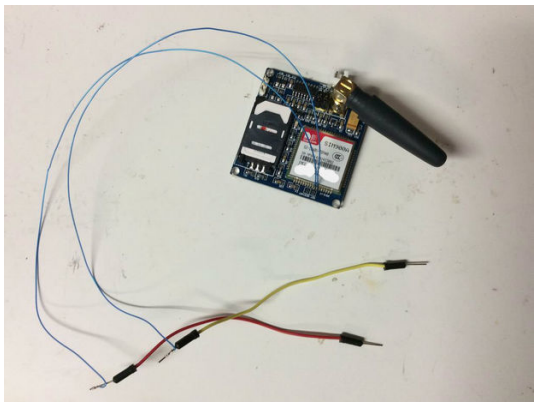


**Figura 4.3.1.2 –** Puerto de depuración del módulo SIM900A

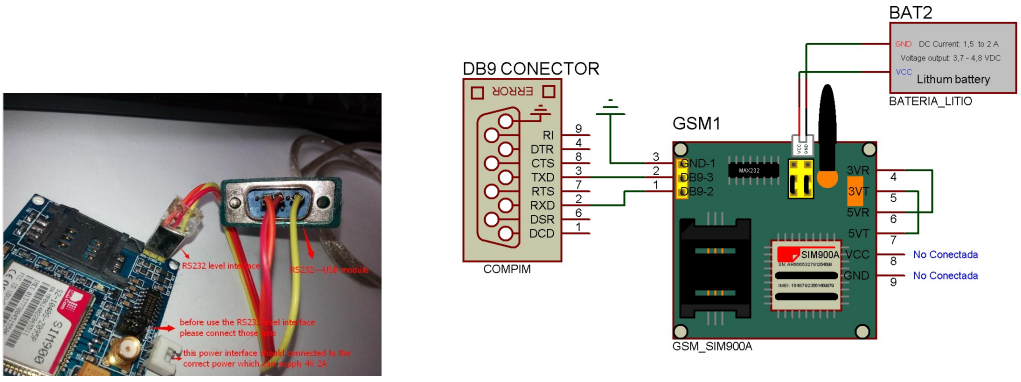
Los pines de salida del puerto serie de comunicaciones en niveles de tensión estándar se muestran en las imágenes 4.3.1.3, en otras versiones del módulo (el módulo que estamos utilizando es la versión de placa ”mini V3.8.2’) dependiendo de la versión de placa que dispongamos, los pines de salida TTL pueden estar colocados de forma diferente, por ello se recomienda identificar los pines de masa con un polímetro.

En resumen, se puede establecer la comunicación con el módulo SIM900A de 2 formas diferentes, la primera seria conectarlo a través de un conversor usb-serie al PC y la segunda seria conectarlo directamente a los microcontroladores, tales como Aduino,Pic, etc :

1. **Conexión directa al PC**

   Este tipo de conexión es viable si nuestro ordenador posee puerto serie de comunicaciones, aunque este puerto la gran mayoría de los ordenadores actuales no lo posee. La imagen 4.3.1.3 muestra este tipo de conexionado, a través de esta conexión ya se puede establecer una comunicación exitosa con el modulo SIM900A. Es muy importante colocar los jumpers tal y como muestra la figura 4.3.1.3 y el esquema realizado, sin ellos ,el PC no detectara que un nuevo dispositivo ha sido conectado. **NOTA: Observar que el pin TX del módulo GSM se tiene que conectar al pin RX del conector DB9 y el pin RX del módulo GSM se tiene que conectar al TX del conector DB9 para obtener una conexión exitosa.**

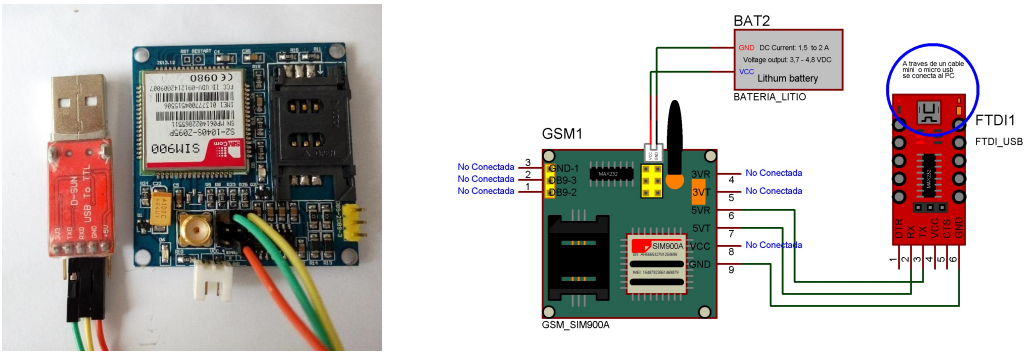
2. **Conexión indirecta al pc**




(a) Imagen (b) Esquema

**Figura 4.3.1.3 –** Conexión directa al PC mediante cable DB9

Esta conexión es la mas común, puesto que la anterior forma de conexión solo se podría realizar teniendo en cuenta que el PC que se fuese a utilizar dispusiese de puerto RS232. Para establecer este tipo de conexión es necesario disponer de un módulo FTDI cuya referencia en la web es "***FT232RL USB FTDI a TTL***", este módulo realiza la misma función que el chip MAX232 que posee el modulo SIM900A con los jumpers conectados, convierte los niveles de tensión del puerto USB del ordenador a niveles TTL. Para realizar esta conexión es necesario extraer los jumpers situados al lado de la antena y conectarlo de la forma que muestra la imagen 4.3.1.4(a) y el esquema 4.3.1.4(b). **NOTA: Observar que el pin TX del módulo GSM se tiene que conectar al pin RX del FTDI y el pin RX del módulo GSM se tiene que conectar al TX del módulo FTDI para obtener una conexión exitosa**.



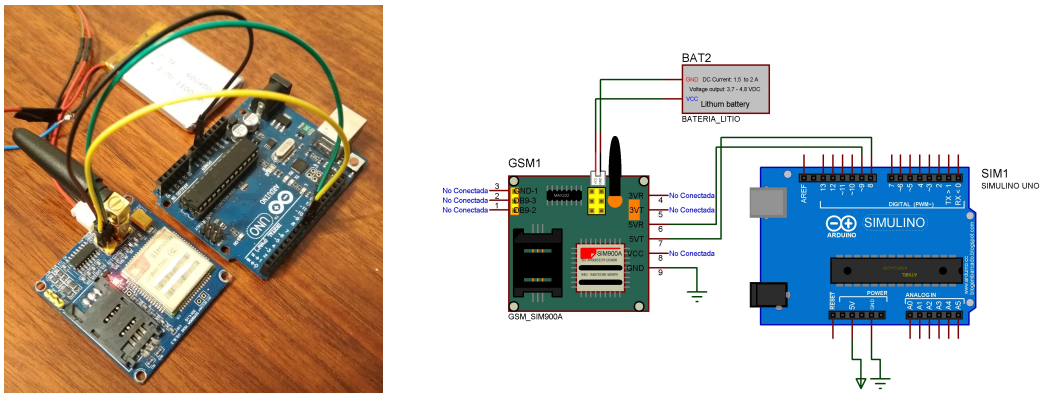

(a) Imagen (b) Esquema

**Figura 4.3.1.4 –** Conexión directa al PC mediante adaptador FTDI

3. **Conexión a Arduino, o a cualquier microcontrolador (Pic, Atmel, Intel...).**

   Esta conexión es la mas sencilla aunque es necesario disponer del código implementado para poder probarla, consiste en conectar las salidas TTL , RX y TX (ya sean las de 5V o las de 3,3V dependiendo del microcontrolador que se utilice para la aplicación, en este caso se muestra el conexionado a un Arduino UNO, ATMEGA328P , que utiliza las

salidas de 5V) a las salidas serie de nuestro microcontrolador RX y TX. En Arduino y gracias a la enorme cantidad de librerías existentes (SoftwareSerial.h en este caso), se puede realizar la comunicación serie en prácticamente cualquier par de pines digitales de Arduino UNO, de esta forma se utilizan pines digitales para la comunicación serie con el módulo GSM dejando libres los pines 0 y 1 libres para poder depurar el código. En la imagen 4.3.1.5(a) y esquema 4.3.1.5(b) se muestra un ejemplo de la conexión en la que el pin RX es el pin 8 de Arduino y el pin 9 el TX. Esto se consigue gracias a la siguiente instrucción: `SoftwareSerial serialSIM(8,9); // RX=PIN8; TX=PIN9`




(a) Imagen (b) Esquema

**Figura 4.3.1.5 –** Conexión del módulo GSM a nuestro Arduino

NOTA: El conexionado es el mismo tanto para el módulo SIM900A como para el módulo SIM800L, de todas formas, el conexionado del módulo SIM800L con Arduino ya se ha mostrado en la figura (2.7.0.1).

#### 4.3.1.3. Reflashear firmware de los módulos del fabricante SIMCOM

Como se ha dicho al principio de este apartado, puede ser necesario reflashear el firmware del módem que se este utilizando (ya sea SIM800L o SIM900A) por diversos motivos que se enumeran a continuación:

1. En el caso del módulo SIM900A, no puede funcionar en la región europea, solo esta preparado para funcionar en países de la región asiática, cargándole un firmware diferente ( el firmware perteneciente a los módulos SIM900), podría funcionar sin problemas.

2. Puede que el módulo venga con un firmware antiguo y existan nuevas versiones que soporten comandos que con el antiguo firmware serían imposibles de ejecutar con éxito.

Las líneas que siguen a continuación servirán de guía para poder utilizar estos módulos con la ultima versión disponible en la web. Los pasos a seguir se enumeran a continuación:

1. Conocer la versión de firmware del módulo del que se disponga para saber si se debe actualizar o no. Es recomendable saberla debido a que puede que en la web no exista la versión que dispone el módulo, y le le flashee una versión de firmware equivocada, con

lo cual el módulo quedaría inservible a menos que se localice la versión de firmware que traía de fabrica y se reflashee de nuevo.

2. Se realiza el montaje que se muestra en la figura (4.3.1.4) para el caso del módulo SIM900A, en el caso del módulo SIM800L se realiza el que se muestra a continuación para despejar cualquier tipo de duda.

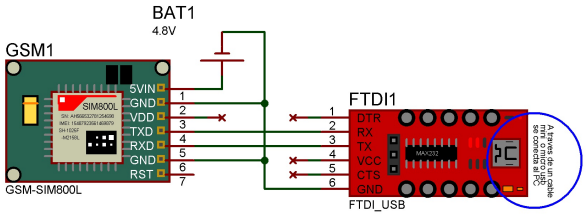



**Figura 4.3.1.6 –** Esquema de conexionado del módulo SIM800L para cargarle un nuevo firmware

3. Se ejecutan unos comandos AT para poder verificar que la conexión esta bien hecha y el módulo responde, es en este paso cuando se debe aprovechar para ejecutar el comando que sirva para conocer la versión de firmware de la que dispone el módulo.

   Este comando es el siguiente: AT+GSV

4. Se necesita descargar el software el «**SIMXXX customer flash loader**» (XXX indica el módulo, puede que exista un programa diferente para cada versión de módulo), este programa sera el encargado de recargar un nuevo firmware a nuestro módulo SIM.

5. Se descarga el firmware que se le desee flashear del siguiente enlace: http://dostmuhammad.com/sim900-firmware-update-tutorials-appnotes/.El fichero necesario posee la extensión (**.cla**).

6. Se abre el programa descargado («SIMXXX customer flash loader») y se se selelecciona la carpeta que contiene el firmware deseado para cargar. En este caso, en la figura (4.3.1.7) se muestra la herramienta utilizada para flashear el firmware de la placa SIM900A, cargándole el firmware de un chip SIM900 (en este caso denominado 1137B03SIM900M64_ST_ENHANCE.cla). De esta forma se elimina la limitación regional que este chip posee y así permitir que funcione en países de la región europea.

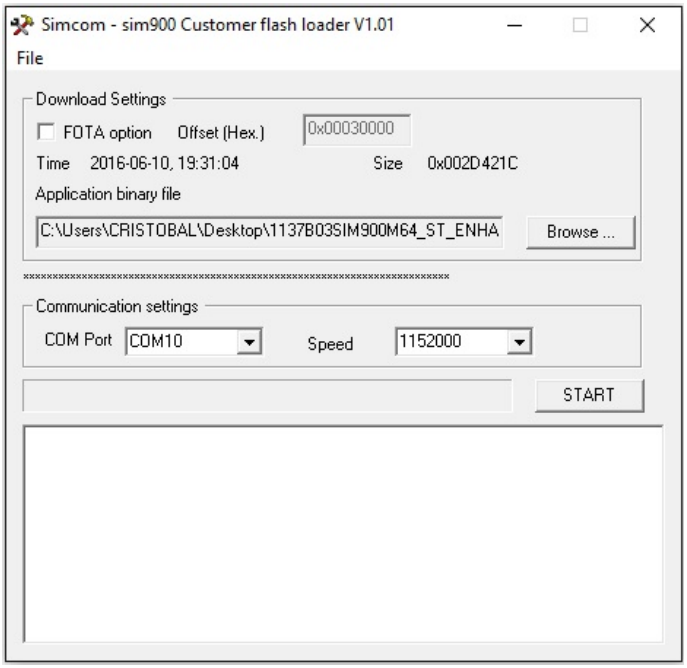
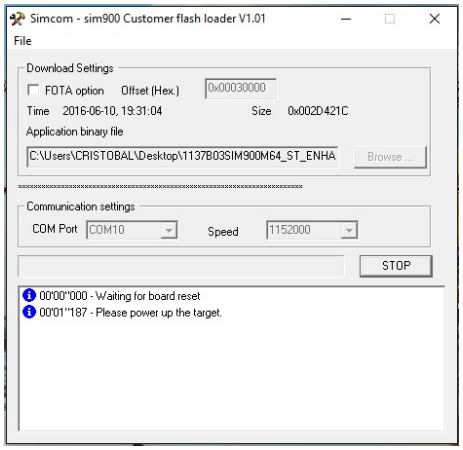


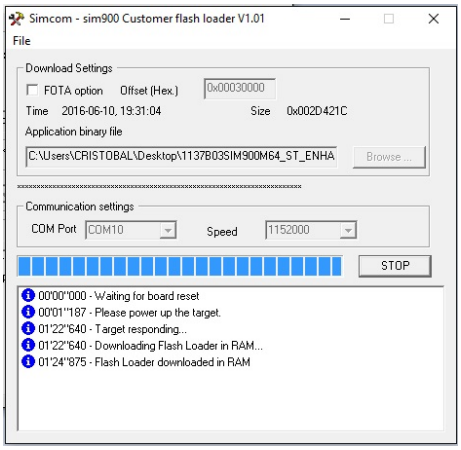

**Figura 4.3.1.7 –** Imagen de la herramienta "**Sim900 customer flash loader**"necesaria para cargar un nuevo firmware a la placa SIM900A

7. Una vez seleccionado el fichero, se selecciona la velocidad del bus para la transferencia del firmware (baudrate). Se recomienda 1152000 que es a la cual ha funcionado durante las pruebas, de elegir una velocidad de transferencia menor, el tiempo de carga del firmware aumentará.

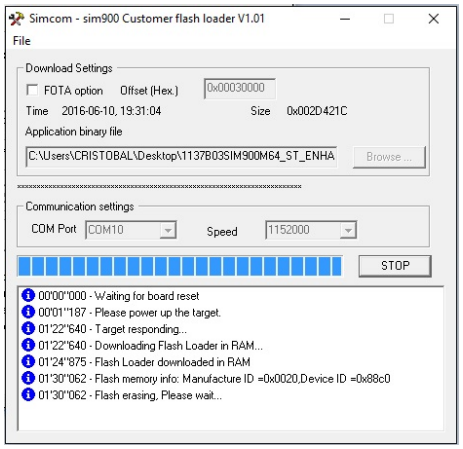
   Es necesario tener mucha paciencia en este punto. Por ultimo se presiona el botón de Start y se requerirá un reset de la placa (desconectando la placa de la alimentación y volviéndola a conectar es mas que suficiente). Ver figura (4.3.1.9).



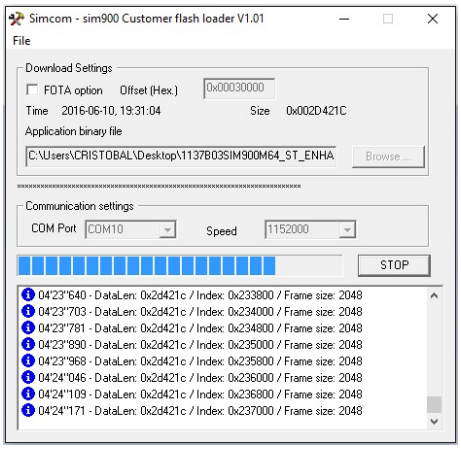

(a) Esperando Reset




(b) Cargando en Ram el firmware

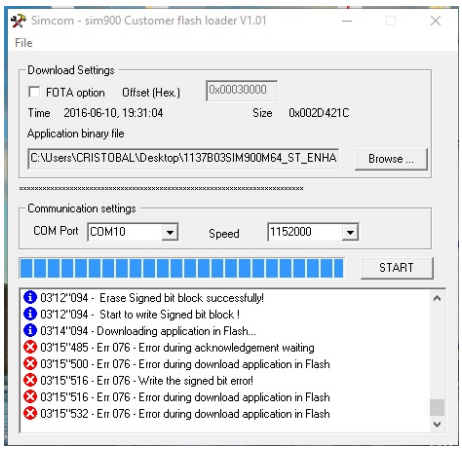



(c) Borrando el firmware anterior de la memoria flash




(d) Flasheando el firmware

**Figura 4.3.1.8 –** Flasheando el firmware de la placa SIM900A




(a) Flasheando el firmware

**Figura 4.3.1.9 –** Error ocurrido durante la carga del firmware de la placa SIM900A

8. Puede darse el caso de que al final de la carga del firmware el proceso haya tenido algún error, ya sea porque la velocidad es excesivamente alta, o porque durante la transferencia

del firmware ha ocurrido algo que no debería. En este caso se aconseja repetir de nuevo los pasos anteriormente mencionados. Este error puede darse al principio, durante, o al final de la carga del firmware. Un ejemplo de este error es el que se muestra en la figura(4.3.1.9(e)).

### 4.3.2. Preparación del modulo Bluetooth HC-05

Al modulo bluetooth no es necesario realizarle una configuración previa para su correcto funcionamiento, si se realizan las pruebas de verificación indicadas en el apartado 2.6 , no debería existir ningún inconveniente para su correcto funcionamiento.

### 4.3.3. Preparación del modulo GPS NEO-6M

Al modulo GPS no es necesario realizarle una configuración previa para su correcto funcionamiento, si se realizan las pruebas de verificación indicadas en el apartado 2.5, no debería existir ningún inconveniente para su correcto funcionamiento.

## 4.4. Montaje de los circuitos

Para el montaje completo del sistema de localización se deben seguir las siguientes pautas:

1. En primer lugar es necesario elaborar la placa de circuito impreso que se indica en el documento de los planos **[Plano 2]**. Esta parte es la que puede ser mas complicada, ya que no se ha externalizado la fabricación de los circuitos impresos utilizados en el presente proyecto.

   A continuación se explica de forma breve el proceso de fabricación de la placa de circuito impreso que se ha realizado.

   - En primer lugar se imprime en papel cebolla el circuito que se indica en el documento de los planos **[Plano 3]**. Este circuito se llama fotolito.
   - Luego, se corta la placa de circuito impreso (fotosensible) con el tamaño que se indica en el **[Plano 3]** del documento de los planos.
   - Una vez hecho esto, se introduce en la insoladora la placa. La insoladora (4.4.0.1) es un aparato mediante el cual se copia una imagen (negativa o positiva, según el proceso) al iluminar el fotolito que la contiene, que se ha puesto en contacto directo con una superficie fotosensible (película, papel, placa o plancha) sobre la que se forma una imagen invertida (negativo) respecto a la existente en el fotolito original. La fuente de luz puede ser interna o un foco externo al aparato.

     Existen insoladoras de una y dos caras, dependiendo del circuito que se desee fabricar, se necesitaría insolar por una cara o ambas caras. En este caso la placa de acople a Arduino Mega es de doble cara, con lo cual se necesita colocar tanto

la cara delantera como la trasera de la placa con el circuito que se ha impreso en papel cebolla (también denominado fotolito).

Se debe asegurar que ambas caras casen de forma perfecta antes del proceso de insolado, de lo contrario, todo el trabajo realizado hasta este punto se echaría a perder.

El proceso de insolado dura alrededor de dos minutos, aunque depende de muchos factores como la cantidad de luz que incide sobre las placas, el tipo de iluminación utilizada (UV, tubos de tungsteno...), para poder calcular de forma exacta el tiempo de insolación se recomienda realizar varias pruebas con placas mas pequeñas y con tiempos de insolado diferentes. Se escogerá el tiempo de insolado de la placa que mejor haya salido del proceso.

**Figura 4.4.0.1 –** Placa de acople introducida en la insoladora

- Una vez extraída la placa de la insoladora, se introduce en liquido revelador, que sacara a la luz el negativo del circuito. Se mantendrá la placa en el liquido hasta que se vea con claridad el circuito, en ese momento se retira.
- Se lava con agua abundante la placa de circuito impreso y se introduce en una solución de cloruro férrico y agua a una temperatura de unos 30 grados centigrados. Este proceso se denomina atacado con ácido. Este proceso es el que mas tiempo consume (alrededor de unos 15 minutos).
- Con acetona se retiran los restos de película fotosensible que queda en la placa para poder soldar correctamente los componentes en el siguiente paso.

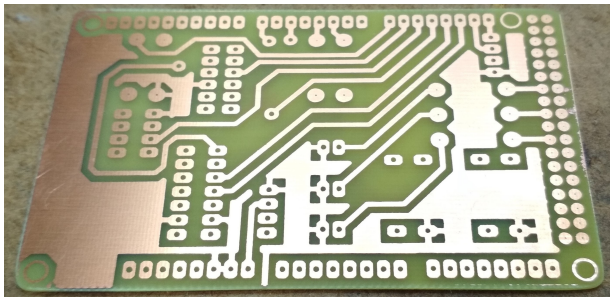


**Figura 4.4.0.2 –** Imagen de la placa de circuito impreso acabada

- Se sueldan los componentes de la placa. Empezando por los componentes que menos espacio abarcan. Una vez soldados todos los componentes, la placa estaría lista para poder introducirle los componentes y módulos de los que dispone.

2. En segundo lugar es necesario realizar las conexiones entre la placa y los dispositivos. En este punto cabe comentar que el motivo de la realización de la placa de acople a Arduino ha sido para facilitar y simplificar el conexionado de nuestro Arduino con los diferentes módulos existentes en el proyecto, esta simplicidad en el montaje ha ahorrado mucho tiempo a la hora de montar y desmontar el circuito en la fase de pruebas y evita posibles cortocircuitos con el dispositivo en movimiento.

   A la placa de acople irán soldados mediante cables de una longitud de aproximadamente unos 5 centímetros los siguientes componentes y placas:

   - Módulo bluetooth
   - Módulo GPRS-GSM
   - Módulo GPS

   En el **[Plano 3]** del documento de los planos se puede ver a qué pines de Arduino MEGA están insertadas las conexiones procedentes de la placa de acople, y en la figura 4.4.0.3 se puede ver el resultado de las conexiones.

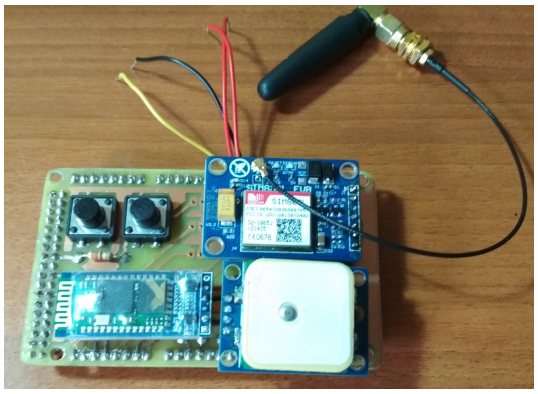


**Figura 4.4.0.3 –** Imagen de la placa de acople a Arduino con los módulos soldados

3. Una vez ejecutados todos los pasos anteriores, se coloca la placa de acople encima de la placa de Arduino MEGA.

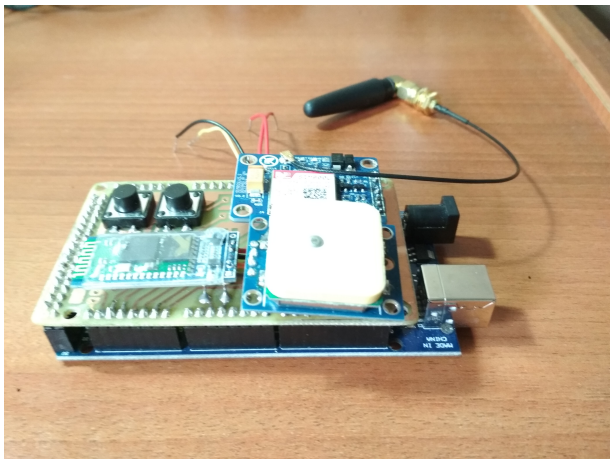


**Figura 4.4.0.4 –** Imagen del sistema de localización implementado

4. El ultimo paso es alimentar el circuito, este paso se explica mejor en el siguiente apartado.

## 4.5. Alimentación del circuito

Para la alimentación del dispositivo se han implementado varias alternativas que se citan a continuación.

En primer lugar, para la alimentación del circuito, se utilizó una batería de litio de 3,7 voltios y 1100mA. Se han utilizado dos módulos elevadores de tensión XL-6009 representados en el **[Plano 8]** del documento de los planos, uno para elevar los 3,7 voltios hasta 7,4 voltios para alimentar Arduino a través del pin VIN y el otro para conseguir 4,8 voltios necesarios para la alimentación del modulo SIM800L. El resto de componentes se alimentaban de los 5 voltios que proporcionaba Arduino a través de su regulador de tensión. En cuanto a la corriente, escasa pero suficiente para hacer funcionar el sistema sin poder realizar ninguna conexión mediante GPRS o envío de E-mails.

En segundo lugar, se utilizó la fuente de alimentación de un ordenador, que posee, entre otras muchas tensiones de salida, 12V y corriente suficiente, dado que dicha fuente es capaz de alimentar un ordenador de sobremesa con un gasto de hasta 500W, lo que ocurría es que el regulador de tensión de Arduino se calentaba un poco, aunque para la realización de todas las pruebas necesarias ha sido suficiente.

El calentamiento del regulador de Arduino se producía debido a que esta alimentado con el límite superior de tensión recomendada por el fabricante para alimentar Arduino, puesto que se recomienda una tensión de entre 6 y 12 V, por tanto esta segunda opción no es muy recomendable para la alimentación del dispositivo durante largos periodos de tiempo.

Por último y la mejor alternativa ha sido utilizar 2 baterías de litio recargables comerciales de 3,7 voltios y 5000mA que, dispuestas en serie, suman 7,4 voltios, tensión suficiente para alimentar Arduino a través del pin VIN (ver figura 4.5.0.1). De esta forma ya sí se dispone de la energía necesaria para afrontar el gasto que conlleva la utilización del modulo SIM800L y sin tener los problemas que se han citado anteriormente. Dos módulos reductores de tensión

(XL4005) que se encuentran en el **[Plano 10]** del documento de los planos serán los encargados de obtener las tensiones de 5 Voltios y de 4,8 voltios, uno para obtener las tensiones de 5 voltios y el otro para obtener la tensión de 4,8 voltios necesaria para el funcionamiento del modulo SIM800L.

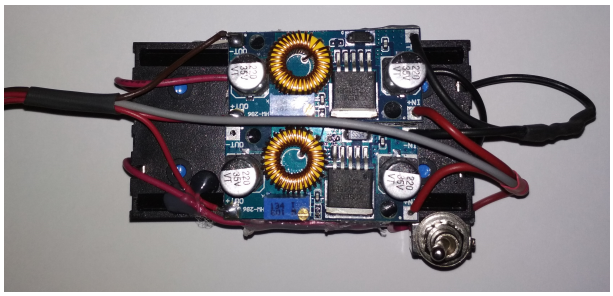
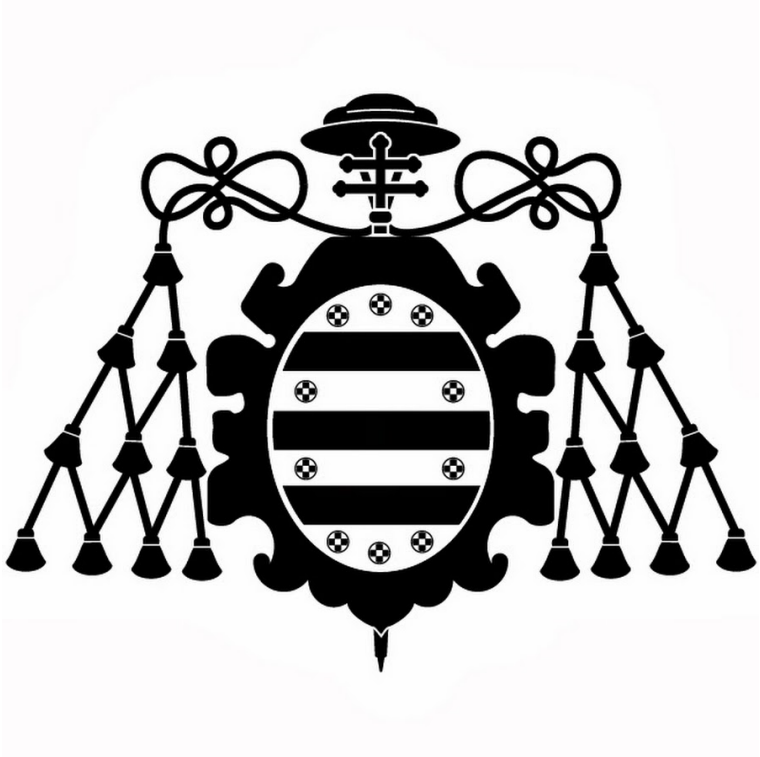


**Figura 4.5.0.1 –** Imagen del circuito de alimentación del sistema de localización

En el **[Plano 1]** del documento de los planos podremos ver el esquema utilizado para alimentar el dispositivo localizador.

La placa de acople que se encuentra en el **[Plano 2]** del documento que contiene los planos dispone de un conector (J8) a través del cual se alimentara Arduino a 7,4 voltios, también dispone de un pin de masa y otros dos pines, uno para subministrar tensión a los módulos GPS y bluetooth y el otro pin se utiliza exclusivamente para alimentar el modulo GPRS-GSM. En el **[Plano 3]** del documento de los planos podremos ver la placa con mas detalle.



# PRESUPUESTO

## Índice del documento PRESUPUESTO



# Índice de figuras

# Índice de tablas

# 1 Presupuesto de componentes electrónicos necesarios

## 1.1. Presupuesto de dispositivos

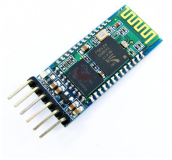
| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | Bluetooth HC-05 |  | 3,02 | 3,02 |
| 2 | Pulsadores tipo botón |  | 0,10 | 0,20 |
| 1 | Módulo SIM 800L |  | 12,05 | 12,05 |
| 1 | Arduino MEGA 2560 |  | 12,00 | 12,00 |
| 2 | DC-DC Adjustable XL4005 Buck Step Down Power Module |  | 1,00 | 2 |
| 2 | Batería Litio 3.7V 5000mAH |  | 2,35 | 4,70 |

**Tabla 1.1.0.1 –** Presupuesto de dispositivos

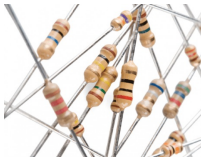
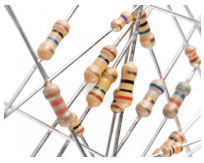
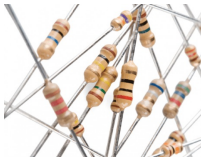
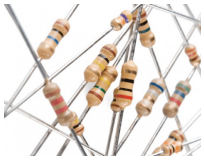
| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | Modulo GPS GYGPS6MV2 |  | 7,70 | 7,70 |
| 1 | Li-ion Rechargeable Support Case with Pin ABS for 18650 3.7V Battery |  | 1,85 | 1,85 |
| 2 | TP4056 + Battery protection LIPO Charger Module Board Micro USB Parts TE420 |  | 1,16 | 2,32 |
| 3 | Resistencias 330 ohm |  | 0,15 | 0,45 |
| 1 | Resistencias 2K ohm |  | 0,15 | 0,15 |
| 1 | Resistencias 1K ohm |  | 0,15 | 0,15 |
| 2 | Resistencias10Kohm |  | 0,15 | 0,30 |
| 3 | Leds |  | 0,30 | 0,90 |
| **SUBTOTAL** | | | | **47,79** |

**Tabla 1.1.0.1 –** Presupuesto de dispositivos

## 1.2. Presupuesto de material para pruebas

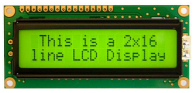
| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | Arduino pro mini Atmega 168 |  | 1,43 | 1,43 |
| 1 | Módulo SIM 800L |  | 9,93 | 9,93 |
| 1 | Modulo SIM900A |  | 10,00 | 10,00 |
| 1 | Modulo GPS GYGPS6MV2 |  | 6,79 | 6,79 |
| 1 | Display LCD 16X2 |  | 1,40 | 1,40 |
| 1 | Interfaz i2c para lcd |  | 1,00 | 1,00 |
| 1 | Adaptador serie FTDI FT232RL |  | 1,53 | 1,53 |
| 1 | TP4056 + Battery protection LIPO Charger Module Board Micro USB Parts TE420 |  | 1,16 | 1,16 |

**Tabla 1.2.0.1 –** Presupuesto de material para pruebas

| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | Expansor de bus I2C PCF8574P DIP-16 |  | 1,55 | 1,55 |
| 2 | RT8272-BOOST-CONVERTER |  | 1,08 | 2,16 |
| 1 | Teclado hexadecimal 4x4 |  | 1,64 | 1,64 |
| 2 | XL6009 DC Adjustable Step Up Down Boost Power |  | 1,22 | 2,22 |
| 1 | 100 Pcs 1P 2.54mm Pitch Dupont Jumper Wire Cable Housing Female Pin Connector |  | 1,56 | 1,56 |
| 1 | 100 Pcs Female Pin Connector Terminal 2.54mm Pitch for dupont jumper wire cable |  | 1,00 | 1,00 |
| 1 | 20 Pcs 2.54mm 40 Pins Single Row Solder Male & Female Straight Header For Arduino |  | 1,58 | 1,58 |

**Tabla 1.2.0.1 –** Presupuesto de material para pruebas

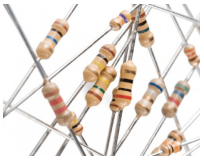
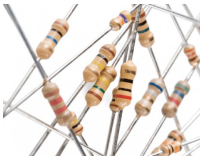
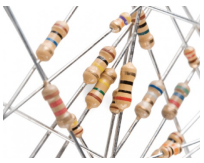
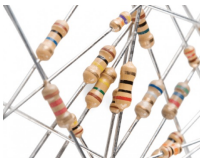
| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | 40 PCS 20cm 2.54mm Female to Female Dupont Wire Cable |  | 1,67 | 1,67 |
| 1 | 40 PCS 20cm 2.54mm Male to Male Dupont Wire Cable |  | 1,78 | 1,78 |
| 1 | 40 PCS 20cm 2.54mm Male to Female Dupont Wire Cable |  | 1,78 | 1,78 |
| 1 | Placa de prototipos |  | 15,00 | 15,00 |
| 1 | Batería Litio 3.7V 1100mAH | | 6,64 | 6,64 |
| 10 | Resistencias 1K ohm |  | 0,15 | 1,50 |
| 10 | Resistencias 2K ohm |  | 0,15 | 1,50 |
| 10 | Resistencias 330 ohm |  | 0,15 | 1,50 |
| 10 | Resistencias10Konhm |  | 0,15 | 1,50 |

**Tabla 1.2.0.1 –** Presupuesto de material para pruebas

| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 10 | Transistor 2N222 |  | 0,10 | 1,00 |
| 10 | Leds |  | 0,30 | 3,00 |
| **SUBTOTAL** | | | | **81,82** |

**Tabla 1.2.0.1 –** Presupuesto de material para pruebas

## 1.3. Presupuesto del servidor(almacenamiento de datos)

| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | RASPBERRY-PI RPI2-MODDBV1.2 SBC, RASPBERRY PI 2 MODEL B V1.2 |  | 37,98 | 37,98 |
| 1 | Dual Band 802.11 Wireless Adapter Wifi LAN Adaptor 300Mbps USB Dongle 5GHz FT5 |  | 1,90 | 1,90 |
| 1 | Cable ethernet 5m |  | 3,95 | 3,95 |
| 1 | Fuente de alimentación para Raspberry PI 2 (5V 2A) |  | 2,75 | 2,75 |

**Tabla 1.3.0.1 –** Presupuesto del servidor

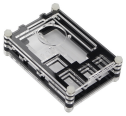
| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | Tarjeta SAMSUNG microSD EVO (con Adaptador SD) 16GB - Clase 10 - 48MB/s |   | 6,75 | 6,75 |
| 1 | 12V Cooler Axial Fan 40x40x10mm Arduino Raspberry Pi 3D Printer Computer |  | 1,00 | 1,00 |
| 1 | Carcasa Protectora Raspberry pi 2 |  | 4,06 | 4,06 |
| 1 | LCD Raspberry pi 7” |  | 62,50 | 62,50 |
| **SUBTOTAL** | | | | **120,89** |

**Tabla 1.3.0.1 –** Presupuesto del servidor

## 1.4. Presupuesto de la placa de acople a Arduino MEGA

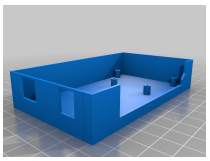
| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | Tira de 40 pines macho 2.54mm Arduino |  | 1,58 | 1,58 |
| 1 | Placa de doble cara |  | 3,50 | 3,50 |
| 1 | Estaño |  | 0,50 | 0,50 |
| 1 | Cables |  | 2,00 | 2,00 |
| 1 | Acople plastico |  | 2,00 | 2,00 |
| 11 | Tornillos allen M3 |  | 0,05 | 0,50 |
| 11 | Tuercas M3 |  | 0,04 | 0,40 |
| **SUBTOTAL** | | | | **10,47** |

**Tabla 1.4.0.1 –** Presupuesto de la placa de acople a Arduino

# 2 Presupuesto de material de montaje

| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | Taladro proxon |  | 70,00 | 70,00 |
| 1 | Soporte proxon |  | 60,00 | 60,00 |
| 1 | Tenazas electrónicas |  | 5,00 | 5,00 |
| 1 | Alicates electrónicos |  | 5,00 | 5,00 |
| 1 | Polímetro |  | 20,00 | 20,00 |
| 1 | Soldador |  | 30,00 | 30,00 |
| 1 | Estaño |  | 5,00 | 5,00 |
| 1 | Impresora 3D |  | 400,00 | 400,00 |

**Tabla 2.0.0.1 –** Presupuesto de material de montaje

| Ud | Concepto | Imagen | Precio unit (€) | Total (€) |
|---|---|---|---|---|
| 1 | Cable |  | 3,00 | 3,00 |
| 100 | Conector Hembra |  | 0,02 | 2,00 |
| 100 | Terminal Macho |  | 0,015 | 1,50 |
| 100 | Terminal Hembra |  | 0,02 | 2,00 |
| 1 | Cable mini USB a USB |  | 2,00 | 2,00 |
| **SUBTOTAL** | | | | **605,50** |

**Tabla 2.0.0.1 –** Presupuesto de material de montaje

No es imprescindible para la realización del proyecto el disponer de una impresora 3D, de un taladro para electrónica o incluso de un polímetro. Estos componentes nos han servido para mejorar el acabado (impresora 3D), realización de pruebas y medición de las conexiones (polímetro), realización de agujeros en la placa del teclado (taladro). El resto de componentes si son imprescindibles para la elaboración del proyecto, acentuando sobre todo la utilización del cable y los terminales y conectores indicados en la tabla 1.2.0.1 que nos facilitan el conexionado entre el Arduino y los diferentes dispositivos que se han utilizado para la elaboración del proyecto.

# 3 Presupuesto de Recursos humanos

| Cantidad (horas) | Concepto | Precio (€)/hora | Total |
|---|---|---|---|
| 300 | Ingeniería | 60 | 18000 |
| 7 | Montaje | 40 | 280 |
| 30 | Horas de tutoración | 90 | 2700 |
| **SUBTOTAL** | | | 20980,00 |

**Tabla 3.0.0.1 –** Presupuesto de recursos humanos

# 4 Presupuesto final

| Capítulos | % | Importe |
|---|---|---|
| **Presupuesto de componentes electrónicos necesarios [1]** | | |
| Componentes electrónicos necesarios | 0,22 | 47,79 |
| Placa de acople | 0,05 | 10,47 |
| Material para pruebas | 0,37 | 81,82 |
| Servidor | 0,55 | 120,89 |
| **Presupuesto del material de montaje [2]** | | |
| Material de montaje | 2,77 | 605,50 |
| **Presupuesto de recursos humanos [3]** | | |
| Ingeniería | 82,39 | 18000,00 |
| Montaje | 1,28 | 280,00 |
| Horas de tutoracion | 12,36 | 2700,00 |
| **Total presupuesto de ejecución de material** | | 21846,47 |
| **Gastos generales (13%)** | | 2840,04 |
| **Beneficio industrial (6%)** | | 1310,79 |
| **Suma** | | 25997,30 |
| **IVA (21%)** | | 5459,43 |
| **TOTAL PRESUPUESTO** | | 31456,73 |

**Tabla 4.0.0.1 –** Presupuesto final

El presupuesto total del proyecto asciende a la cantidad de: ***Treinta y un mil cuatrocientos cincuenta y seis euros y setenta y tres céntimos*** (31456,73€)