industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Dispositivo electrónico para el estudio de cobertura de redes LPWAN

TRABAJO FIN DE GRADO

GRADO EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Pedro Martínez García**
Director: Juan Suardíaz Muro
Codirector: Jesús Rubio Aparicio

Cartagena, octubre de 2018

Universidad Politécnica de Cartagena

Para mis padres, por toda la ayuda y apoyo durante todos estos años.
Agradecido es poco.

ÍNDICE

CAPÍTULO 1

INTRODUCCIÓN

1.1. Propósito y justificación del proyecto

El presente proyecto consiste en saber si es posible dotar de conectividad LPWAN (Low Power Wide-Area Network) a dispositivos que requieran un enlace de datos y si la cobertura donde va a estar nuestro dispositivo es suficiente para poder establecer la comunicación. Debido a este problema surge este dispositivo que permite transmitir tramas a través de la red de forma periódica, recibir el nivel de señal devuelto y ver instantáneamente la trama de radio de la red usada y demás datos útiles; determinando así si es posible establecer la comunicación entre dos dispositivos con este tipo de red.

1.2. Objetivos del proyecto

El objetivo general del proyecto es crear un dispositivo electrónico capaz de establecer comunicación con redes operativas utilizando el protocolo Sigfox.

Sigfox es una red de baja velocidad permite que todo objeto pueda conectarse a Internet, de forma asequible y con un bajo consumo energético. Además, es realmente útil para la validación de aplicaciones como sensores de red, rastreo, edificios inteligentes, seguridad, mediciones o M2M. Dado que ya se encuentran en el mercado dispositivos similares mi objetivo será intentar realizar un prototipo con datos que el otro dispositivo no aporta. Datos como puede ser la relación señal/ruido (en inglés, Signal to noise ratio), conseguir mandar y recibir datos pudiendo elegir la frecuencia y la potencia de la señal de envío o la cobertura en dB y dando más opciones de uso como establecer que a cada cierta distancia obtengamos la señal y conseguirlo de la forma más

económica y con el menor tamaño posible. Para que nuestro dispositivo sea capaz de realizar estas mediciones de cobertura y conectarse a la red constará de los siguientes dispositivos conectados entre sí como se muestra en el esquema:

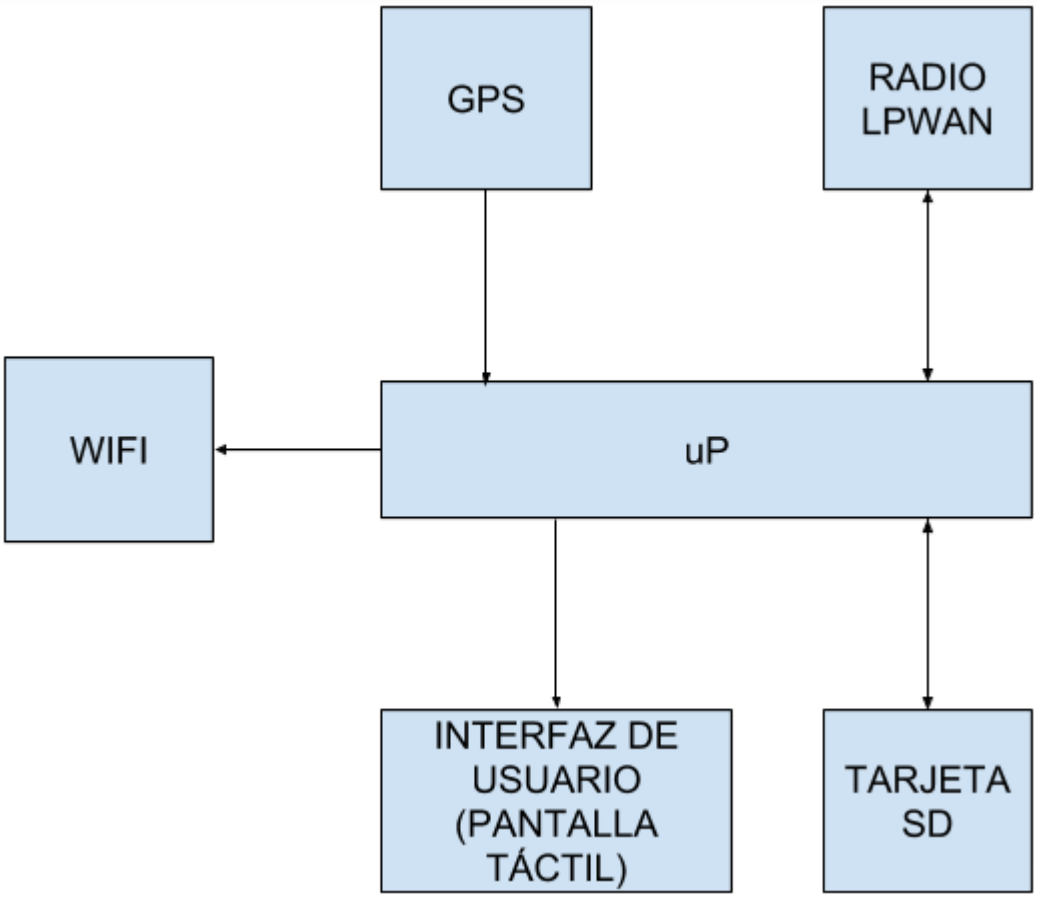

Ilustración 1. Diagrama de bloques del funcionamiento del prototipo

El objetivo de este proyecto es la creación de un dispositivo capaz de recopilar los datos de cobertura de las redes LPWAN, los almacene y los envíe a internet junto a la posición GPS. Mostrando los datos a través de una pantalla y una interface de usuario. Para conseguir este objetivo general se prevén los siguientes objetivos específicos:

- Analizar requerimientos de comunicación.
- Selección de un dispositivo de bajo coste adecuado.
- Desarrollo de la programación de la interfaz.

1.3. Requisitos hardware

1.3.1. Interconectar y configurar cada uno de los módulos hardware necesarios

Para llevar a cabo el proyecto se necesita un módulo Radio LPWAN para enviar tramas a través de la red de forma periódica y recoger el nivel de señal devuelto, comprobando la cobertura de la red. Tarjeta SD para almacenar el dato junto con la posición GPS para asegurar el correcto posicionamiento de nuestros dispositivos, información que se presentará al usuario en pantalla de forma que se pueda ver toda la información de funcionamiento. A continuación, se hará el conexionado y la programación de cada uno de los componentes individualmente con una protoboard (placa de pruebas), de manera que se pueda realizar cualquier cambio o mejora en cualquier momento, asegurando su correcto funcionamiento. Cuando el conexionado y la programación de cada uno de los componentes sea el deseado se realizará el esquemático de una shield.

El último paso será procesar todos los datos obtenidos de la red para poder presentarlos al usuario en pantalla (interface de usuario) de la forma más sencilla e intuitiva posible. Además, se estudiará la posibilidad de utilizar una app móvil para la interface de usuario.

1.3.2. Diseñar una shield para Arduino con esta funcionalidad

Con el software de diseño Diptrace se realizará el diseño del esquemático de una shield que contenga los componentes hardware anteriormente mencionados y con el que se pasará a PCB de forma que pueda ser impresa la placa. Dicho shield deberá ser compatible con el dispositivo microcontrolado que haya sido seleccionado para la mejor resolución del proyecto tanto técnica como económicamente.

1.4. Requisitos software

La implementación software para la radio LPWAN de forma que sea capaz de enviar y recibir tramas de la red de Sigfox. Intentando conseguir la máxima información de esta que sea posible. Intentando, además, que sea lo más configurable posible, es decir, que se puedan modificar

parámetros como frecuencia y potencia de transmisión mediante la interfaz de usuario, sin necesidad de conectarlo al ordenador.

Se debe implementar una interfaz de usuario lo más sencilla e intuitiva posible y que muestre al usuario toda la información de cobertura de la red de Sigfox recogida por la radio LPWAN, acompañada de la posición (latitud y longitud) obtenida por el GPS. Para que esto sea posible el GPS deberá de buscar y actualizar la posición en la que se encuentra el dispositivo cada un mínimo intervalo de tiempo, de forma que el dispositivo almacene con exactitud donde ha sido recibida la trama.

Además, debe almacenar toda esta información de una forma ordenada en la memoria SD. De esta manera se podrá hacer un seguimiento del dispositivo y del nivel de señal devuelto en cada punto en el que se haya enviado una trama.

1.5. Planificación

Para el correcto desarrollo del proyecto es necesaria una gran planificación. Para ello, se dividirá en etapas y en metas a alcanzar. Estas metas se deberán ir completando en el plazo de tiempo estimado. Para ello se ha realizado el diagrama de Gantt.

	Nombre	Duracion	Inicio	Terminado
1	**1. Dispositivo electrónico capaz de adquirir**	98 days	**23/10/17 8:00**	**7/03/18 17:00**
2	**1.1 Documentación**	12 days	**23/10/17 8:00**	**7/11/17 17:00**
3	1.1.1 Estado del arte	5 days	23/10/17 8:00	27/10/17 17:00
4	1.1.2 Especificaciones del producto	3 days	30/10/17 8:00	1/11/17 17:00
5	1.1.3 Estudio del mercado	4 days	2/11/17 8:00	7/11/17 17:00
6	**1.2 Implementación**	35 days	**8/11/17 8:00**	**26/12/17 17:00**
7	1.2.1 Compra de los componentes	5 days	8/11/17 8:00	14/11/17 17:00
8	1.2.2 Conexionado y programación del GPS	5 days	15/11/17 8:00	21/11/17 17:00
9	1.2.3 Conexionado y programación Radio LPW	5 days	22/11/17 8:00	28/11/17 17:00
10	1.2.4 Conexionado y programación pantalla	5 days	29/11/17 8:00	5/12/17 17:00
11	1.2.5 Programación interface de usuario	10 days	6/12/17 8:00	19/12/17 17:00
12	1.2.6 Almacenamiento en tarjeta SD	5 days	20/12/17 8:00	26/12/17 17:00
13	**1.3 Desarrollo de la shield**	27 days	**27/12/17 8:00**	**1/02/18 17:00**
14	1.3.1 Desarrollo del esquemático	15 days	27/12/17 8:00	16/01/18 17:00
15	1.3.2 Conversión PCB	3 days	17/01/18 8:00	19/01/18 17:00
16	1.3.3 Impresión PCB	8 days	22/01/18 8:00	31/01/18 17:00
17	1.3.4 Montaje del dispositivo	1 day	1/02/18 8:00	1/02/18 17:00
18	**1.4 Testeo del dispositivo**	19 days	**2/02/18 8:00**	**28/02/18 17:00**
19	1.4.1 Toma de datos	9 days	2/02/18 8:00	14/02/18 17:00
20	1.4.2 Correccion de posibles errores	10 days	15/02/18 8:00	28/02/18 17:00
21	**1.5 Memoria**	98 days	**23/10/17 8:00**	**7/03/18 17:00**
22	1.5.1 Elaboración de la memoria	93 days	23/10/17 8:00	28/02/18 17:00
23	1.5.2 Elaboración del PowerPoint	6 days	28/02/18 8:00	7/03/18 17:00

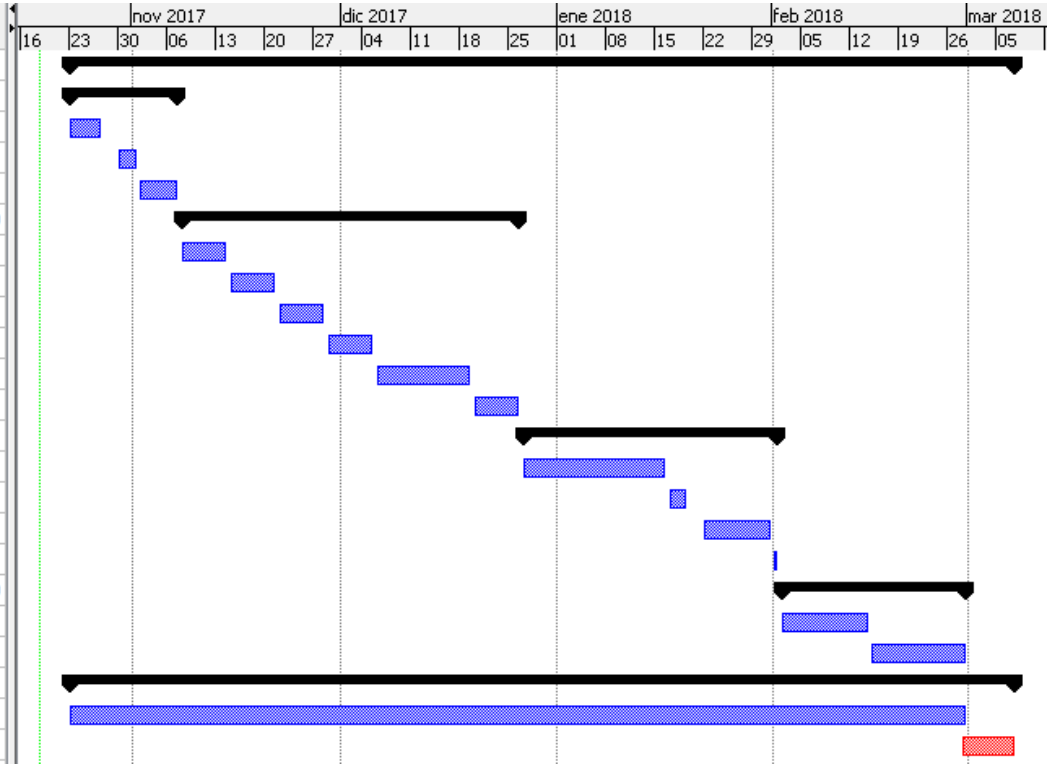

Ilustración 2. Diagrama de Gantt

CAPÍTULO 2

ESTADO DEL ARTE

En este segundo capítulo se tratará sobre el estado del arte del mundo del IoT, así como un estudio de mercado, sobre los aspectos más importantes que engloban a éste.

Para comenzar, se hará una introducción a las redes LPWAN, su funcionamiento, sus posibles aplicaciones, así como su diferencia con otras redes como pueden ser las móviles y sus ventajas e inconvenientes con respecto a estas.

Se hará un repaso de los dispositivos que se encuentran en el mercado de mediciones de cobertura de las redes LPWAN. Así como sus especificaciones técnicas como físicas.

Posteriormente, se estudiarán los componentes electrónicos que se encuentran en el mercado para llevar a cabo el prototipo. Se explicará qué componentes se han seleccionado, así como sus características técnicas y físicas. Seleccionando los que mejor se ajustan a nuestro proyecto valorando tanto la sencillez de implementación hardware y software, como su robustez y precio.

2.1. Introducción a las redes LPWAN

Sigfox es una solución de conectividad celular mundial para el Internet of Things pensada para comunicaciones de baja velocidad que permite reducir los precios y el consumo de energía para los dispositivos conectados. La solución de conectividad SIGFOX se basa en una infraestructura de antenas y de estaciones de base totalmente independientes de las redes existentes.

Sigfox desata el potencial del IoT proporcionando una red mundial para objetos conectados altamente escalable, los dispositivos SIGFOX Ready™ se conectan a Internet sin costes suplementarios ligados a la situación geográfica y sin configuración de red para una ubicación específica.

Esta solución de conectividad mundial es administrada por Sigfox gracias a su programa de colaboración con los operadores de redes móviles, conectando los ecosistemas locales a las redes mundiales. A diferencia de los operadores de telefonía intentando aumentar el ancho de banda a 2/3/4G para dar servicio a móviles y tablets, el objetivo principal de Sigfox consiste en permitir que todo objeto pueda conectarse a Internet, de forma asequible**,** dondequiera que se encuentre y sin necesitar de recargar su batería.

Es imposible imaginar el despliegue de billones de dispositivos conectados con unos costes de comunicación tan elevados, y considerando semejantes volúmenes, cada centímetro cuadrado de microchip cuenta. El protocolo SIGFOX es compatible con la mayoría de transceptores existentes, y por ende es fácilmente accesible para los fabricantes de módulos y dispositivos.

A nivel tecnológico Sigfox utiliza la UNB (Ultra Narrow Band), basada en una tecnología de radio, para conectar dispositivos a su red mundial. La utilización de la UNB es esencial para suministrar una red de alta capacidad, evolutiva, de muy bajo consumo energético, conservando una infraestructura celular fácil de implantar.

Sigfox es una solución de conectividad que se centra en los dispositivos de baja velocidad. Con Sigfox se puede enviar entre 0 y 140 mensajes por día y cada mensaje puede contener hasta 12 bytes de datos reales de carga útil. El protocolo ya transmite el ID del dispositivo, de modo que los 12 bytes representan la carga útil real y no existe ningún límite sobre cómo estructurar estos 12 bytes.

Sigfox además ofrece un API y un CLOUD SIGFOX que ofrece una interfaz de aplicación web para la gestión de dispositivos y la configuración de integración de datos, así como APIs web estándares para automatizar la gestión de los dispositivos e implementar la integración de datos.

2.2. Estado del arte de dispositivos de prueba de campo de las redes LPWAN

Como ya se ha mencionado, ya se encuentran en el mercado dispositivos similares. En este apartado se va a desarrollar con más profundidad que dispositivos se pueden encontrar y qué características poseen.

En el mercado se encuentran Dispositivos de Prueba de Campo (en inglés, Field Test Device, abreviado FTD) como el *Field Test Device Sigfox* y el *Field Test Device LoRaWAN* de la empresa Adeunis. Encontrando además diferencias entre dispositivos en función de la banda de frecuencias dependiendo de si nos encontramos en USA (915MHz) o en Europa (868MHz). El

dispositivo de prueba de campo de Adeunis es un dispositivo compatible con Sigfox o LoRa que permite comunicarse con todas las redes mediante estos protocolos. El sistema hace posible transmitir y recibir tramas de radio y visualizar instantáneamente los resultados. Equipado con una pantalla LCD, se puede visualizar la información relacionada con el funcionamiento de la red (Uplink, Downlink, PER, etc.) además de la información de los sensores (coordenadas GPS, temperatura, nivel de batería, etc. Gracias a su batería recargable, el FTD permite varias horas de funcionamiento y se puede recargar utilizando un micro-usb.

- Las especificaciones técnicas de este dispositivo son:

Radio	
Comunicación	Protocolo Sigfox y DBPSK Sigfox
Velocidad radio	Variable (SF12/125kHz (~183 bps) a FSK (~50kbps)
Frecuencia	Banda ISM863-870MHz
Potencia RF	14dBm (25mW)
Sensibilidad	Hasta -140 dBm en SF 12/CR4
Rango (abierto)	Hasta 15km
Estándares	EN 300-220, EN 301-489, EN 60950
Network area	RC1
Alimentación	
Conector	Micro-USB - 5V - 500mA
Batería	Ion-Litio Polímero 3.7V 2Ah 7.4Wh
Operacional	
Configuración del dispositivo	Comandos AT
Velocidad del serial	115.2 kbps
Paridad	Ninguno
Cantidad de datos	8
Stop bit	1
Temperatura de funcionamiento	-30°C / +70°C

Tabla 1. Especificaciones técnicas del Field Test Device

- Las especificaciones físicas son:

Características mecánicas	
Dimensiones	186.20 x 75.20 x 22.80
Peso	140g

Tabla 2. Especificaciones físicas del Field Test Device

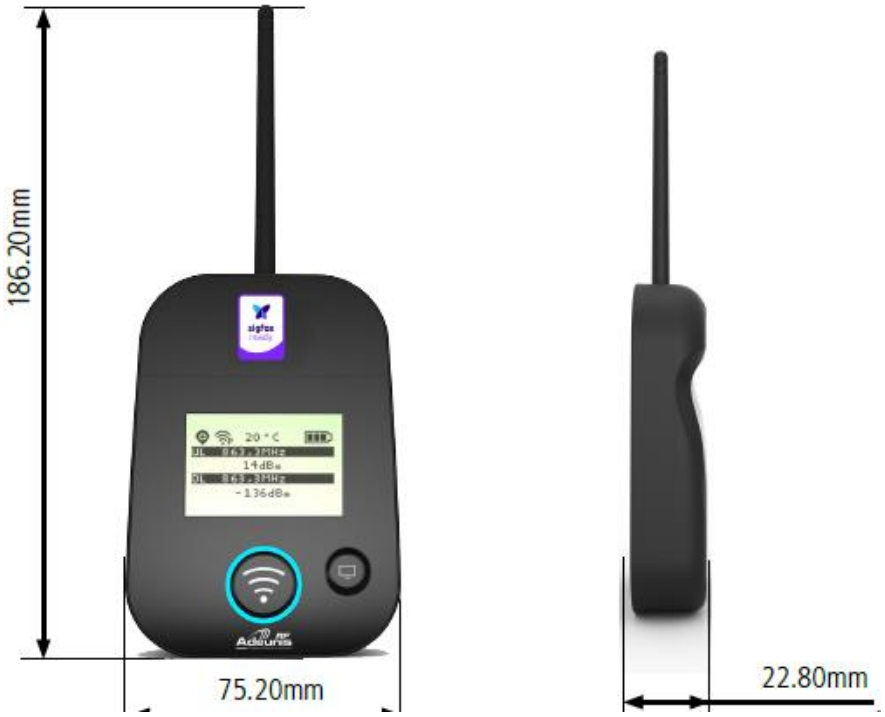

Ilustración 3. Adeunis Field Test Device

2.3.Estudio de mercado

Para llevar a cabo el desarrollo del producto se ha hecho un estudio de mercado de los diferentes dispositivos microcontrolados, seleccionando los que menos componentes complementarios necesite, abaratando la solución y simplificando la resolución. Se ha realizado el estudio de la familia Arduino, Raspberry y SiPy.

2.3.1. Familia Arduino

Arduino es una plataforma electrónica abierta que permite controlar todo tipo de sistemas, ya que cuenta tanto con un software como con un hardware flexible. Arduino fue creado bajo la visión de lo que se conoce como open hardware. Permite a través del ordenador, mediante programación, que el usuario logre interactuar con circuitos electrónicos y controlarlos por software. De igual forma el Arduino es capaz de actuar de manera autónoma sin estar conectado a un ordenador. Existen múltiples modelos de Arduino con diferentes características. Cada modelo posee un nombre, formas, capacidades y funciones distintas. Por tanto, se analizarán las placas de Arduino más relevantes para poder seleccionar el que más se ajusta a nuestro dispositivo.

- Arduino UNO

En primer lugar, se encuentra el Arduino Uno, esta placa se ha convertido en la más estandarizada de la plataforma Arduino, su ajustado precio unido a la gran capacidad de ampliación que ofrece mediante shields, la ha convertido en la más utilizada.

Ilustración 4. Arduino UNO

- Arduino MEGA

Tiene una serie de características más avanzadas que Arduino UNO. El Arduino Mega 2560 es una placa basada en el microcontrolador ATmega2560. Tiene 54 pines digitales de entrada / salida (de los cuales 15 se pueden usar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP, y un botón de reinicio. El MEGA 2560 está diseñado para proyectos más complejos.

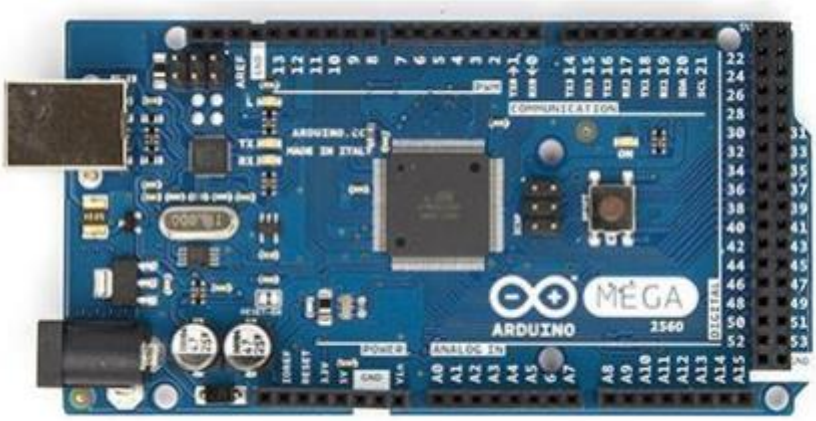

Ilustración 5. Arduino MEGA

- Arduino YUN

El Arduino Yún es una placa basada en el microprocesador ATmega32u4 y el Atheros AR9331. La placa tiene soporte integrado Ethernet y WiFi, un puerto USB-A y ranura para tarjeta micro-SD. El Yún se distingue de otras placas Arduino por su capacidad para comunicarse con Linux, ofreciendo una poderosa computadora en red con la facilidad de un Arduino.

Ilustración 6. Arduino YUN

- Arduino MKR FOX 1200

Ha sido diseñado para ofrecer una solución práctica y rentable para los fabricantes que buscan agregar conectividad Sigfox a sus proyectos. Está basado en el módulo Atmel SAMD21 y ATA8520 SigFox.

Ilustración 7. Arduino MKRFOX 1200

Debido a la amplia gama de dispositivos que tiene Arduino se ha optado por mencionar tan solo los más relevantes y que mejor se ajustan a una posible solución para nuestro dispositivo. Sin mencionar aquellos que, a pesar de tener un menor coste, no se ajustan por su menor potencia, memoria, cantidad de pines digitales I/O (entrada/salida) y no tener suficientes UART.

2.3.2. SiPy

El SiPy es un microcontrolador de pycom compatible con MicroPython. Esta placa posee una radio Sigfox, ideal para aplicaciones de IoT y el chip de WiFi ESP32. Su precio es de 35€ y la placa de expansión de 16€.

Ilustración 8. SiPy

2.3.1. Raspberry Pi

Raspberry Pi es una placa computadora (SBC) de bajo costo. Como Arduino, Raspberry Pi es un proyecto open hardware y open source, pero a diferencia de éste se trata de un dispositivo que cuenta con un microprocesador en vez de un microcontrolador. Cuyas diferencias radican en

que un microprocesador es un dispositivo electrónico que necesita de todos los periféricos para poder funcionar correctamente, es decir, tener una tarjeta madre como soporte (con todos los buses que necesite el microprocesador: bus de dirección, datos, control, etc), tener también el banco de memoria tanto RAM como ROM y más. Pudiendo hacer cualquier función que se le ordene dependiendo del software que se le implemente. En un microcontrolador, internamente ya están implementados todos los buses, el banco de memoria, reloj, temporizadores, etc; además necesita de un software que lo gobierne, pero este software es único y tiene una sola función. Como se puede observar se trata de una buena alternativa a Arduino, con una serie de características que lo hacen idóneo para determinadas situaciones, su precio ajustado unido a su conectividad a Internet vía Ethernet, suponen unos añadidos muy valiosos a la hora de su elección. La Raspberry Pi tiene un precio de unos 37€.

Ilustración 9. Raspberry Pi

2.3.2. Elección del dispositivo microcontrolado

Como se ha podido observar existe una amplia diversidad de microcontroladores que se pueden utilizar en el proyecto. Se han analizado los dispositivos más relevantes, pero hay muchas más alternativas de las descritas anteriormente. Sin embargo, en este caso se ha optado por crearlo mediante el uso de los microcontroladores que ofrece Arduino, más adelante se detallará cual. El principal motivo de esta elección son las grandísimas facilidades que ofrece Arduino, con una gran capacidad de expansión mediante shields y una relación calidad/precio muy alta. Además de una gran facilidad de programación gracias a la gran cantidad de librerías que existen al ser una plataforma basada en open-source.

Tras el análisis de los modelos más relevantes de la plataforma Arduino, se ha realizado una tabla comparativa de sus características. A continuación, se comparan los parámetros de mayor importancia a la hora de obtener un prototipo eficiente y con viabilidad:

	UNO	MEGA	NANO	MINI	LEO	YÚN	MKR FOX 1200
Microcontrolador	ATmega328	ATmega2560	ATmega328		ATmega32u4	ATmega32u4 +AR9331	Atmel SAMD21 +ATA8520
Pines Digitales I/O	14 (6 PWM)	54 (15 PWM)	14 (6 PWM)		20 (7 PWM)		8
Pines Analógicos	6	16	8		12		8
Memoria flash (KB)	31,5	256	30		32		256
UART	1	4	1			1 (no disponible)	1
SPI	Sí						
I2C	Sí						
Precio (€)	20,00	35,00	20,00	14,00	18,00	60,00	35

Tabla 3. Comparativa de la familia Arduino

El Arduino elegido es Arduino Mega. Esta elección se debe a su bajo coste, su mayor número de pines de entrada/salida con 4 puertos serie frente a 1 puerto serie del resto de Arduinos, necesarios para la conexión de varios shields mediante UART, no necesitando, por tanto, placas de expansión y abaratando así el prototipo. Teniendo, además, un microprocesador más potente y más memoria, necesarios para poder implementar el código necesario para todos los shields. A continuación, se va a explicar en mayor profundidad las especificaciones técnicas, mecánicas y eléctricas de Arduino MEGA.

- **El pinout de Arduino MEGA es:**

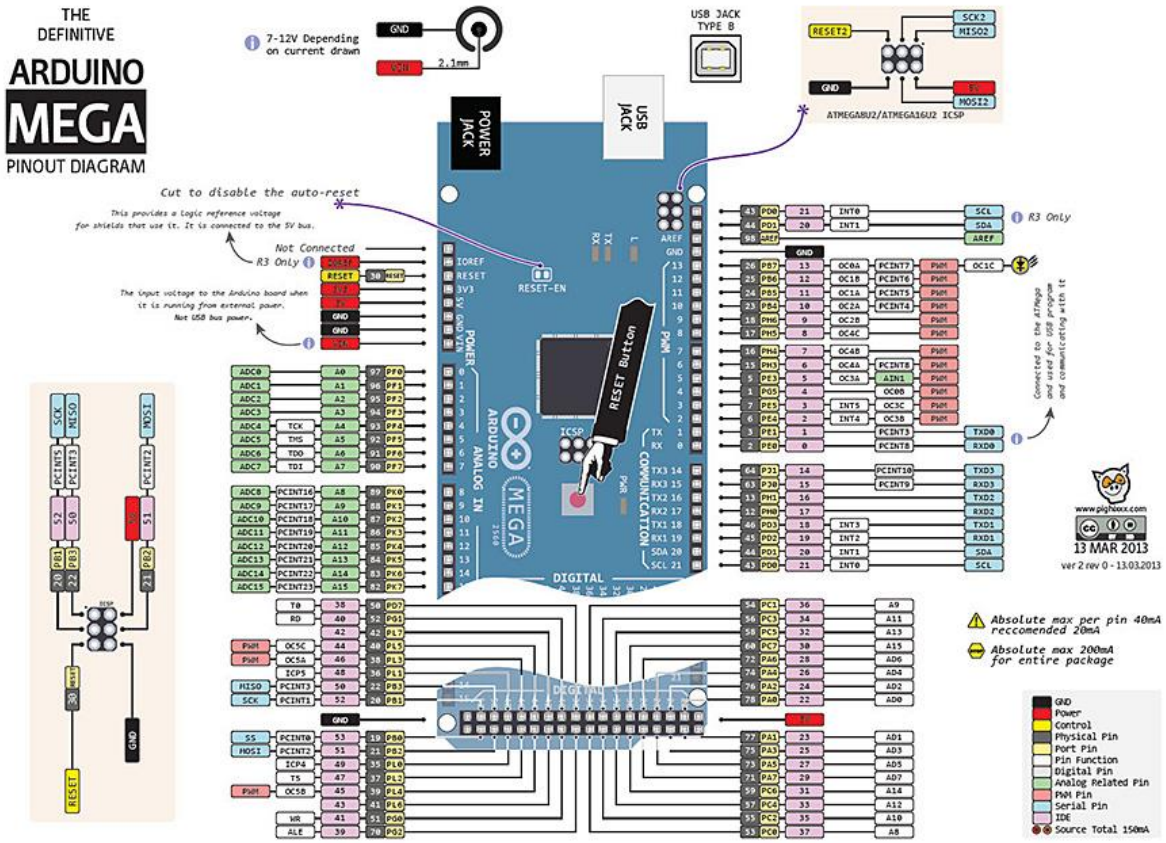

Ilustración 10. Pinout de Arduino MEGA

- **El esquemático de Arduino MEGA es:**

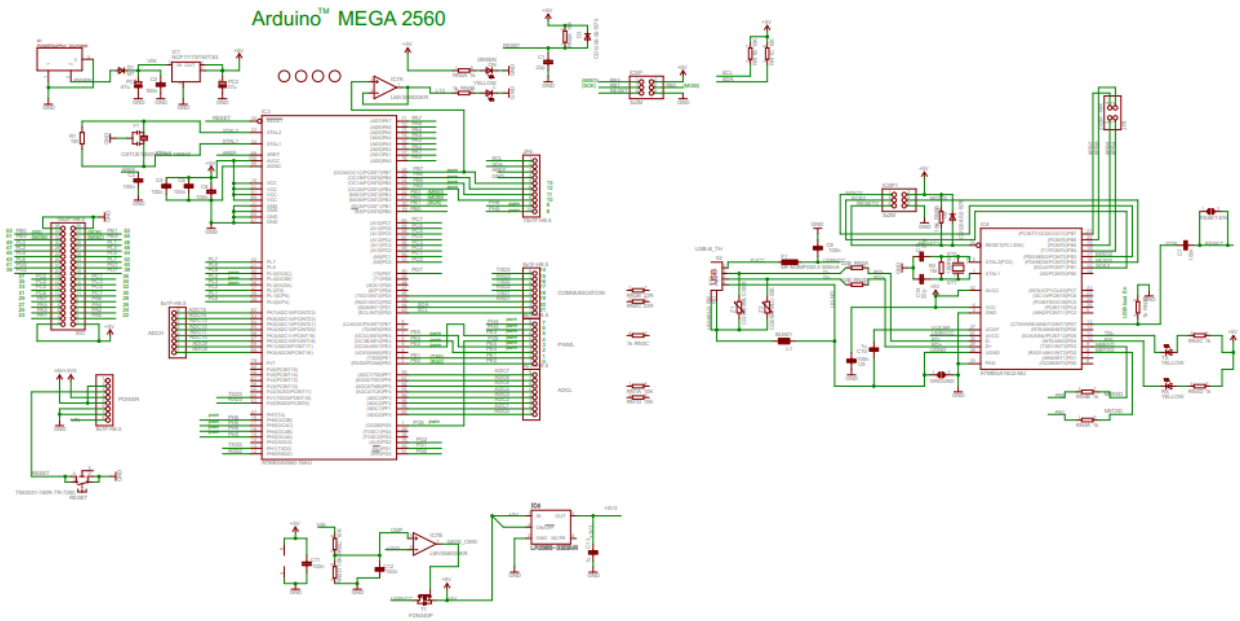

Ilustración 11. Esquemático de Arduino MEGA

- **Alimentación**

El Mega 2560 puede alimentarse a través de la conexión USB o con una fuente de alimentación externa. La fuente se selecciona automáticamente. La alimentación externa (sin USB) puede provenir de un adaptador AC/DC o de una batería. El adaptador se puede conectar enchufando un conector positivo de 2,1 mm (power jack) en el conector de alimentación de la placa. Los cables de una batería se pueden insertar en los pines GND y Vin del conector POWER.

La placa puede operar con alimentación externa de 6 a 20 voltios. Sin embargo, si se suministra con menos de 7 V, el pin de 5 V puede suministrar menos de cinco voltios y la placa puede volverse inestable. Si usa más de 12 V, el regulador de voltaje puede sobrecalentarse y dañar la placa. El rango recomendado es de 7 a 12 voltios. Los pines de alimentación son los siguientes:

- Vin. El voltaje de entrada a la placa cuando está usando una fuente de alimentación externa (a diferencia de 5 voltios de la conexión USB u otra fuente de alimentación regulada). Se puede suministrar voltaje a través de este pin o, si suministra voltaje a través del jack de alimentación, se puede acceder a través de este pin.
- 5V. Este pin produce 5 V regulados desde el regulador de la placa. La placa se puede alimentar con la toma de alimentación de DC (7-12 V), el conector USB (5 V) o el pin VIN de la placa (7-12 V). El suministro de voltaje a través de los pines de 5V o 3,3V evita el regulador y puede dañar la placa.
- 3V3. Se consigue una fuente de 3,3 V generada por el regulador que incorpora Arduino. El consumo máximo de corriente es de 50 mA.
- GND. Pines de tierra
- IOREF. Este pin en la placa proporciona la referencia de voltaje con la que opera el microcontrolador. Un shield correctamente configurado puede leer el voltaje del pin IOREF y seleccionar la fuente de alimentación apropiada o habilitar traductores de voltaje en las salidas para trabajar con 5V o 3,3V.

- **Memoria**

El ATmega2560 tiene 256 KB de memoria flash para almacenar el código (de los cuales 8 KB se utilizan para el gestor de arranque), 8 KB de SRAM y 4 KB de EEPROM (que se pueden leer y escribir con la librería EEPROM).

- **Entradas y salidas**

Cada uno de los 54 pines digitales del Mega se puede usar como entrada o como salida, usando las funciones pinMode(), digitalWrite() y digitalRead(). Operan a 5 voltios. Cada pin puede proporcionar o recibir 20 mA según las condiciones de funcionamiento recomendadas y tiene una resistencia interna de pull-up (desconectada por defecto) de 20-50 KΩ. Para evitar daños permanentes en el microcontrolador no deben excederse los 40 mA.

Además, algunos pines tienen funciones especiales:

- Serial: 0 (RX) y 1 (TX); Serial 1: 19 (RX) y 18 (TX); Serial 2: 17 (RX) y 16 (TX); Serial 3: 15 (RX) y 14 (TX). Se usa para recibir (RX) y transmitir (TX) datos en serie TTL. Los pines 0 y 1 también están conectados a los pines correspondientes del chip serie ATmega16U2 USB a TTL.
- Interrupciones externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3) y 21 (interrupción 2). Estos pines se pueden configurar para activar una interrupción en un nivel bajo, flanco ascendente o descendente, o un cambio en el nivel.
- PWM: Los pines del 2 al 13 y del 44 al 46. Proporcionan salidas PWM de 8 bits con la función analogWrite().
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Estos pines soportan la comunicación SPI utilizando la librería SPI.
- LED: 13. Hay un LED integrado conectado al pin digital 13. Cuando el pin tiene un nivel alto, el LED está encendido, cuando el pin tiene nivel bajo, está apagado.
- TWI: 20 (SDA) y 21 (SCL). Soporte de comunicación TWI utilizando la librería Wire.
- Arduino Mega 2560 tiene 16 entradas analógicas, cada una de las cuales proporciona 10 bits de resolución (es decir, 1024 valores diferentes). Por defecto, miden desde tierra a 5 voltios, aunque es posible cambiar el extremo superior de su rango usando el pin AREF y la función analogReference().
- AREF. Voltaje de referencia para las entradas analógicas. Usado con analogReference().
- Reiniciar. Poner esta línea a LOW para reiniciar el microcontrolador. Normalmente se usa para agregar un botón de reinicio para los shields que obstaculizan el que está en la placa.

- **Comunicación**

La placa Mega 2560 tiene un gran número de opciones para comunicarse con un ordenador, otra placa u otros microcontroladores. El ATmega2560 proporciona cuatro UART de hardware para

comunicación serie TTL (5V). La placa canaliza uno de estos a través de USB y proporciona un puerto virtual para el software en la computadora. El software Arduino (IDE) incluye un monitor serie que permite que se envíen datos de texto simples hacia y desde la placa. Los LED RX y TX de la placa parpadearán cuando los datos se transmitan a través del USB a la computadora (pero no para comunicación serial en los pines 0 y 1).

- **Características físicas y compatibilidad**

La longitud y el ancho máximos de la PCB Mega 2560 son de 4 y 2,1 pulgadas respectivamente, con el conector USB y el conector de alimentación extendiéndose más allá de la dimensión anterior. Tres orificios para tornillos permiten que la placa se una a una superficie o caja. Tenga en cuenta que la distancia entre los pines digitales 7 y 8 es de 160 mil (0.16 "), no es un múltiplo par del espaciado de 100 mil de los otros pines.

El Mega 2560 está diseñado para ser compatible con la mayoría de los shields diseñados para el Uno. Los pines digitales de 0 a 13 (y los pines adyacentes AREF y GND), las entradas analógicas de 0 a 5, el encabezado de alimentación y el encabezado ICSP están todos en ubicaciones equivalentes. Además, el UART principal (puerto serie) está ubicado en los mismos pines (0 y 1), al igual que las interrupciones externas 0 y 1 (pines 2 y 3, respectivamente). SPI está disponible a través del encabezado ICSP.

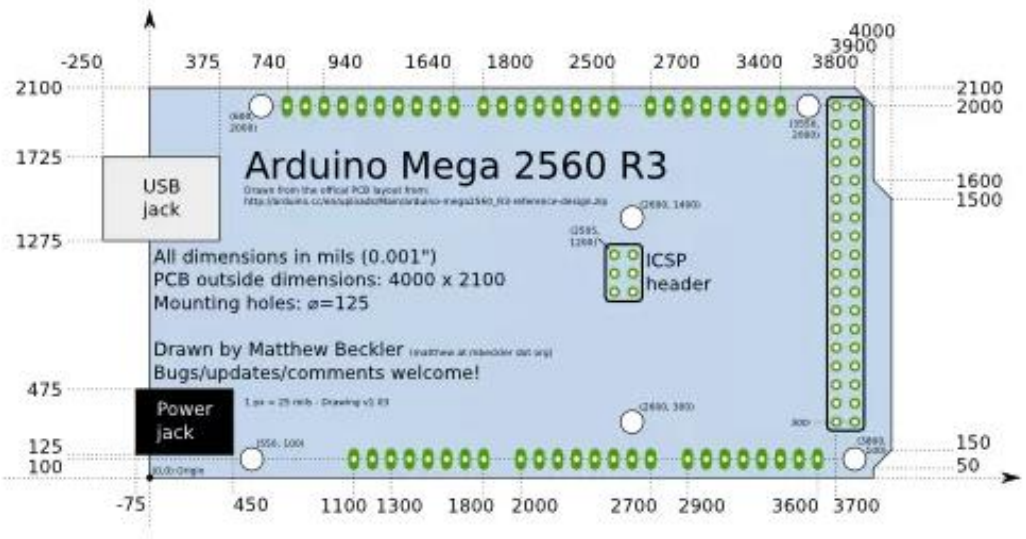

Ilustración 12. Dimensiones de Arduino MEGA

- **Restablecimiento automático (software)**

En lugar de requerir una pulsación física del botón de reinicio antes de cargar el nuevo software, el Mega 2560 está diseñado de forma tal que permite su reinicio mediante un software que se ejecuta en una computadora conectada. Una de las líneas de control de flujo de hardware (DTR) del ATmega8U2 está conectada a la línea de restablecimiento del ATmega2560 a través de un condensador de 100 nanofaradios. Cuando esta línea se pone a nivel bajo, la línea de reinicio cae lo suficiente como para restablecer el chip. El software Arduino (IDE) usa esta capacidad para permitirle cargar código simplemente presionando el botón de carga en el entorno Arduino. Esto significa que el gestor de arranque puede tener un tiempo de espera más corto, ya que la disminución de DTR puede coordinarse bien con el inicio de la carga.

Esta configuración tiene otras implicaciones. Cuando la placa Mega 2560 está conectada a una computadora que ejecuta Mac OS X o Linux, se restablece cada vez que se realiza una conexión desde el software (a través de USB). Durante el siguiente medio segundo más o menos, el gestor de arranque se está ejecutando en el ATMega2560. Mientras está programado para ignorar datos mal formados (es decir, cualquier cosa además de una carga de código nuevo), interceptará los primeros bytes de datos enviados a la placa después de que se abra una conexión. Si un boceto que se ejecuta en la placa recibe una configuración de una sola vez u otros datos cuando se inicia por primera vez, asegúrese de que el software con el que se comunica espera un segundo después de abrir la conexión y antes de enviar esta información.

La placa Mega 2560 contiene un jumper que se puede cortar para desactivar el reinicio automático. Los pads a cada lado de la pista pueden soldarse juntas para volver a habilitarlo. Está etiquetado como "RESET-EN". También puede desactivar el restablecimiento automático conectando una resistencia de 110 ohmios desde 5 V a la línea de reinicio.

CAPÍTULO 3

IMPLEMENTACIÓN DE LA ARQUITECTURA HARDWARE

En este capítulo se explicarán los dispositivos que componen la parte relativa al hardware del sistema. Para el nodo central se ha escogido Arduino Mega. A continuación, se detallarán los componentes necesarios, sus características, así como su conexionado con Arduino. Además, se detallará la función que van a desempeñar.

3.1. Componentes

3.1.1. Pantalla táctil

Para la interface gráfica se ha optado por una pantalla táctil resistiva con ranura microSD incorporada, lo que le da un valor añadido a la hora de su elección ya que es imprescindible el uso de una para almacenar la información. De esta forma no se necesitará un shield adicional para la microSD. A continuación, se va a proceder a la descripción detallada de sus características.

- **Alimentación y comunicación, configuración y compatibilidad**

La pantalla trabaja a 5 V por lo que se podrá conectar directamente con Arduino. Se conectará a través de SPI, con 5 pines, SCK, MISO, MOSI y dos SS (uno para la TFT y otro para el STMPE610). Los pines SCK, MISO y MOSI están compartidos tanto para el TFT, como para el STMPE610 y la microSD. Además, son necesarios, uno adicional para controlar la microSD y otro usado por Arduino para decidir si quiere enviar datos o comandos. Esto hace un total de 7 pines necesarios para controlar tanto la pantalla como la microSD.

Si se usa Arduino MEGA la pantalla tiene unos jumper que permiten cambiar la configuración. Así, cortando los jumper del 11-13 y soldando los del ICSP, los pines de comunicación SPI se obtienen de las salidas del ICSP del ATmega2560 de Arduino MEGA. Los pines que se necesitarán son:

Arduino UNO		Arduino MEGA
Pin 13 SCK (SPI) Pin 12 MISO (SPI) Pin 11 MOSI (SPI) Pin 10 SS/CS (TFT Chip Select) Pin 9 TFT DC (Data/command select) Pin 8 SS/CS (STEMP610 Chip Select) Pin 4 uSD	Cortando y soldando los jumper SMD →	Pin 66 SCK (SPI) Pin 64 MISO (SPI) Pin 67 MOSI (SPI) Pin 10 SS/CS (TFT Chip Select) Pin 9 TFT DC (Data/command select) Pin 8 SS/CS (STEMP610 Chip Select) Pin 4 uSD

Tabla 4. Conexión de la pantalla con Arduino UNO y con MEGA

- **Características físicas**

Esta TFT tiene 2,8” (unos 71 cm aproximadamente) de diagonal, 240x320 píxeles, 4 LED de retroiluminación y 18 bit que le da 262000 tonos diferentes.

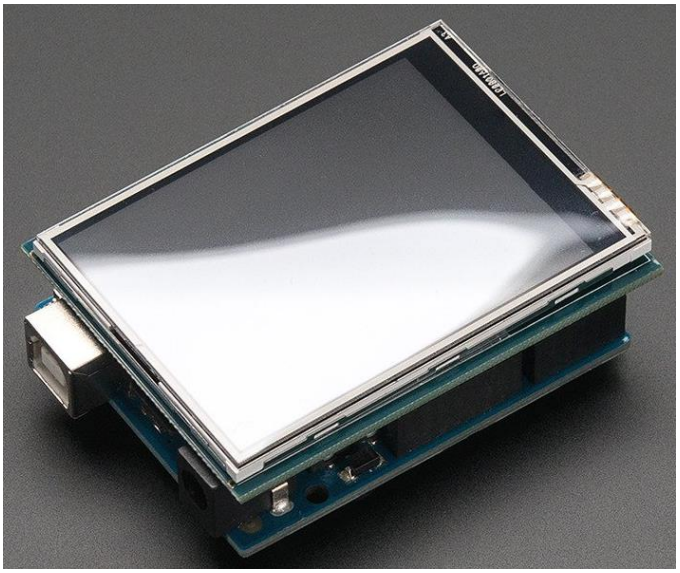

Ilustración 13. Pantalla táctil I

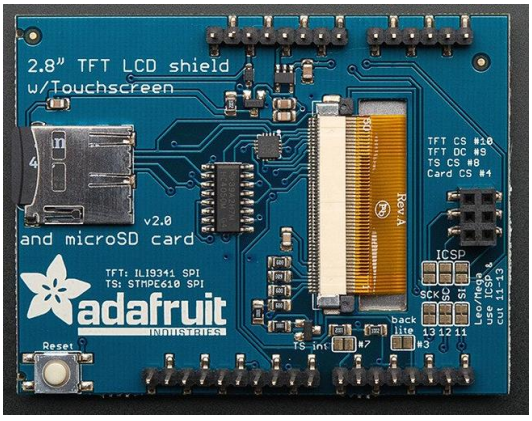

Ilustración 14. Pantalla táctil II

3.1.2. Módulo GPS NEO6MV2

Módulo basado en el IC NEO6MV2 con capacidad para decodificar las señales de los satélites GPS (Global Position System).

- **Alimentación y comunicación**

Se alimenta a 5V de Arduino Este módulo se conectará a través de UART con el puerto serie 1 de Arduino (Tx1 y Rx1). Tiene 4 pines, dos de ellos para la alimentación, Vcc y GND, yotros dos para la comunicación serie con Arduino, Rx y Tx.

GPS	Arduino MEGA
1 GND	GND
2 Tx	19 Rx1
3 Rx	18 Tx1
4 Vcc	5 V

Tabla 5. Pines de conexión del GPS con Arduino MEGA

- **Características físicas**

Este módulo tiene un tamaño de 25 x 35 mm y 25x 25 mm la antena.

Ilustración 15. Módulo GPS NEO6MV2

3.1.3. Módulo WIFI ESP8266

Este es un módulo transceptor serie WiFi, basado en el SoC (System on a Chip) ESP8266. Es un chip altamente integrado diseñado para conectarse a una red WiFi a través del protocolo TCP/IP que tiene integrado.

- **Alimentación y comunicación**

Se alimenta a 3,3 V y se conectará a través de UART con el puerto serie 2 de Arduino (Tx2 y Rx2). Como en el caso de la radio, nos encontramos con la problemática de que este funciona 3,3 V y no a los 5 V de los pines de Arduino. Si se alimenta a 3,3 V y los pines de comunicación se conectan a los 5 V de los pines de comunicación de Arduino funcionará, pero acortará la vida útil del componente, por lo que se ha optado por usar un elevador lógico de tensión que luego se integrará en el diseño de la PCB. Tiene 8 pines, cuyo conexionado es:

WiFi	Arduino MEGA
1 GND	GND
2 GPIO2	Desconectado
3 GPIO0	Desconectado
4 Rx	16 Tx2
5 Tx	17 Rx2
6 CH PD (chip enable)	3,3 V
7 RST	Desconectado
8 Vcc	3,3 V

Tabla 6. Pines de conexión del módulo WiFi con Arduino MEGA

Los pines de GPIO (General Purpose Input/Output) se utilizan para actualizar el firmware.

- **Características físicas**

Este módulo tiene un tamaño de 21,1 x 13,2 mm.

Ilustración 16. Módulo WiFi ESP8266

3.1.4. microSD

Al llevar la pantalla una ranura para uSD no se necesitará comprar un shield adicional para esta. En la uSD se almacenará toda la información relevante que se obtiene de la radio, acompañada de la posición (latitud y longitud) en la que se ha obtenido dicha información con la hora y la fecha. Además, se almacenarán las imágenes y todo aquel material que se necesite para la interfaz de usuario de la pantalla.

3.1.5. Conversor lógico de niveles

Para poder conectar los dispositivos de 3,3V a un sistema de 5V como es Arduino se ha usado un conversor lógico de nivel bidireccional. Es un dispositivo pequeño que baja con seguridad las señales de 5V a 3,3V y aumenta 3,3V a 5V al mismo tiempo. Este convertidor de nivel también funciona con dispositivos de 2,8V y 1,8V. Cada convertidor de nivel tiene la capacidad de convertir 4 pines en el lado alto a 4 pines en el lado bajo con dos entradas y dos salidas provistas para cada lado. Para su funcionamiento la placa necesita ser alimentada desde las dos fuentes de voltaje, alto voltaje (5 V) al pin 'HV', bajo voltaje (3,3 V) a 'LV', y tierra desde el sistema al pin 'GND'. En este caso se utilizará para convertir todos los pines GPIO y de comunicación serie de Arduino, de 5 a 3,3 V, en todos aquellos dispositivos que necesiten ser alimentados a 3,3 V.

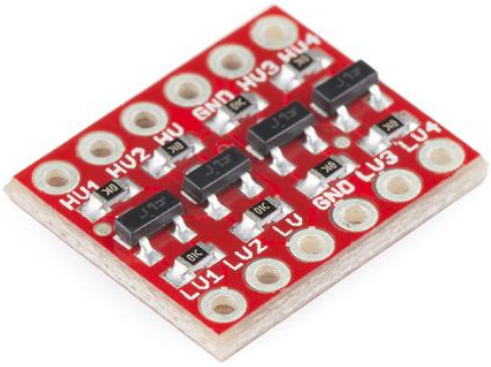

Ilustración 17. Elevador lógico de tensión

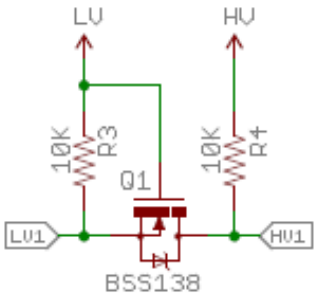

Ilustración 18. Esquemático de una salida del elevador lógico de tensión

A la hora de hacer el diseño de la PCB se integrará esta placa. Por lo que además de los shields descritos anteriormente se necesitarán 12 resistencias de 10 KΩ y 6 BSS138. Los BSS138 son transistores MOSFET SMD (surface-mount device) de canal N cuyo funcionamiento interno es el siguiente:

Ilustración 19. Transistor BSS138

3.1.6. Radio LPWAN

El XM001-EU es una familia de módulos inalámbricos de largo alcance y baja potencia que operan en banda ISM 868MHz. Con su gran robustez y el bajo consumo de energía, es la mejor solución para aplicaciones que requieren un largo alcance, una duración máxima de la batería y un enlace de radio seguro. Este módulo integra el módulo MM002-EU en un formato XBEE estándar. Esta antena es dual, es decir, es capaz de conectarse tanto con la red de Sigfox como con la de LoRa.

Ilustración 20. Módulo Radio LPWAN

- **Alimentación y comunicación**

Este módulo se alimenta a 3,3 V y se comunica a través de UART con el puerto serie 3 de Arduino (Tx3 y Rx3). Para poder comunicarse con Arduino necesitará un adaptador, ya que funciona a 3,3 V, el adaptador convierte los 5 V con los que trabaja Arduino a los 3,3 V. Para la puesta en marcha de este módulo se necesitan, además de los pines de alimentación y los de comunicación, dos pines digitales de Arduino para dar un pulso (flanco de bajada) al pin de reset (pin 5) y de wake up (pin 12) del módulo.

Radio	Arduino MEGA
1 Vcc	3,3 V
2 Tx	15 Rx3
3 Rx	14 Tx3
4	Desconectado
5 RESET	2
6-9	Desconectado
10 GND	GND
11	Desconectado
12 WAKE UP	3
13-20	Desconectado

Tabla 7. Pines de conexionado del módulo de radio con Arduino MEGA

Con el software integrado que usa el modo de ahorro de energía, el pin de reactivación debe usarse para reactivar el módulo antes de cualquier comunicación a través de UART. El cronograma de reactivación es el siguiente:

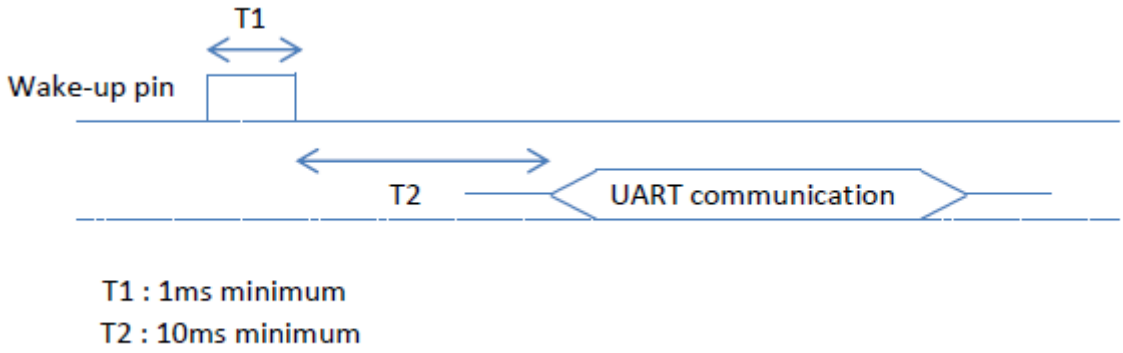

Ilustración 21. Cronograma de activación del módulo de radio

Al realizar el diseño del PCB se tendrá esto en cuenta y no será necesario el adaptador de niveles, que se ha utilizado para probar el módulo por separado.

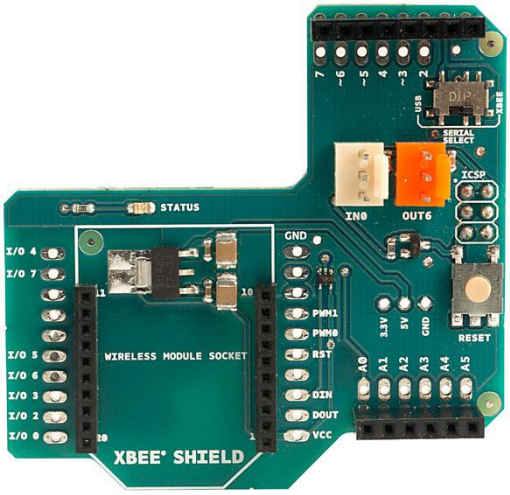

Ilustración 22. Adaptador radio formato XBEE

3.2. Conexionado

A continuación, se va a proceder a describir con más detalle y con diagramas ilustrativos el conexionado de cada uno de los shields utilizados en este proyecto.

3.2.1. Diagrama de bloques

Tras haber estudiado los esquemáticos y los datasheets de los componentes para saber su correcto funcionamiento se hace un diagrama de bloques del conexionado. Determinando gráficamente la cantidad de pines necesarios y cómo se hará la transmisión de datos.

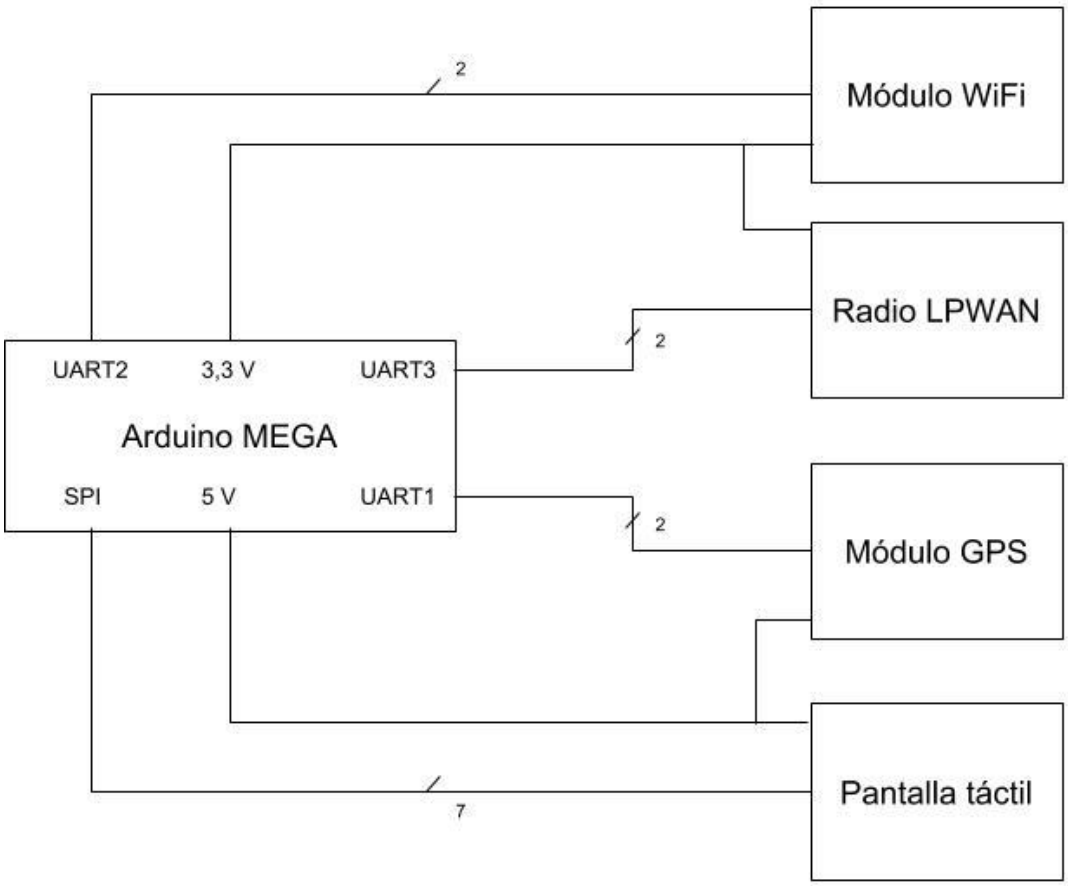

Ilustración 23. Diagrama de bloques del conexionado

3.2.2. Conexiones con Arduino MEGA

Antes de realizar el conexionado para probar individualmente cada uno de los módulos se hará un diagrama más detallado. De esta forma se evitará la rotura de los componentes y se evitarán fallos en el esquemático del PCB. Hay que tener en cuenta que en algunos de los módulos las conexiones serie irán a 3,3 V y no se hace una conexión directa a los pines de Arduino. Los pines de Arduino trabajan a 5 V por lo que en algunos casos se necesitará un conversor lógico de nivel 5-3,3 V. Por lo tanto, este diagrama es solo una representación de cómo se conectará teniendo en cuenta lo dicho anteriormente.

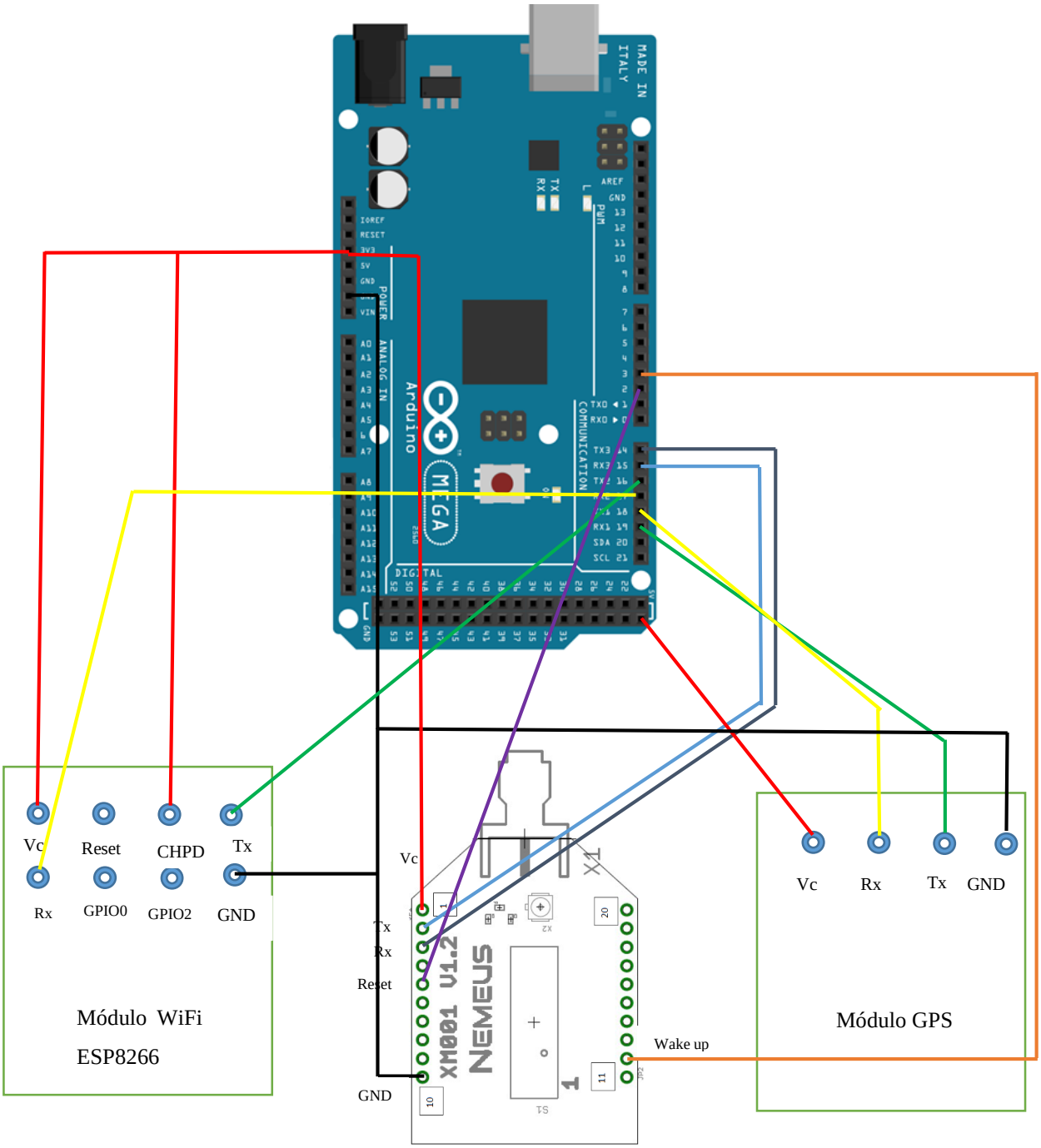

Ilustración 24. Diagrama de conexionado

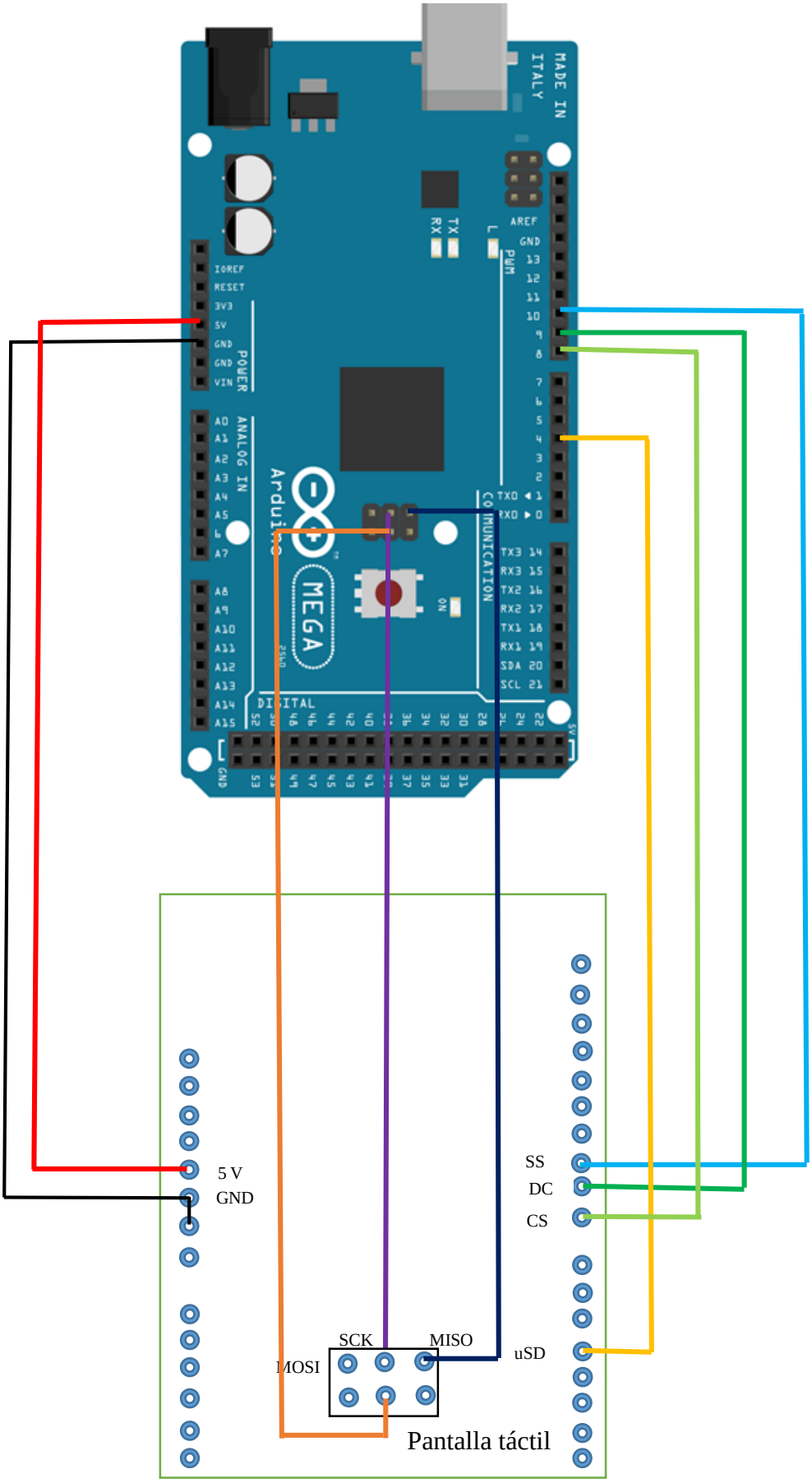

Ilustración 25. Diagrama de conexionado de la pantalla

Para probar el funcionamiento de todos los shields a la vez es necesario hacer una placa que conecte entre sí todos los módulos y Arduino MEGA. Esta PCB será una placa adaptadora, que actuará como intermediaria, tendrá dimensiones parecidas a Arduino MEGA y se pinchará en él, así mismo los módulos se soldarán en esta PCB, facilitando la conexión de todos los módulos sin necesidad de una protoboard y cableado. El programa utilizado para hacer el diseño de la PCB es DipTrace y los pasos a seguir son los siguientes:

3.3. Creación de los shields en DipTrace

Un componente regular en DipTrace consiste en un símbolo esquemático, un patrón o huella (footprint) y, posiblemente, un modelo 3D. Los tres representan la misma entidad, pero en diferentes etapas del diseño: esquemático, diseño de PCB y visualización/exportación 3D, respectivamente. Para crear un nuevo componente, siempre es mejor comenzar dibujando una huella PCB, que se asigna a un cierto símbolo esquemático en el Component Editor. En este caso como no había ningún patrón disponible en las librerías, se ha tenido que realizar la huella de cada uno de los componentes de la PCB en el Pattern Editor, para luego ir al Component Editor y dibujar el símbolo esquemático, adjuntar ese patrón existente y guardar el componente completo en la biblioteca de componentes.

3.3.1. Pattern Editor

En el Pattern Editor se dibujará la huella que es la forma y el tamaño que ocupa y la disposición de los pads utilizados para unir físicamente y conectar eléctricamente un componente a una placa de circuito impreso. Todos los footprints se han obtenido de los datasheets de los componentes. La huella debe ser lo más exacta posible, de otra forma los componentes no se ajustarán a las huellas, pudiendo faltar espacio de forma que no será posible soldar los componentes (no cabrán), o sobrar, aumentando el tamaño de la placa innecesariamente.

3.2.1.1 Arduino MEGA

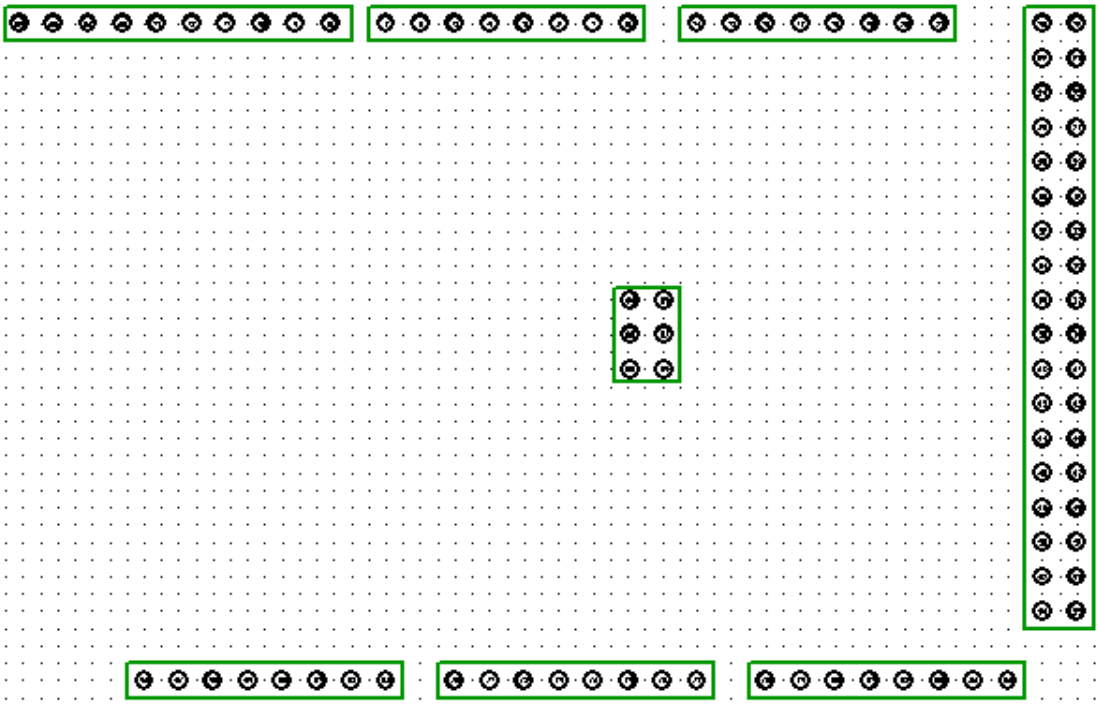

Ilustración 26. Arduino MEGA en DipTrace

3.2.1.2 Pantalla táctil:

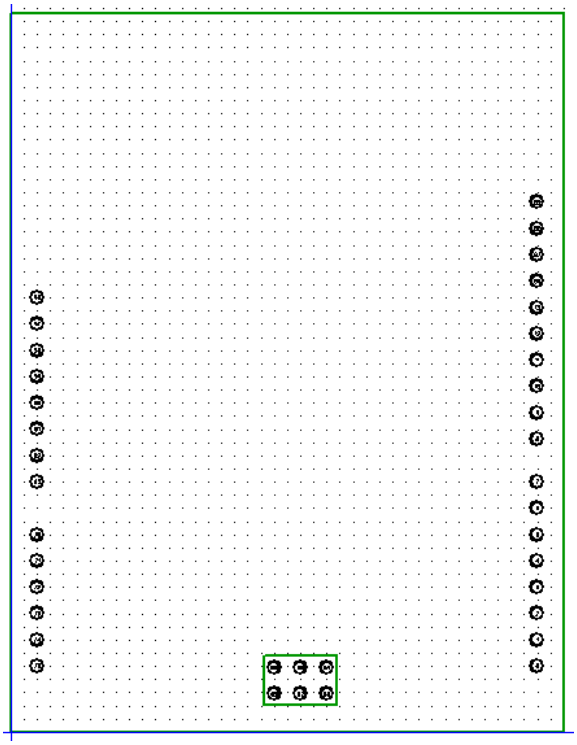

Ilustración 27. Pantalla táctil en DipTrace

3.2.1.3 Módulo GPS NEO6MV2

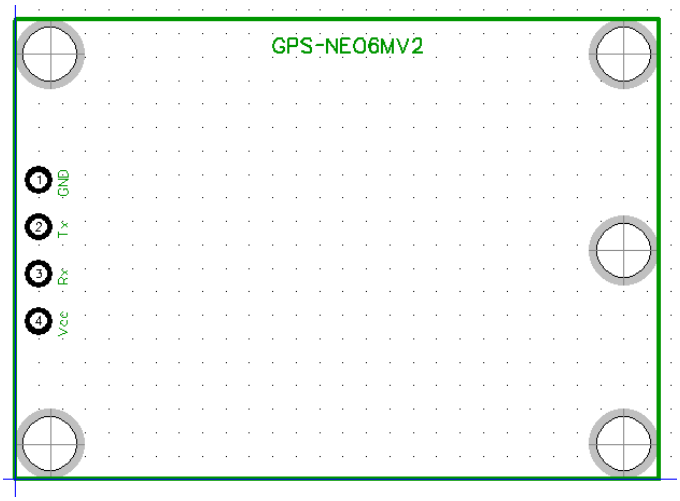

Ilustración 28. Módulo GPS en DipTrace

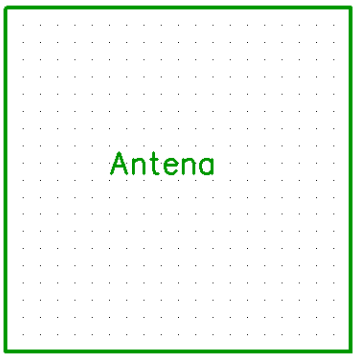

Ilustración 29. Antena del GPS en DipTrace

3.2.1.4 Módulo WIFI ESP8266:

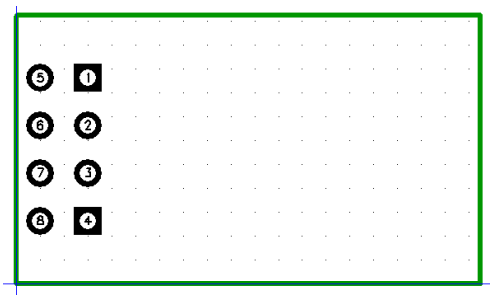

Ilustración 30. Módulo WiFi en DipTrace

3.2.1.5 BSS138

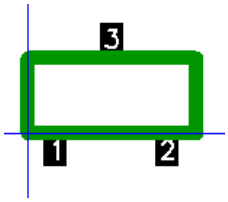

Ilustración 31. Transistor BSS138 en DipTrace

3.2.1.6 Radio LPWAN

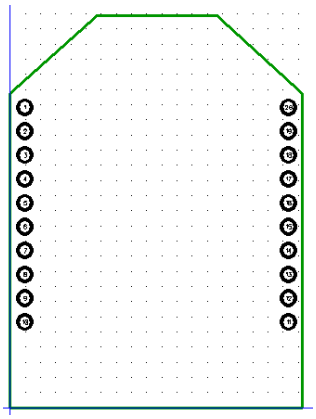

Ilustración 32. Módulo Radio LPWAN en DipTrace

3.3.2. Component Editor

Al contrario que en el Pattern Editor, para el Component Editor no es necesaria la precisión de forma y tamaño. Lo único a tener en cuenta será el número de pines y su función, ya que al tener un patrón adjunto si no coinciden el conexionado en la PCB no se hará correctamente. Por esta razón no se mostrarán individualmente cada uno de los componentes como en el caso anterior, aunque se podrán observar todos los componentes en el esquemático.

3.4. Esquemático

En el esquemático se realiza el conexionado de todos los módulos entre sí y del que luego se obtendrá la PCB.

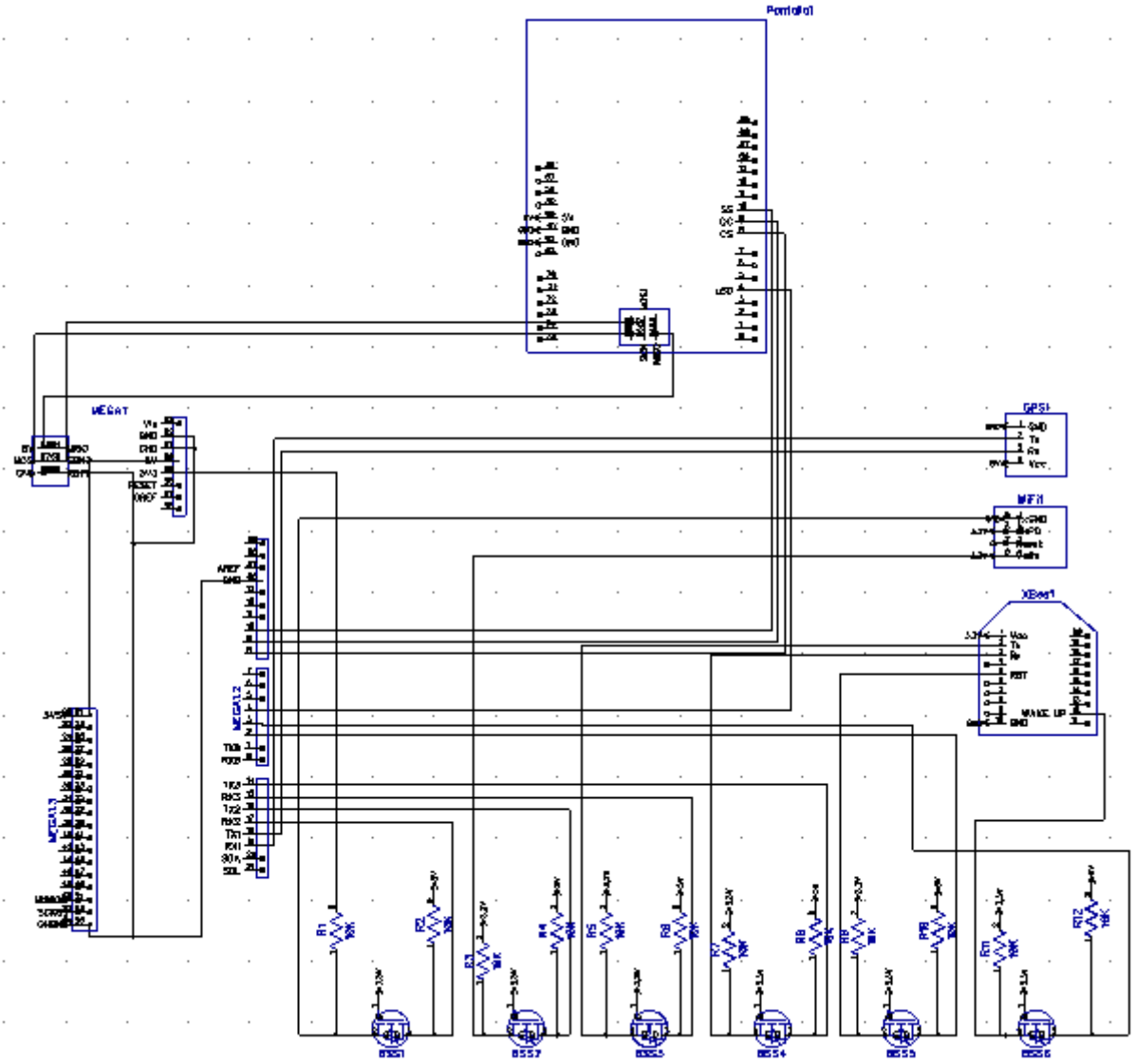

Ilustración 33. Esquemático de la PCB

3.5. PCB Layout

A partir del esquemático se consigue la PCB que, mediante el routing de las pistas, estableciendo la anchura y separación de las pistas, así como su colocación para ocupar el menor tamaño posible obtenemos la PCB.

Las pistas de la PCB se han dividido en dos grupos, cada uno de ellos con características diferentes. Se han dividido en pistas por defecto que son todas las de comunicaciones y en pistas de alimentación que son todas las de alimentación como son las pistas de 5 V, 3,3 V y GND. Como se puede ver a continuación las propiedades de ambos grupos son diferentes, ancho de pistas y

separación entre ellas, teniendo un mayor tamaño las pistas de alimentación. Las propiedades de ambos grupos serán iguales para ambas caras de la PCB (Top y Bottom).

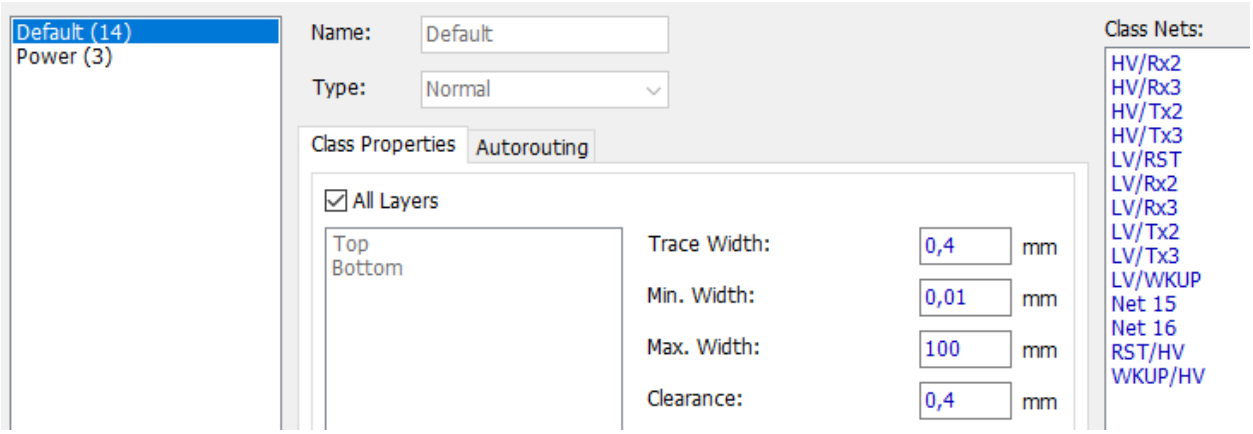

Ilustración 34. Configuración de las pistas de la PCB

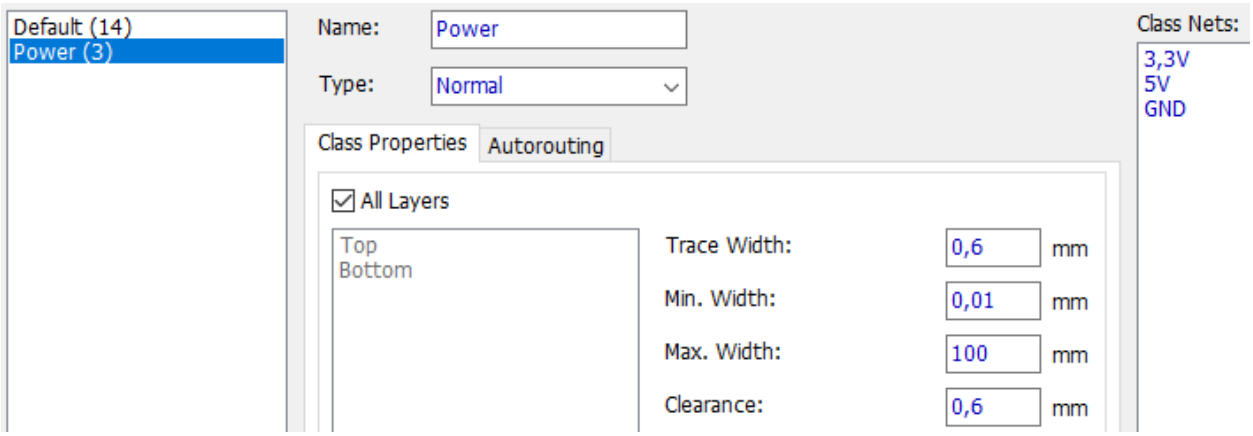

Ilustración 35. Configuración de las pistas de alimentación de la PCB

Además, se ha asignado la capa de cobre a GND con las siguientes propiedades:

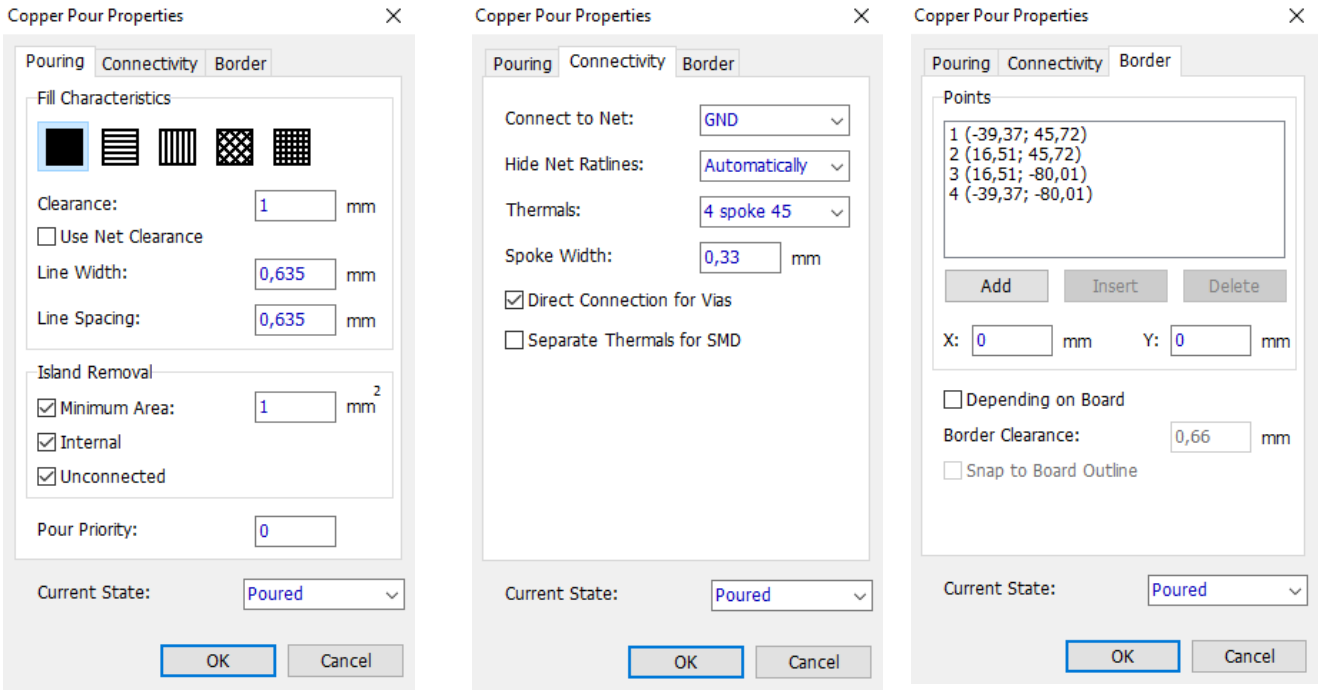

Ilustración 36. Propiedades de la capa de cobre de la PCB

De esta forma conseguimos una PCB como la que sigue:

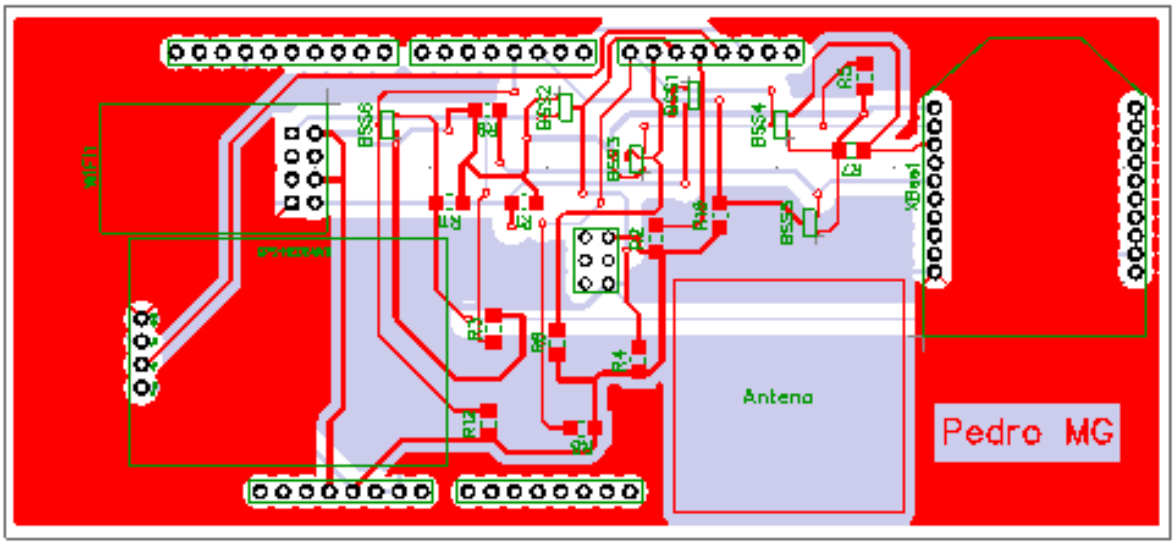

Ilustración 37. PCB

CAPÍTULO 4

IMPLEMENTACIÓN DE LA ARQUITECTURA SOFTWARE

En este apartado se va a explicar qué software se ha utilizado para implementar el código y el funcionamiento general del código con la ayuda de un flujograma. Además de qué librerías se han utilizado para cada uno de los shields y una explicación de las funciones que se han implementado.

El software de código abierto utilizado es Arduino, que está formado por dos elementos: un entorno de desarrollo (IDE) basado en el entorno de Processing y en la estructura del lenguaje de programación Wiring, y en el bootloader (gestor de arranque) que es ejecutado de forma automática dentro del microcontrolador en cuanto este se enciende. Las placas de Arduino se programan mediante un ordenador, usando comunicación serial, en este caso mediante USB, a través del Serial 0 de Arduino MEGA. Este software se puede usar para cualquier placa Arduino.

4.1. Descripción del software

En este apartado se va a desarrollar con más detalle el funcionamiento del software descritas en el apartado 1.4 (Requisitos software). El funcionamiento del software se ha reflejado en el flujograma de la figura adjunta:

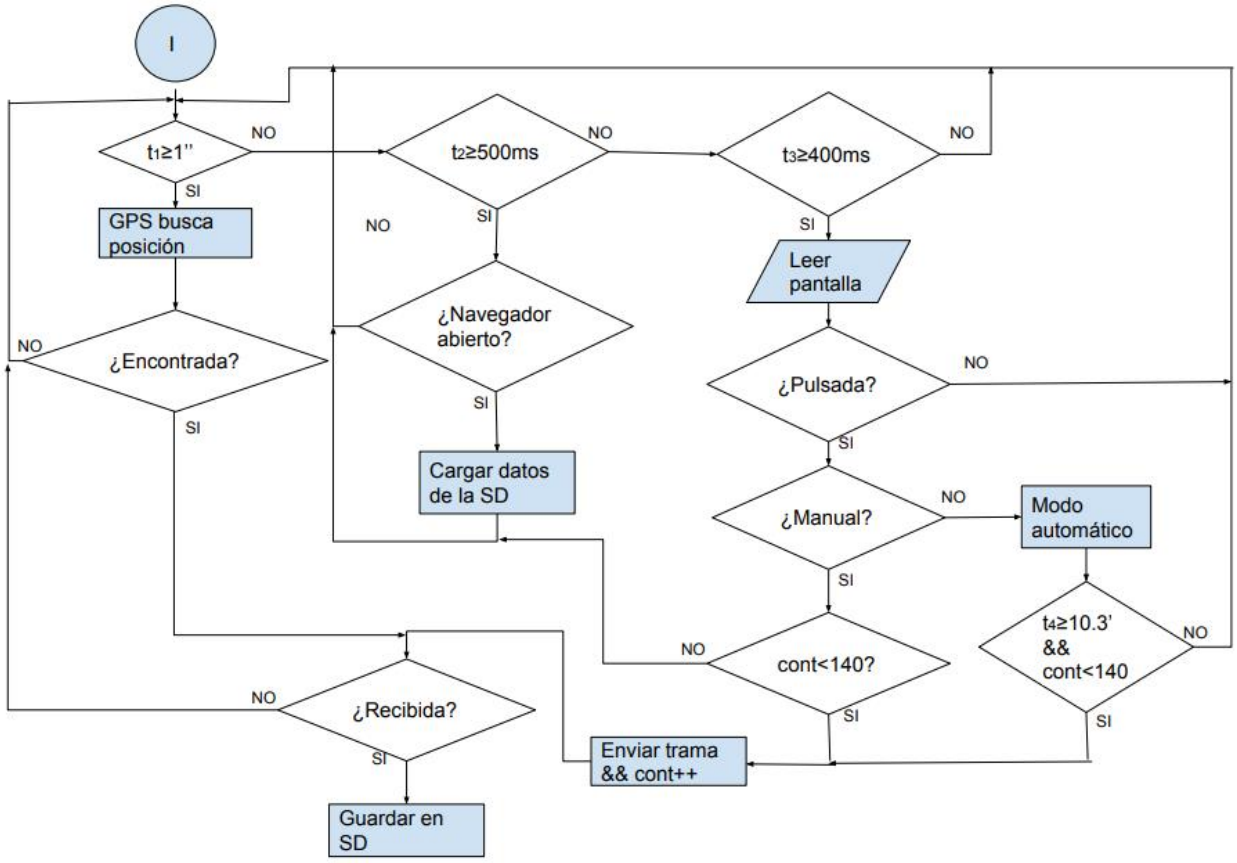

Ilustración 38. Flujograma

Cuando se encienda el dispositivo y tras comprobar que el controlador de la pantalla y de la SD se han iniciado correctamente el GPS comienza a buscar posición. Si en cualquier momento alguien se conecta a la red con ID “ESP8266” y contraseña “password” creada por el dispositivo y abre el navegador e introduce la IP “192.168.4.1” se cargará una página web con todos los datos almacenados en la SD. Además, cada un corto intervalo de tiempo se comprobará si la pantalla táctil ha sido tocada y en qué punto. Dependiendo de donde haya sido presionada el dispositivo se irá a un menú o activará un modo de funcionamiento. En caso de estar en modo automático el dispositivo enviará una trama cada 10,3 minutos, lo que equivale a 139 mensajes al día de forma que nunca sobrepasará los 140. Al ser todas las tramas que se envían contabilizadas en caso de que se envíen tramas en modo manual y luego se ponga en modo automático de nuevo en el momento en que llegue a 140 no se podrá volver a enviar una trama, de forma que en ningún caso se sobrepasará la limitación de los 140 mensajes de Sigfox diarios. Toda la información recogida será almacenada en la SD. Esto está descrito en profundidad en el apartado 4.7 (Interfaz de usuario del dispositivo).

4.2. Módulo GPS NEO6MV2

La librería utilizada para el módulo GPS es TinyGPS++ es una nueva biblioteca de Arduino para analizar flujos de datos NMEA provenientes de los módulos GPS.

NMEA es una especificación eléctrica y de datos combinados para la comunicación entre dispositivos electrónicos marinos, como ecosonda, sonar, anemómetro, giroscópico, piloto automático, receptores GPS y muchos otros tipos de instrumentos. El estándar NMEA usa ASCII, protocolo de comunicaciones en serie que define cómo se transmiten los datos en una "oración" de un "hablante" a múltiples "oyentes" a la vez. Un hablante puede tener una conversación unidireccional con un número casi ilimitado de oyentes.

Al igual que su predecesora, TinyGPS, esta biblioteca proporciona métodos compactos y fáciles de usar para extraer la posición, fecha, hora, altitud, velocidad y rumbo de los dispositivos GPS del consumidor. Sin embargo, la interfaz del programador de TinyGPS++ es considerablemente más fácil de usar que TinyGPS. TinyGPS ++ no es tan "pequeño" como su hermano mayor, pero su nuevo y poderoso modelo de objetos extremadamente útil y nuevo conjunto de características útiles lo convierten en una alternativa atractiva. Al usar Arduino MEGA no hay problemas con el espacio de almacenamiento de programa, por lo que no hay problemas para usar esta librería. Este GPS funciona a 9600 baudios.

La librería contiene los siguientes objetos:

- location: la última posición fijada.
- date: la última fecha establecida.
- time: la última hora establecida.
- speed: velocidad de avance actual.
- course: actual dirección.
- altitude: última altitud fijada.
- satellites: la cantidad de satélites visibles y participes.
- hdop: dispersión de la precisión horizontal (incertidumbre).

La librería incluye una gran cantidad de funciones, todas estas funciones son de tipo gps. A continuación, se van a explicar las utilizadas:

- **encode()**

Del tipo gps.encode(). Para hacer que TinyGPS++ funcione, tienes que obtener los caracteres (tipo int) desde el módulo GPS usando la función encode().

- **altitude.isUpdated()**

Del tipo gps.*object*.isUpdated(), donde se usará algún objeto de los mencionados anteriormente. Esta función se usa para comprobar si el valor del objeto se ha actualizado (no necesariamente cambiado) desde la última vez que lo consultó.

- **isValid()**

Del tipo gps.isValid(). Esta función puede examinar el valor de un objeto en cualquier momento, pero a menos que TinyGPS++ haya obtenido recientemente caracteres desde el GPS, no debe considerarse válido y actualizado. El método isValid() le dirá si el objeto contiene datos válidos y es seguro consultarlo.

- **age()**

Del tipo gps.age(). Si se desea saber si los últimos datos que se han obtenido están obsoletos, se puede llamar al método age(), que devuelve el número de milisegundos desde su última actualización. Si esto devuelve un valor superior a 1500 (1,5 s) o más, puede ser un signo de un problema como un valor de posición perdido.

- **charsProcessed()**

Del tipo gps.charsProcessed(). Esta función obtiene el número total de caracteres obtenidos por el GPS.

- **location.lat()**

Del tipo gps.location.lat(). Esta función obtiene la latitud en grados y devuelve el valor en una variable tipo double.

- **location.lng()**

Del tipo gps.location.lng(). Esta función obtiene la longitud en grados y devuelve el valor en una variable tipo double.

- **altitude.meters()**

Del tipo gps.altitude.meters(). Esta función obtiene la altitud en metros y devuelve el valor en una variable tipo double.

- **satellites.value()**

Del tipo gps.satellites.value(). Esta función obtiene el número de satélites en uso y devuelve el valor en una variable tipo u32.

- **date.year()**

Del tipo gps.date.year(). Esta función obtiene el año actual y lo devuelve en una variable tipo u16.

- **date.month()**

Del tipo gps.date.month(). Esta función obtiene el mes actual y lo devuelve en una variable tipo u8.

- **date.day()**

Del tipo gps.date.day(). Esta función obtiene el día actual y lo devuelve en una variable tipo u8.

- **time.hour()**

Del tipo gps.time.hour(). Esta función obtiene la hora actual y la devuelve en una variable tipo u8.

- **time.minute()**

Del tipo gps.time.minute(). Esta función obtiene los minutos de la hora actual y los devuelve en una variable tipo u8.

- **time.second()**

Del tipo gps.time.second(). Esta función obtiene los segundos de la hora actual y los devuelve en una variable tipo u8.

- **speed.mps()**

Del tipo gps.speed.mps(). Esta función obtiene la velocidad en m/s y los devuelve en una variable tipo double.

- **speed.kmph()**

Del tipo gps.speed.kmph(). Esta función obtiene la velocidad en km/h y los devuelve en una variable tipo double.

- **distanceBetween(newlati,newlon,lati,lon)**

Del tipo TinyGPSPlus::distanceBetween(newlati,newlon,lati,lon). Esta función obtiene la distancia en Km de la posición actual a la última posición obtenida. Devuelve una variable tipo double.

4.3.Módulo WiFi ESP8266

Para este módulo no se ha utilizado ninguna librería. Esto se debe a que la mayor parte de las librerías existentes son para utilizar el ESP8266 como procesador y no como un shield de Arduino. Este módulo funciona a 115200 baudios y se programa con comandos AT, los que se han utilizado para su desarrollo y final implantación son:

- **AT**

Comando	Respuesta	Función
AT	OK	Prueba que se ha iniciado y si el módulo responde correctamente

- **AT+RST**

Comando	Respuesta	Función
AT+RST	OK	Resetéa el módulo

- **AT+CWMODE**

Comando	Respuesta	Función
AT+CWMODE=?	+CWMODE:(1-3) OK	Lista los modos validos
AT+CWMODE?	+CWMODE:modo OK	Pregunta en qué modo AP está actualmente el módulo
AT+CWMODE=modo	OK	Establece el módulo en el modo dado 1 = Modo estación (cliente) 2 = Modo AP (huésped) 3 = Modo AP + Estación (modo dual)

- **AT+CWLAP**

Comando	Respuesta	Función
AT+CWLAP	AT+CWLAP:ecn,ssid,rssi,mac OK	Lista los Acess Points disponibles para conectarse. ecn: codificación, puede ser: 0 = Abierto

		1 = WEP 2 = WPA PSK 3 = WPA2 PSK 4 = WPA WPA2 PSK ssid: String que contiene el SSID del AP rssi: Fuerza de la señal mac: String que contiene la dirección MAC
AT+CWLAP=ssid,mac,ch	+CWLAP:ecn,ssid,rssi,mac OK	Busca Acess Points disponibles para conectarse con las condiciones especificadas

- **AT+CWJAP**

Comando	Respuesta	Función
AT+CWJAP?	+CWJAP:ssid OK	Imprime el SSID al que el módulo esta conectado
AT+CWJAP=ssid,pwd	OK	El módulo se conecta al la red con el nombre ssid indicado y la contraseña pwd suministrada

- **AT+CWQAP**

Comando	Respuesta	Función
AT+CWQAP	OK	Se desconecta de la red que está actualmente conectado

- **AT+CWSAP**

Comando	Respuesta	Función
AT+CWSAP?	+CWSAP:ssid,pwd,ch,ecn OK	Pregunta la configuración actual del softAP

- **AT+CIPSTATUS**

Comando	Respuesta	Función
AT+CIPSTATUS	STATUS:status +CIPSTATUS:id,type,addr,port,tetype OK	status: 2 = Se obtuvo IP 3 = Conectado 4 = Desconectado id: ID de la conexión en caso de multiconexión (1-4) type: Tipo de conexión, "TCP" o "UDP" addr: Dirección IP de la conexión port: Numero del puerto tetype: 0 = El módulo corre como cliente 1 = El módulo corre como servidor

- **AT+CIPMUX**

Comando	Respuesta	Función
AT+CIPMUX	AT+CIPMUX	Habilitar o deshabilitar multiples conexiones

- **AT+CIPSTART**

Comando	Respuesta	Función
AT+CIPSTART=type,addr,port	OK	Empieza una conexión como cliente (en modo conexión única) type: puede ser "TCP" o "UDP" addr: String que contenga la dirección IP remota port: String que contenga el puerto remoto
AT+CIPSTART=id,type,addr,port	OK	Empieza una conexión como cliente (En modo conexión múltiple) id: ID de la conexión (1-4)

AT+CIPSTART=?	[+CIPSTART:(id)(“type”),(“ip address”),(port)] OK	Lista los posibles comandos

- **AT+CIPCLOSE**

Comando	Respuesta	Función
AT+CIPCLOSE=?	OK	
AT+CIPCLOSE=id	OK	Cierra la conexión TCP o UDP con el ID "id" en modo conexión múltiple
AT+CIPCLOSE	OK	Cierra la conexión TCP o UDP para modo de conexión única

- **AT+CIPSEND**

Comando	Respuesta	Función
AT+CIPSEND=?	OK	
AT+CIPSEND=length	SEND OK	Establece la longitud de datos a enviarse (máximo 2048). Para un envío normal (modo conexión única)
AT+CIPSEND=id,length	SEND OK	Establece la longitud de datos a enviarse en la conexión número "id". Para un envío normal (modo conexión múltiple)
AT+CIPSEND		Envía datos sin adornos cada 20ms. El módulo retorna ">" después ejecutar el comando, si se recibe el comando "+++" se regresa al modo comando.

- **AT+CIFSR**

Comando	Respuesta	Función
AT+CIFSR=?	OK	
AT+CIFSR	+CIFSR:ip OK	Retorna la dirección IP local del módulo como cliente.
AT+CIFSERVER		Configurar como servidor
AT+CIPSERVER=mode[,port]	OK	Configura el módulo como servidor donde el modo: 0 = Borrar servidor 1 = Crear servido puerto: numero del puerto, por defecto es el 333

- **AT+CIOBAUD**

Comando	Respuesta	Función
AT+CIOBAUD=?	+CIOBAUD:(9600-921600) OK	Nos informa que las velocidades de transmisión permitidas están en este rango
AT+CiOBAUD?	+CIOBAUD:baudrate OK	Nos indica que el módulo está actualmente configurado a 'baudrate'
AT+CIOBAUD=baudrate	OK	Configura la velocidad de transmisión a 'baudrate'

4.4. Radio LPWAN

La configuración UART de la radio es la siguiente:

- Baud Rate: 38400
- Data: 8 bits
- Parity: None
- Stop: 1 bit
- Flow control: None
- End line character: <LF>

Para este módulo, al igual que en el del módulo WiFi, no se han usado librerías. Al funcionar con comandos AT este código se ha inspirado en el del ESP8266, siendo el modo de solicitar comandos AT y esperar respuesta muy similar. Los comandos AT se usan como una interfaz con los módulos de comunicación de Nemeus. Los módulos pueden manejarse en 3 niveles diferentes:

- Nivel de radio: el servidor AT incorporado utiliza la API del controlador RF SX127x (se pueden usar modulaciones LoRa ™ y FSK). Cuando se utiliza el nivel de radio, no se requiere personalización del dispositivo, depende de la aplicación del cliente AT implementar la capa de red de acuerdo con la red a la que está conectado el dispositivo (el módulo no conoce la red).
- Nivel WAN de LoRa ™: el servidor AT incorporado utiliza la API de la biblioteca WAN de LoRa ™ (disponible solo cuando la biblioteca está presente en el software incorporado).
- Nivel SIGFOX ™: el servidor AT incorporado utiliza la API de biblioteca SIGFOX ™ (disponible solo cuando la biblioteca está presente en el software incorporado).

Antes de poder utilizar el módulo, la aplicación cliente AT debe activar el nivel que desea usar. Solo se puede activar un nivel a la vez. Al activar un nivel, el nivel previamente activado se desactiva automáticamente. En este caso se ha utilizado la radio en su nivel de Sigfox, los comandos AT son los siguientes:

- **AT+SF=HELP**

Este comando se usa para saber la lista de subcomandos.

Respuesta	OK

- **AT+SF=ON**

Este comando inicia el nivel de SIGFOX y es obligatorio su uso para poder utilizar cualquier otro comando de SIGFOX.

Respuesta	Si el nivel de LoRa está activado responde: ERROR Sino: OK

- **AT+SF=OFF**

Desactiva la capa de SIGFOX.

Respuesta	Si el nivel de SIGFOX está usando la radio responde: ERROR Sino: OK

- **AT+SF=?**

Lee el estado actual de la capa de SIGFOX.

Respuesta	Si tiene la librería de SIGFOX responde: OK Sino: ERROR +SF:<state>,<NMS_lib_ver>,<SFX_lib_ver>,<dev_id>,<initial_pac>

<state> es ON, OFF o DUAL. DUAL significa que el nivel de LoRa™WAN y SIGFOX™ entán ambas en ON.

<NMS_lib_ver> es la versión de la librería de Nemeus usada para comunicarse con la red de SIGFOX™.

<SFX_lib_ver> es la versión de la librería de SIGFOX™.

<dev_id> es el identificador del dispositivo en la red de SIGFOX™.

<initial_pac> es el Portability Access Code usado para registrar el dispositivo en la red de SIGFOX™. Sólo se usa una vez para el registro.

- **AT+SF=SNDBIN**

Transmisión de una trama binaria.

AT+SF=SNDBIN,<binpayload>,<ack>

Si el valor <ack> es 0 entonces la tramaes enviada en modo desconocido (el valor por defecto cuandose omite <ack>).

Si el valor <ack> es 1, entonces la trama se envía en modo conocido.

The frame is sent when channel becomes free regarding duty cycle limitations.

La trama se envía cuando el canal se libera, hay limitaciones del ciclo de trabajo.

Respuesta	+SF: <time_on_air> se envía justo antes de la respuesta solicitada. El tiempo en el aire está dado en ms, es utilizado por la aplicación del cliente para administrar el ciclo de trabajo. OK si la capa SIGFOX ™ está activada y la trama se ha enviado correctamente. ERROR si la capa de SIGFOX ™ está DESACTIVADA o si la trama no se ha enviado con éxito.
Respuestas no solicitadas	Una indicación sobre la fecha Tx: +SF: SND, <busytime> <busytime> está en ms. Se puede enviar 2 veces: • Una vez con busytime> 0 si la subbanda Tx no estaba libre debido a la restricción del ciclo de trabajo. • Una vez con busytime = 0 en el tiempo Tx. La respuesta no solicitada que hay a continuación se envía cuando se solicitó el modo de respuesta. +SF: RCVBIN, <binpayload>, <rssi>

- **AT+SF=SNDBIT**

Transmisión de bit.

AT+SF=SNDBIT, <bitvalue>, <ack>

Si el valor <ack> es 0, entonces el bit se envía en modo desconocido (valor predeterminado cuando se omite <ack>).

Si el valor <ack> es 1, entonces el bit se envía en modo reconocido.

Respuesta	+SF: <time_on_air> se envía justo antes de la respuesta solicitada. El tiempo en el aire es en ms, es utilizado por la aplicación del cliente para administrar el ciclo de trabajo. OK si la capa SIGFOX ™ está activada y el cuadro se ha enviado correctamente (y se anotó si se solicitó el modo de respuesta). ERROR si la capa de SIGFOX ™ está DESACTIVADA o si el marco no se ha enviado con éxito (o no se ha añadido después de las repeticiones si se solicitó el modo de respuesta).

Respuestas no solicitadas	Una indicación sobre la fecha Tx: +SF: SND, <busytime> <busytime> está en ms. Se puede enviar 2 veces: • Una vez con busytime> 0 si la subbanda Tx no estaba libre debido a la restricción del ciclo de trabajo. • Una vez con busytime = 0 en el tiempo Tx. La siguiente respuesta no solicitada se envía sólo si se solicitó el modo de respuesta. +SF: RCVBIN, <binpayload>, <rssi>

- **AT+SF=SNDOOB**

Transmisión de mensajes fuera de banda.

Respuesta	+SF: <time_on_air> se envía justo antes de la respuesta solicitada. El tiempo en el aire está en ms, puede ser utilizado por el cliente para administrar el ciclo de trabajo. OK si la capa SIGFOX ™ está ENCENDIDA y el cuadro se ha enviado correctamente. ERROR si la capa de SIGFOX ™ está DESACTIVADA o el marco no se ha enviado con éxito.
Respuestas no solicitadas	Una indicación sobre la fecha Tx: +SF: SND, <busytime> <busytime> está en ms. Se puede enviar 2 veces: Una vez con busytime> 0 si la subbanda Tx no estaba libre debido a la restricción del ciclo de trabajo. Una vez con busytime = 0 en el tiempo Tx.

- **AT+SF=RTXF**

Lea el macro canal del operador de salida.

Respuesta	+SF: <tx_frequency> se envía justo antes de la respuesta solicitada. El valor predeterminado es 868130000 Hz. Siempre OK.

- **AT+SF=STXF**

Responde si la frecuencia de transmisión establecida está en un rango válido.

Respuesta	OK si 863000000 <= tx_frequency <= 870000000. ERROR si tx_frequency no está en el rango válido.

- **AT+SF=RRXF**

Leer el canal macro del operador de recepción.

Respuesta	+ SF: <rx_frequency> se envía justo antes de la respuesta solicitada. El valor predeterminado es 869525000 Hz.

- **AT + SF = SRXF,**

Establezca el macro canal del operador de recepción.

AT + SF = SRXF, <rx_frequency>

Respuesta	OK si 863000000 <= rx_frequency <= 870000000. ERROR si rx_frequency no está en rango válido.

- **AT + SF = RTXP (para fines de prueba)**

Leer la potencia de Tx aplicada a la onda continua FSK

Respuesta	SF: <tx_power> se envía justo antes de la respuesta solicitada. El valor predeterminado es 14 dBm. Siempre OK.

- **AT + SF = STXP (para fines de prueba)**

Establezca la potencia de Tx aplicada a la onda continua FSK

Respuesta	OK si tx_power <= 14 dBm. ERROR si tx_power> 14 dBm.

4.5.Pantalla táctil

Para la pantalla se han utilizado las librerías que proporciona Adafruit, las librerías son Adafruit_STMPE610 y Adafruit_ILI9341. Cada una de ellas se encarga de una parte de la pantalla.

Adafruit_ILI9341 es la librería encargada de traducir e interpretar coordenadas para poder trabajar con el panel táctil resistivo de 4 hilos que incluye nuestro shield. La librería Adafruit_STMPE610 es la que se encarga de la comunicación a bajo nivel con el chip del controlador STMPE610.

Se ha renombrado la clase Adafruit_STMPE610 como ts (las siglas de touch screen). Estas son las funciones que se utilizan para controlar el panel táctil de la pantalla:

- **ts.begin();**

Esta función se usa para inicializar el controlador del panel táctil resistivo. Debe ir al principio del código para asegurar el correcto funcionamiento del controlador. Debe devolver true lo que significa que el driver se ha encontrado, de lo contrario devolverá false.

- **ts.bufferEmpty();**

Esta función se usa para comprobar si el buffer está vacío. Devuelve true si está vacía y false en caso contrario.

- **ts.bufferSize();**

Esta función se usa para comprobar el tamaño del buffer. Devuelve el tamaño del buffer.

- **ts.touched();**

Esta función se usa para saber si la pantalla está siendo tocada en ese momento. Devolverá true en caso de que esté siendo tocada y false en caso contrario.

- **ts.getPoint();**

Esta función obtiene del buffer el punto en el que se ha tocado la pantalla. Para obtener el punto más antiguo del buffer TS_Point al que declararemos como p, tiene puntos de datos .x, .y y .z. Los puntos x e y van de 0 a 4095. El punto z es la presión y va de 0 a 255, cuanto más se presione, menor será su número. Como los datos del STMPE610 van de 0-4095 pero la pantalla mide 320x240 píxeles, podemos usar map para convertir 0-4095 en 0-320 o 0-240.

Para el caso de la clase Adafruit_ILI9341se ha optado por renombrarla como tft. Las funciones de esta librería son:

- **tft.begin();**

Esta función se usa para inicializar el controlado. Debe ir al principio del código para asegurar el correcto funcionamiento del controlador.

- **tft.fillScreen(ILI9341_COLOR);**

Esta función pone el color del fondo de la pantalla que se desee. Hay varios colores implementados y se puede añadir el que se quiera añadiéndolo en la librería, en este caso con los que ya llevaba implementados ha sido suficiente.

- **tft.drawLine(pto ini x, pto ini y, pto fin x, pto fin y, ILI9341_COLOR);**

De esta forma se consigue dibujar una línea con origen y fin en los puntos x e y que se deseen, además de poder elegir el color de línea que se quiera.

- **tft.fillRect(pto ini x, pto ini y, pto fin x, pto fin y, ILI9341_COLOR);**

Con esta función se consigue dibujar un rectángulo del color que se quiera. Pudiéndose elegir el punto x e y de inicio y de fin del cuadrado. De esta forma se consigue el tamaño y la posición del rectángulo que se desee.

- **tft.drawRect(pto ini x, pto ini y, pto fin x, pto fin y, ILI9341_COLOR);**

Esta función es similar a la anterior con la diferencia de que en este caso sólo se dibujan los bordes del cuadrado. Pudiendo mostrarse así, el contenido del interior. El color que se asigna será el de los bordes del rectángulo.

- **tft.setRotation(orientacion);**

Con esta función conseguimos darle a la pantalla la orientación que se desee. Las posibles orientaciones son: 0, 1, 2 y 3. La posición de las coordenadas de la pantalla dependerá de la orientación que se haya utilizado, de modo que una aplicación que se haya desarrollado en una orientación no valdrá para otra orientación, ya que cambiarán todas las coordenadas de la pantalla.

- **tft.setTextColor(ILI9341_COLOR);**

Con esta función se puede cambiar el color de fuente del texto que aparece por pantalla. Escribiendo en COLOR el color de letra que se desee.

- **tft.setTextSize(size);**

De esta forma se elige el tamaño de la letra que aparece por pantalla. Los tamaños de letra se asignan pasando por referencia en size valores del 1 al 4 y de menor a mayor respectivamente.

- **tft.setCursor(x,y);**

Esta función se utiliza para situar el cursor en la posición del LCD deseada. Poniendo en x e y las coordenadas donde se quiera colocar el cursor para escribir. Las posiciones máximas son el tamaño del LCD, Xmax=240 e Ymax=320.

- **tft.println("Text");**

Para imprimir por pantalla se escribe el texto que se quiera entre comillas. También se le puede pasar una variable del tipo String sin las comillas. Un ejemplo sería guardar los valores obtenidos por el GPS en una variable tipo String y pasárselos por referencia para mostrarlos en el LCD.

- **bmpDraw(bmpfilename, x, y);**

Esta función sirve para hacer nuevos mapas de bits, asegurándose de que tengan menos de 320x240 píxeles y guardarlos en formato BMP de 24 bits (el formato más sencillo para Arduino). Se pueden dibujar o cargar tantas imágenes de la SD como se desee, aunque los nombres deben tener menos de 8 caracteres. Las imágenes pueden colocarse en cualquier sentido y posición de la pantalla. bmpfilename será el nombre con el que se ha guardado dicha imagen en la SD y x e y la posición en coordenadas en la pantalla donde se situará la imagen.

4.6.uSD

SD es la librería que incluye el entorno de Arduino por defecto para el acceso y comunicación con tarjetas SD, o en este caso, MicroSD. Esta librería incorpora funciones para el manejo de ficheros y directorios, bajo las clases SD y File.

La librería SD proporciona funciones para acceder a la tarjeta SD y manipular sus archivos y directorios. Las funciones utilizadas con la clase SD son las siguientes:

- **SD.begin(cspin);**

Inicia la SD, se le pasa por referencia el pin conectado al chip de selección de la tarjeta microSD. Devuelve true si lo consigue y false en caso contrario.

- **SD.exists(filename);**

Esta función se encarga de comprobar si existe un fichero con el mismo nombre ya almacenado en la SD que se le pase por referencia (se pueden incluir directorios delimitados por /). Devuelve true en caso de que el fichero exista y false en caso contrario.

- **SD.remove(filename);**

Elimina un fichero con el nombre que se le pasa por referencia. Devuelve true en caso de que el fichero haya sido eliminado con éxito y false en caso contrario.

- **SD.open(filepath, mode);**

Esta función abrirá un fichero con el nombre que se le pase por referencia (se pueden incluir directorios delimitados por /). Además, tiene dos posibles modos de funcionamiento, FILE_READ para sólo lectura y FILE_WRITE para lectura y escritura (si no se especifica, se abrirá por defecto en modo de lectura). Devuelve un objeto File que hace referencia al archivo abierto; si el archivo no se puede abrir, este objeto devolverá un false.

- **SD.mkdir(filename);**

Crea un directorio en la SD, filename es el nombre del directorio que se creará. Se pueden crear subdirectorios separados por /. Devuelve true si el directorio se ha creado correctamente, false en caso contrario.

- **SD.rmdir(filename);**

Eliminar un directorio de la SD, filename es el nombre del directorio que se eliminará. Se pueden eliminar subdirectorios separados por /. Devuelve true si el directorio se ha eliminado con éxito, false en caso contrario.

La librería SD también incorpora funciones para la lectura y escritura en archivos individuales, bajo la clase File. Las funciones de esta clase son:

- **file.name();**

Esta función devuelve el nombre del documento.

- **file.available();**

Comprueba si quedan bytes por leer en el archivo. Devuelve un int con el número de bytes disponibles.

- **file.close();**

Cierra el archivo y se asegura de que todos los datos escritos en él se guardan en la tarjeta SD.

- **file.flush();**

Garantiza que los bytes escritos en el archivo se guarden en la tarjeta SD. Esto se hace automáticamente cuando el archivo se cierra.

- **file.size();**

Con esta función se obtiene el tamaño del archivo. Devuelve el tamaño del archivo en bytes en una variable de tipo unsigned long.

- **file.read();**

Esta función lee del fichero. Devuelve el siguiente byte (o carácter), o -1 si no hay ninguno disponible.

- **file.write(data);**

Con esta función se puede escribir en el archivo. Imprimirá lo que se le pase por referencia en data, ya sea una variable del tipo char, byte o string Devuelve el número de bytes escritos.

- **file.print(data);**

Imprime datos en el archivo, que debe haber sido abierto para poder escribir en él. Imprime números como una secuencia de dígitos, cada uno de un carácter ASCII. Imprimirá lo que se le pase por referencia en data, ya sea una variable del tipo char, byte, int, long o string. Esta función devuelve el número de bytes escritos.

- **file.println(data);**

Esta función es como la anterior, pero en este caso imprime datos en el archivo seguidos por un retorno de carro y una nueva línea. Debe haberse abierto para poder escribir en él. Imprime números como una secuencia de dígitos, cada uno de un carácter ASCII. Imprimirá lo que se le pase por referencia en data, ya sea una variable del tipo char, byte, int, long o string. Esta función devuelve el número de bytes escritos.

- **file.position();**

Obtiene la posición actual de lectura/escritura dentro del archivo (es decir, la ubicación a la que se leerá o escribirá el siguiente byte). Devuelve la posición dentro del archivo en una variable del tipo unsigned long.

- **file.seek(pos);**

Mueve el punto de lectura/escritura actual a la posición que se le pasa, esta posición debe estar entre 0 y file.size() (el tamaño total del fichero). Devuelve true si lo consigue, false en caso contrario.

- **file.peek();**

Lee un byte del archivo sin avanzar al siguiente. Es decir, las llamadas sucesivas a peek() devolverán el mismo valor. Devuelve el siguiente byte (o carácter), o -1 si no hay ninguno disponible.

4.7. Interfaz de usuario del dispositivo

Para mostrar la interfaz de usuario, se muestran a continuación imágenes, que explican aspectos fundamentales del funcionamiento del dispositivo. Se explicarán las funcionalidades de las teclas y de las opciones disponibles a través de los diferentes menús.

En primer lugar, al encender el dispositivo, aparece la pantalla principal. En esta pantalla se muestra el estado en el que se encuentra el dispositivo. En la barra de estado podemos visualizar los iconos de WiFi y de GPS, así como el modo de funcionamiento y la hora. Tanto el icono del WiFi como el de GPS aparecerán en blanco mientras un dispositivo no se conecte a la red del shield WiFi y abra el navegador y mientras el GPS no encuentre posición (latitud y longitud). El icono del GPS permanecerá parpadeando mientras busca la posición. La letra A nos indica que el dispositivo se encuentra en modo automático y permanecerá así mientras no se pulse el botón

superior derecha (MANUAL/AUTOMAT) para cambiar a modo manual. En modo automático el dispositivo envía una trama cada 10,3 minutos, lo suficiente para no sobrepasar el límite de 140 mensajes de Sigfox diarios. La hora es obtenida por el GPS, que tras unos segundos de ser encendido el dispositivo. Tanto la información de subida como la de bajada es la que se establece por defecto al encender el dispositivo. El RSSI será 0 mientras no se envíe una nueva trama y se reciba la potencia con la que llega la señal de la estación base. Una vez el GPS obtiene la posición se actualizarán en la pantalla las coordenadas con una precisión de 6 dígitos.

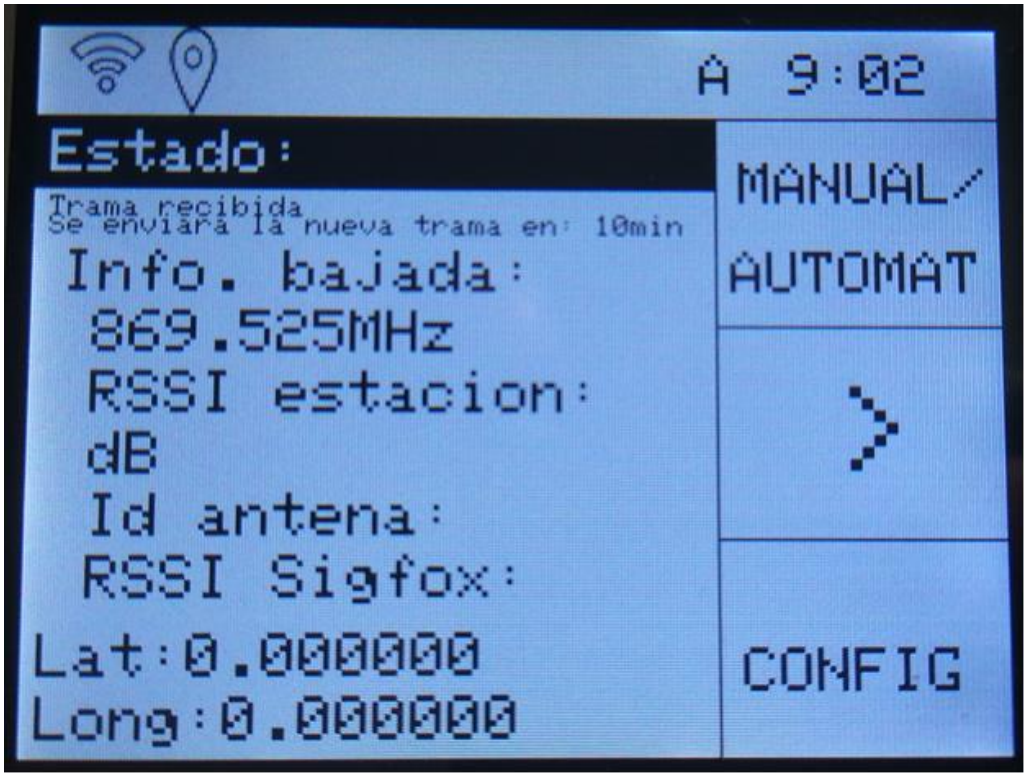

Ilustración 39. Pantalla de inicio (principal)

- Pulsando “MANUAL/AUTOMAT”

Una vez que el GPS encuentra posición se actualizarán los valores de latitud y longitud que inicialmente eran 0. Además, si presionamos el botón superior derecho de la pantalla (MANUAL/AUTOMAT) entrará en modo manual y enviará una trama de Sigfox. Tras un corto

periodo de tiempo se recibe la respuesta de la que obtendremos la RSSI de la estación, el identificador de la antena y la RSSI de la red de Sigfox que se actualizará en la pantalla principal.

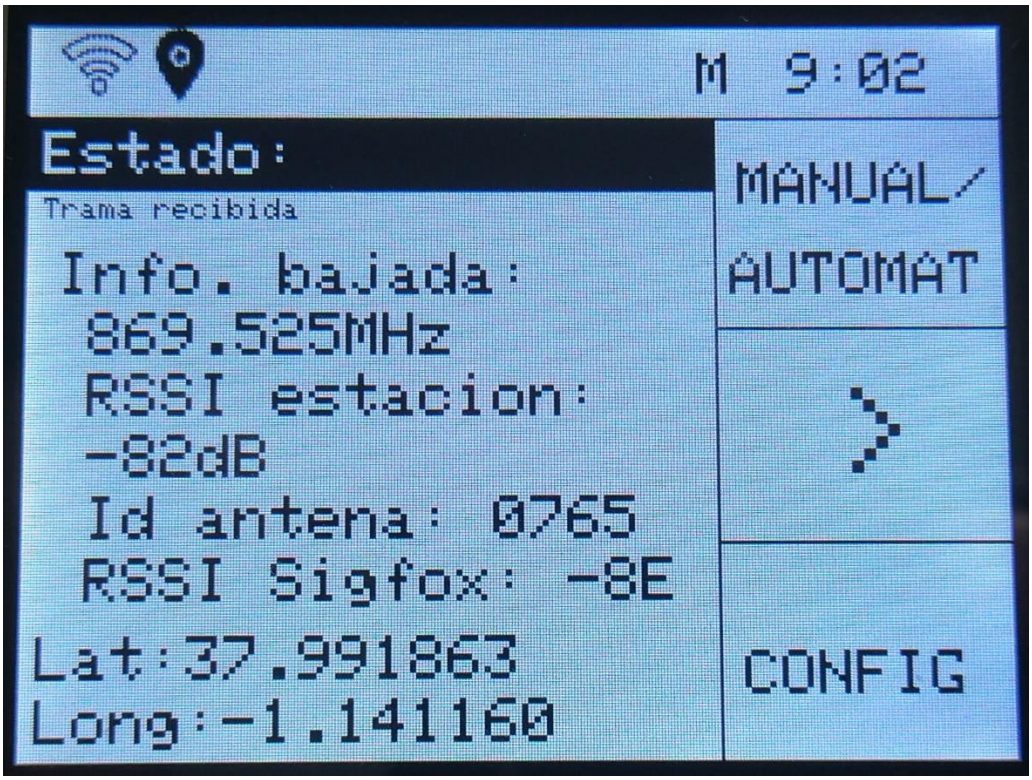

Ilustración 40. Pantalla principal en modo manual (trama recibida)

- Pulsando “>”

Esta es la pantalla que muestra toda lo información recogida por la radio.

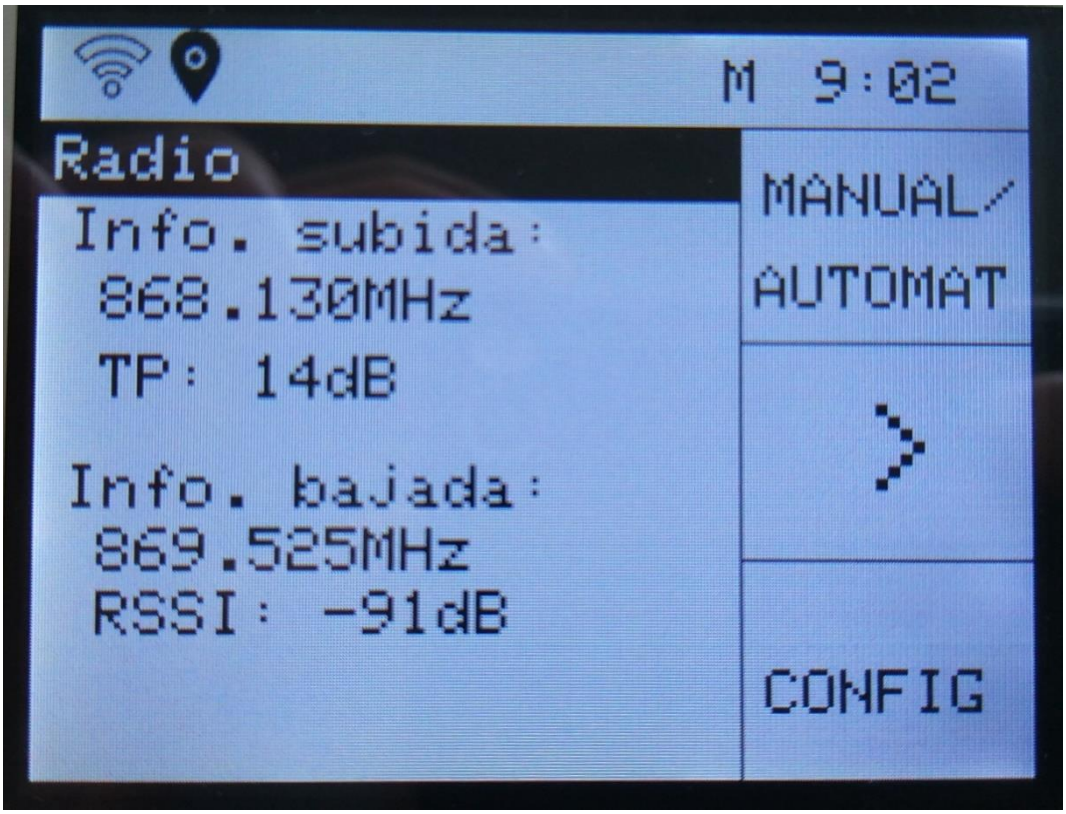

Ilustración 41. Pantalla de la Radio

- Pulsando “>”

En esta pantalla se muestra el PER (del inglés, Packet Error Rate) que es la tasa de error de los paquetes recibidos respecto a los enviados.

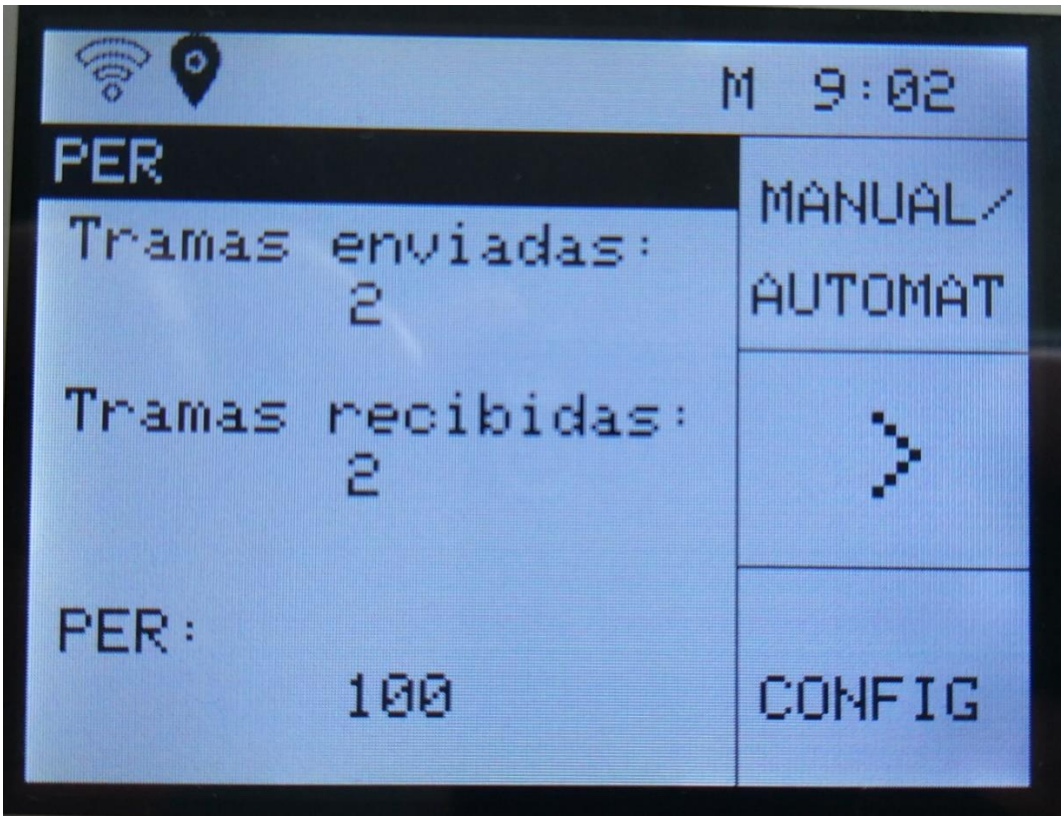

Ilustración 42. Pantalla del PER

- Pulsando “>”

Esta es la pantalla que muestra toda lo información recogida por el GPS.

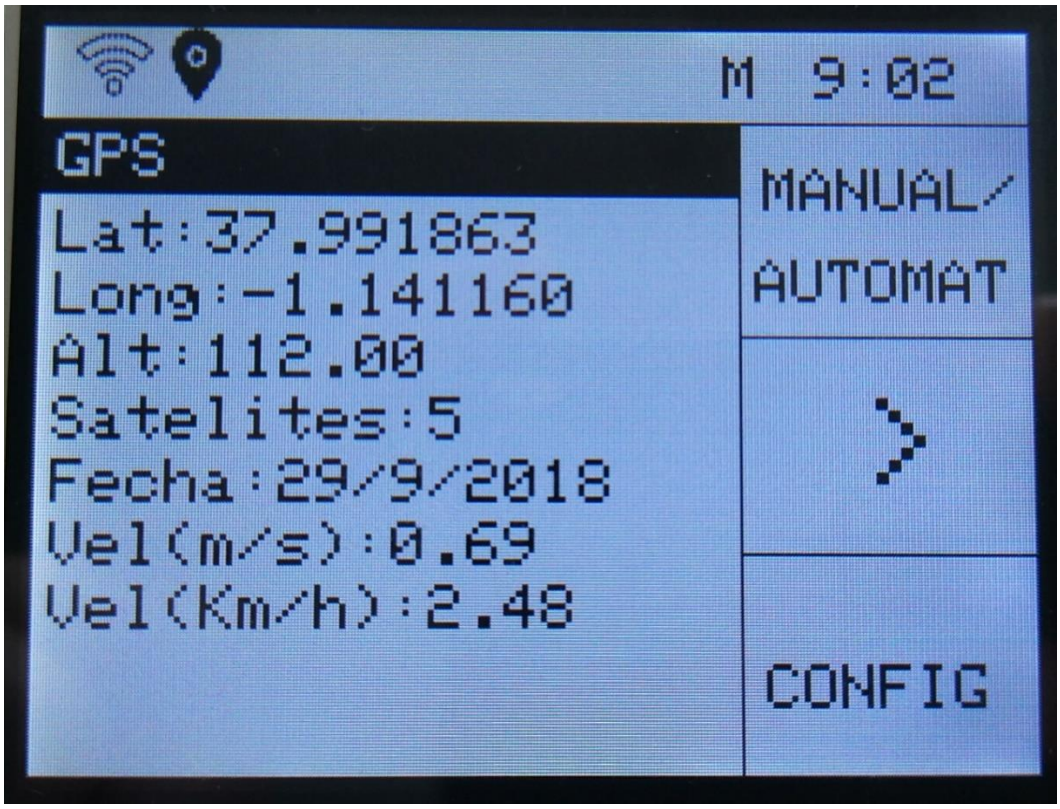

Ilustración 43. Pantalla del GPS

- Pulsando “>”

Esta imagen muestra la pantalla de la información del WiFi. En ella, se muestran los pasos que se deben seguir para poder conectarse a la red WiFi del dispositivo y cómo visualizar en la web la información almacenada en la memoria SD. Si se vuelve a pulsar “>” se volverá a la primera pantalla, la pantalla principal.

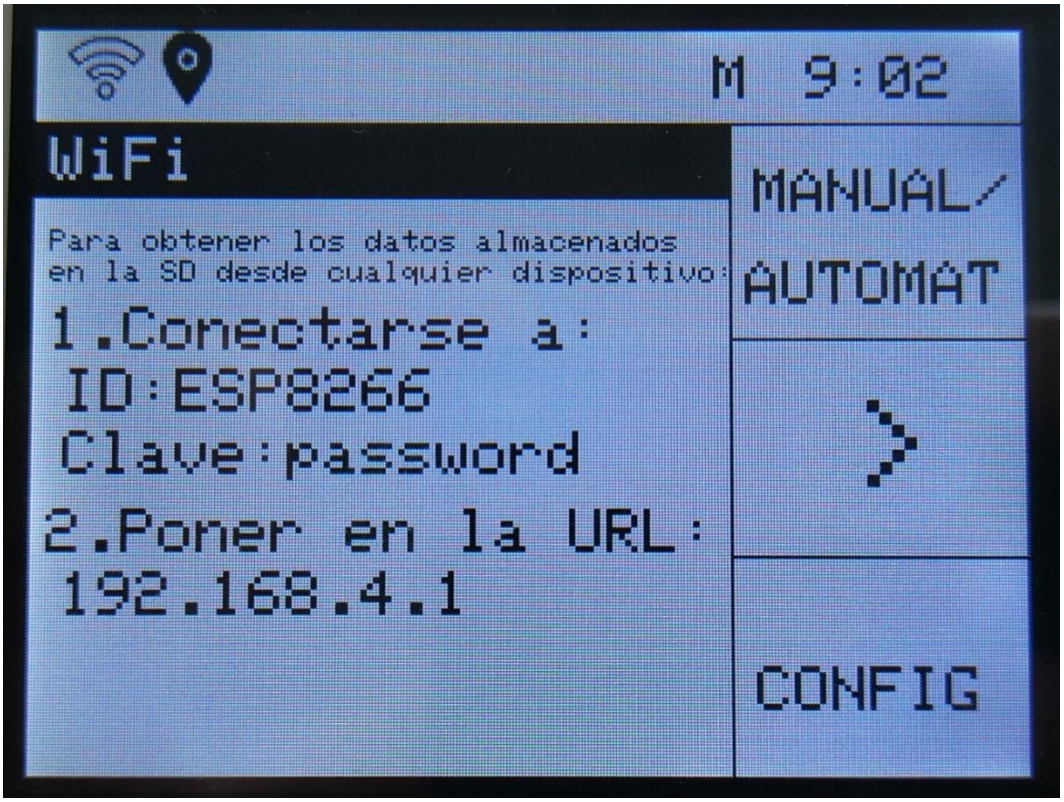

Ilustración 44. Pantalla del WiFi

- Pulsando “CONFIG”

Si se pulsa el botón inferior derecho (CONFIG), se mostrarán opciones de configuración de parámetros como la potencia de transmisión, la frecuencia de transmisión y la de recepción. El tiempo por si se desea modificar cada cuanto enviar una trama, se ponga el tiempo que se ponga nunca se superarán los 140 mensajes diarios.

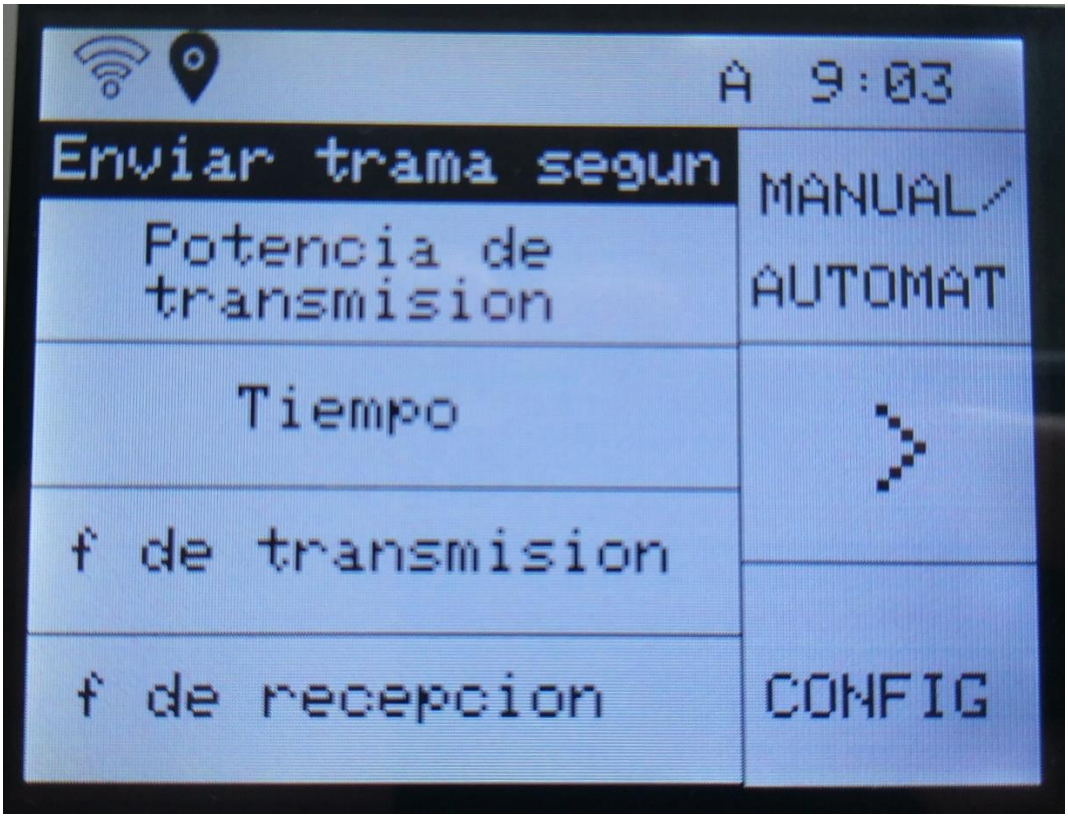

Ilustración 45. Pantalla de configuración

Pulsando en los números hasta llegar al valor deseado y tras pulsar en el rectángulo central se seleccionará el valor y se establecerá para los siguientes envíos que se hagan, ya sean de forma manual o automática. Si se establece un valor que no se encuentre entre los que se indican en cada una de las configuraciones el valor no se establecerá, manteniendo el que haya anteriormente, ya sea el que se encontraba por defecto o el último que se puso.

- Pulsando “Potencia de transmision”

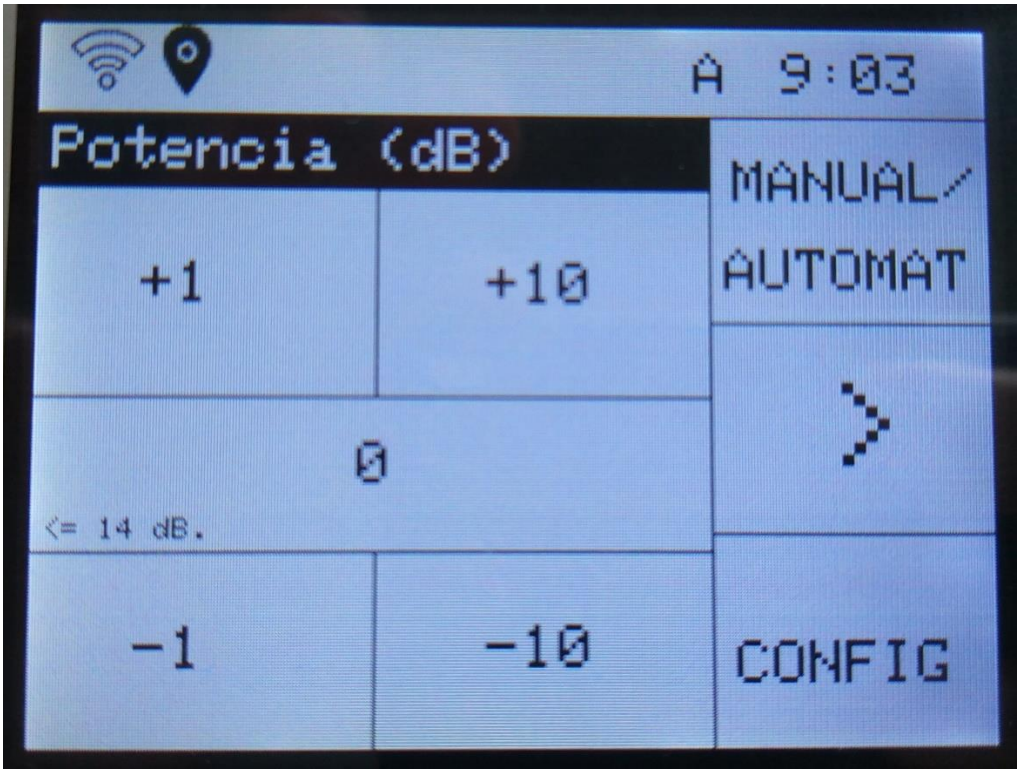

Ilustración 46. Pantalla de configuración de potencia de transmisión

- Pulsando “Tiempo”

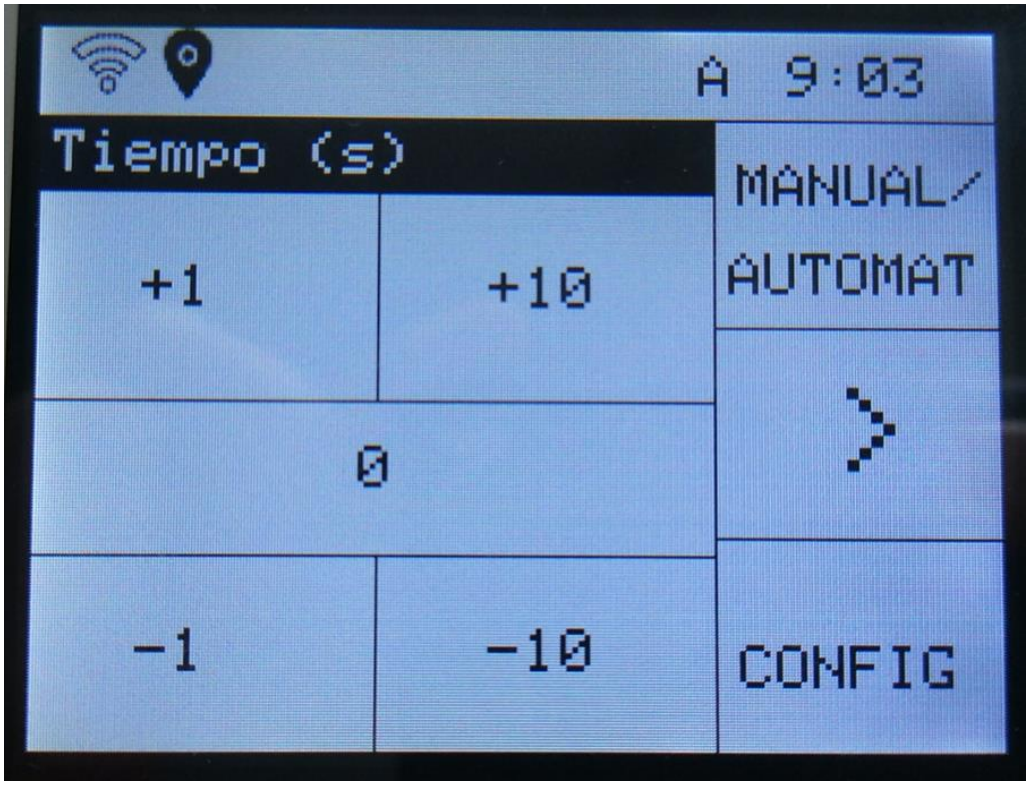

Ilustración 47. Pantalla de configuración de intervalo de tiempo de envío

- Pulsando “f de transmision”

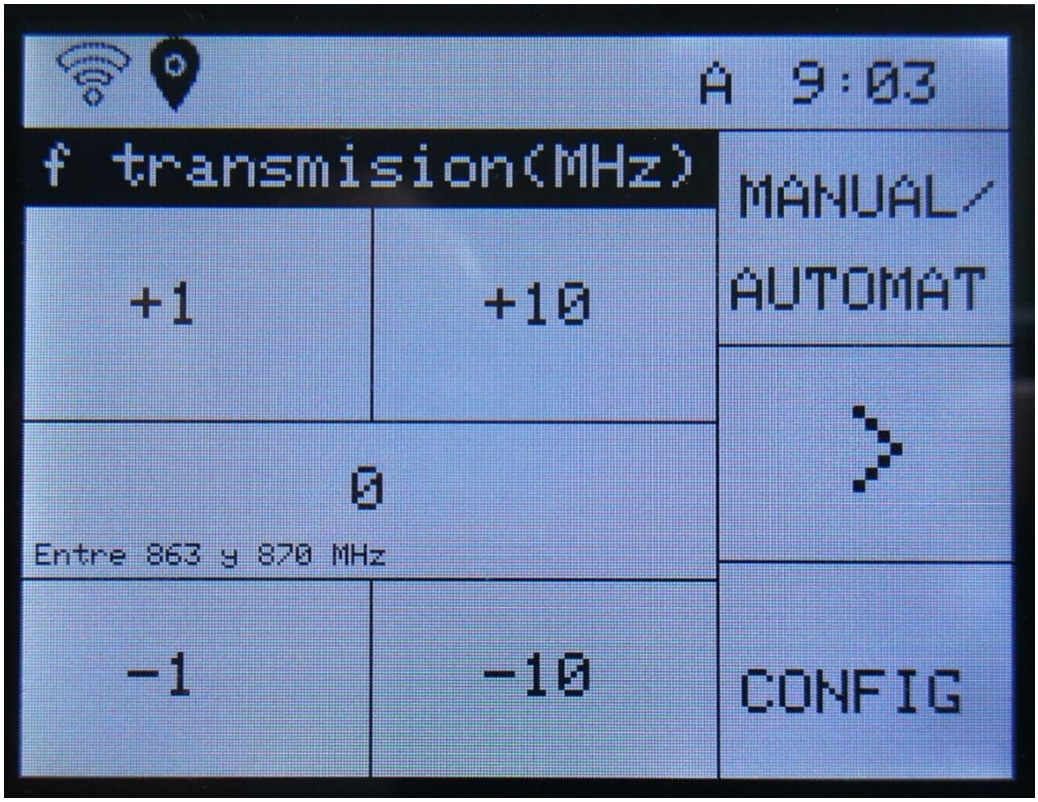

Ilustración 48. Pantalla de configuración de frecuencia de transmisión

- Pulsando “f de recepcion”

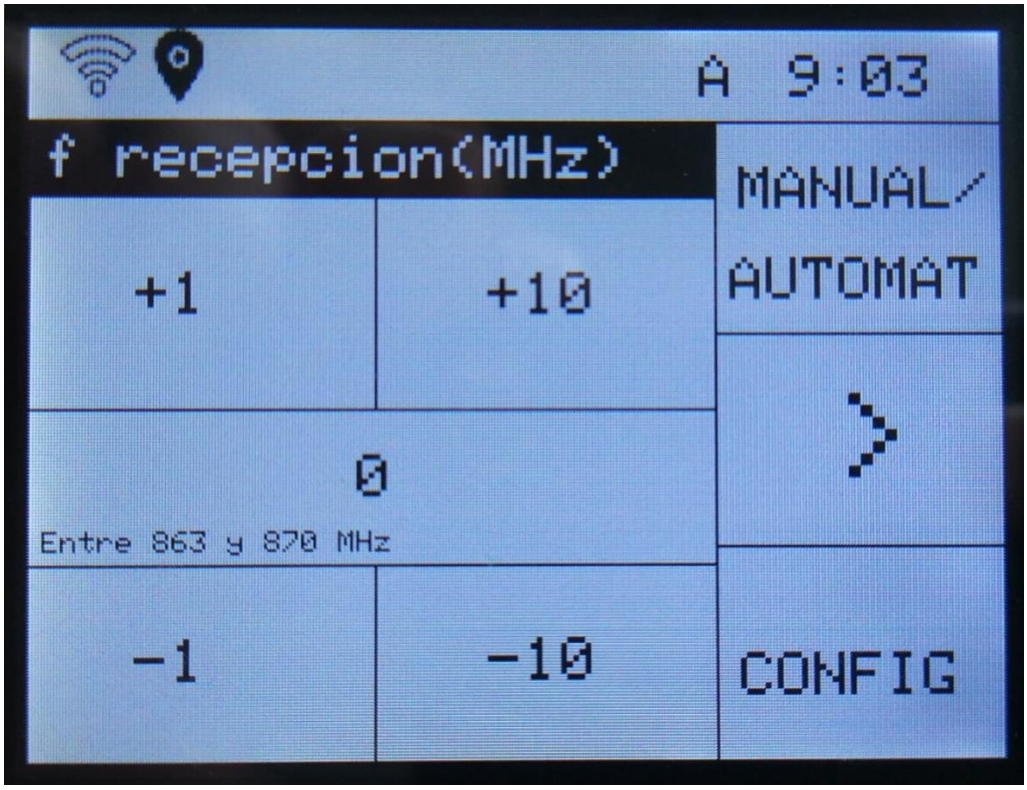

Ilustración 49. Pantalla de configuración de frecuencia de recepción

CAPÍTULO 5

MONTAJE FINAL Y PRUEBAS DE CAMPO

En este capítulo se va a describir el proceso de montaje de la PCB paso a paso. Incluyendo fotografías de cada etapa del montaje de la placa desde su inicio hasta su finalización con todos los componentes soldados y en correcto funcionamiento.

5.1.Proceso de montaje

En primer lugar, se sueldan todas las resistencias y transistores SMD y a continuación las tiras de pines.

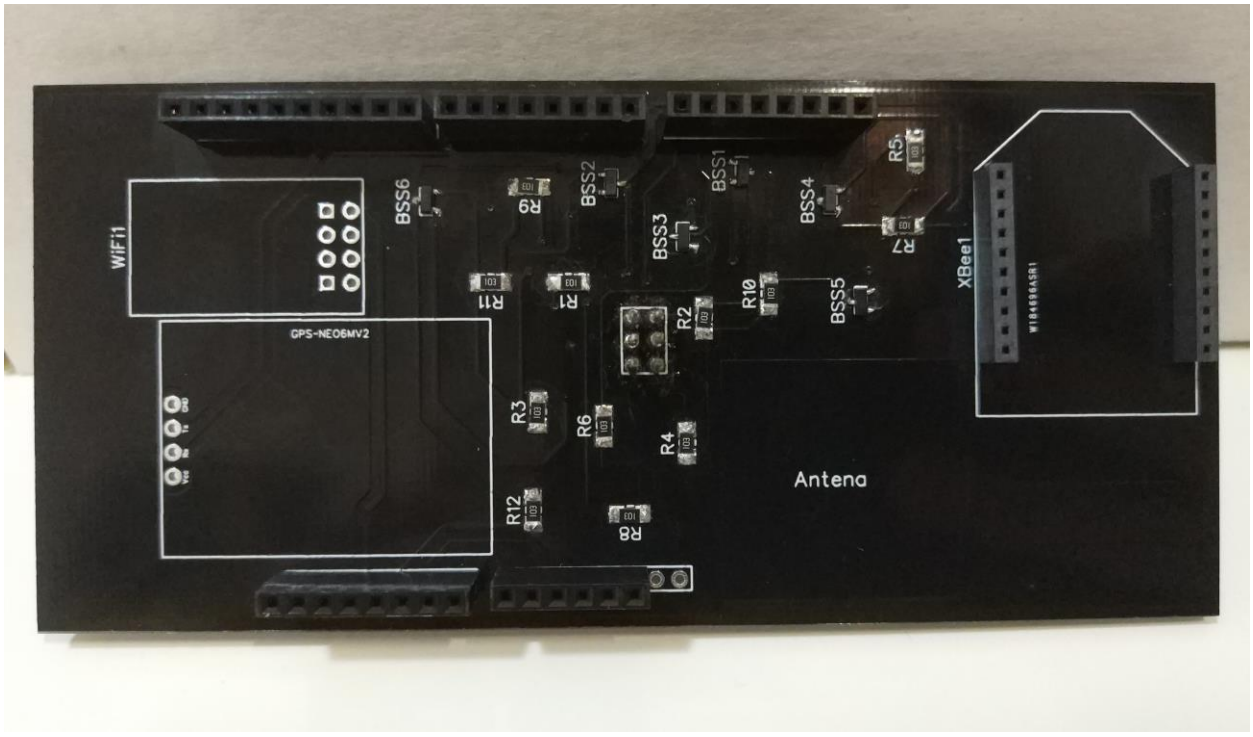

Ilustración 50. Montaje PCB con componentes SMD

Previamente a soldar el resto de shields, se ha probado la continuidad de los pads de los componentes con sus respectivas conexiones con Arduino MEGA. Además, con un sencillo programa se ha comprobado que los componentes SMD están soldados correctamente, midiendo con el polímetro y asegurándonos que las tensiones en cada punto son las correctas. Una vez que se han hecho estas comprobaciones se puede proceder a soldar el resto de sheilds.

Ilustración 51. Montaje PCB con los shields

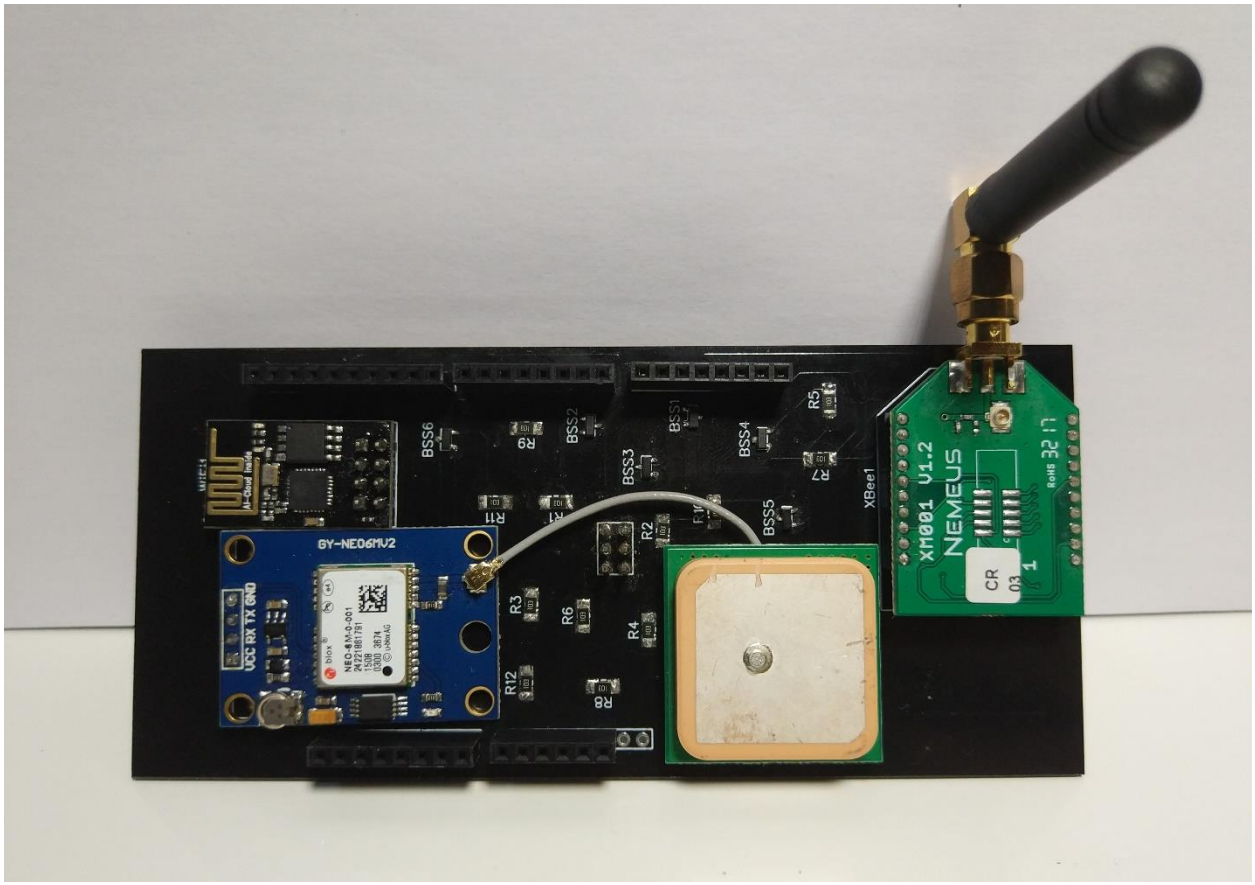

Ilustración 52. Montaje PCB con los shields II

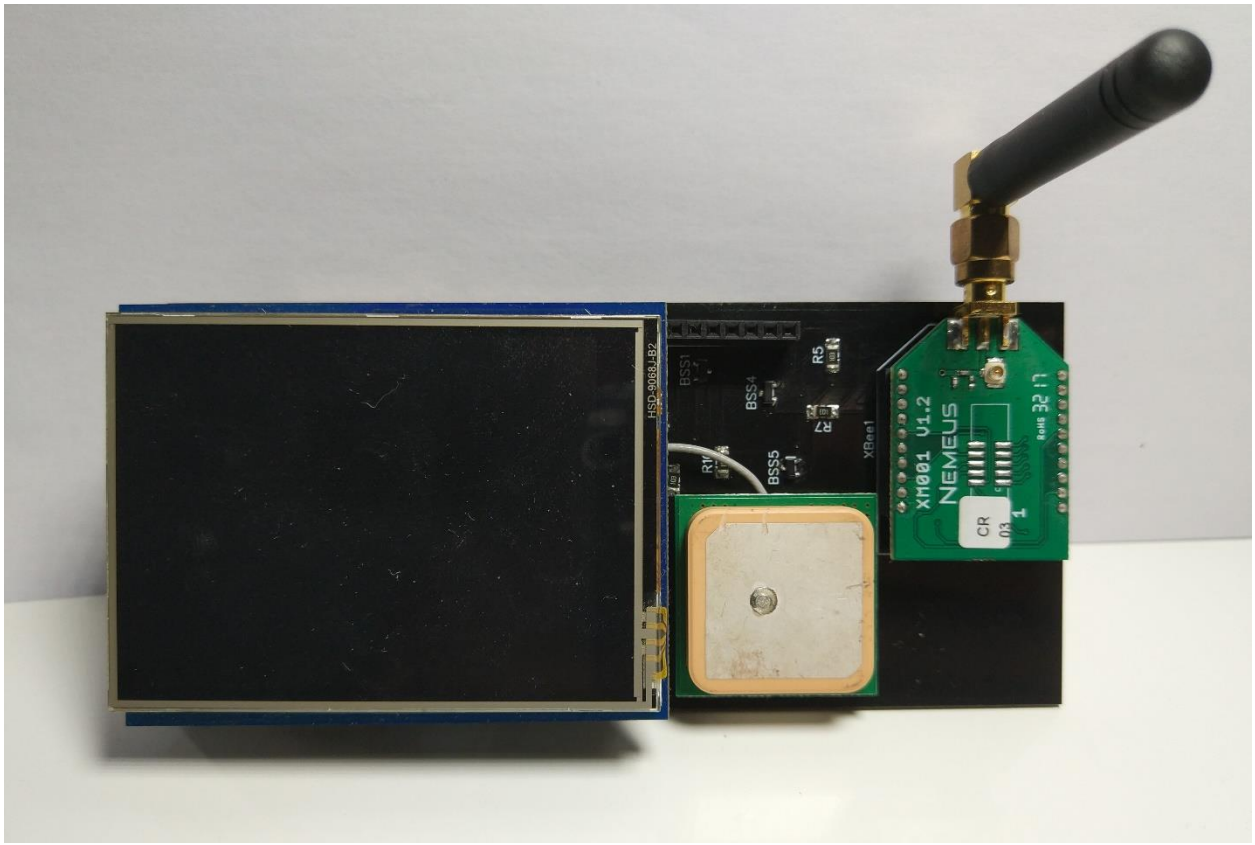

Ilustración 53. Montaje PCB con pantalla

Para conseguir ganar espacio, la pantalla se coloca encima del shield del WiFi y del GPS como se puede apreciar en las anteriores fotografías. La antena del GPS se deja el exterior para evitar que la TFT le haga apantallamiento, interfiriendo en la señal del GPS. Como se puede apreciar a continuación, este es el resultado final del prototipo con todos los shields y con Arduino MEGA:

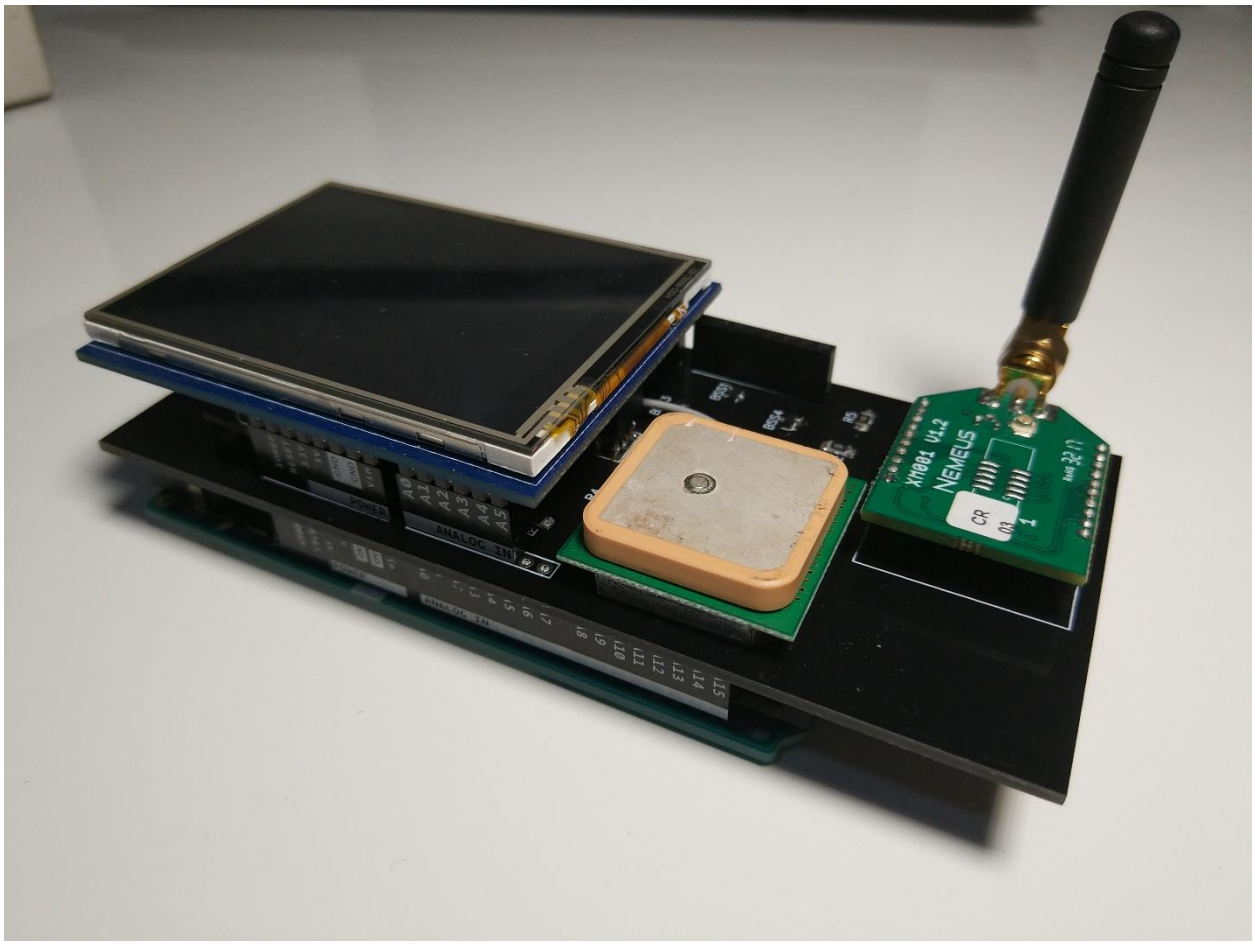

Ilustración 54. Montaje PCB con Arduino MEGA

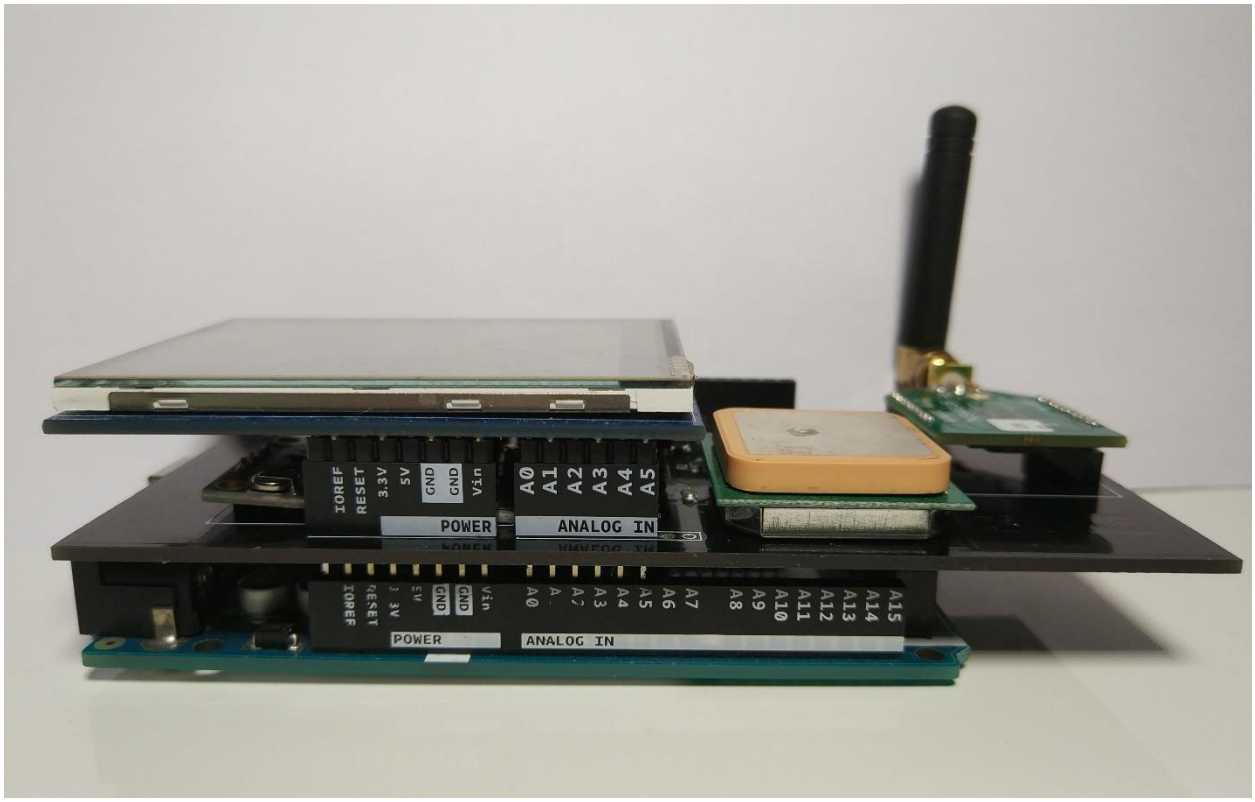

Ilustración 55. Montaje PCB con Arduino MEGA II

5.2. Pruebas de campo

Las pruebas de campo se han realizado encendiendo el dispositivo y esperando a que el dispositivo encuentre posición. Una vez se ha encontrado, las tramas de Sigfox se almacenan junto a la latitud y la longitud a la hora y la fecha en la que se ha recibido el nivel de señal devuelto. También se ha comprobado el correcto funcionamiento al cambiar las opciones de configuración de envío de las tramas, como la potencia y la frecuencia de transmisión y de recepción. Aunque en este caso se ha dejado la configuración por defecto para las pruebas.

En la SD que se encuentra en el shield de la pantalla táctil es donde se almacenará toda la información recopilada por el dispositivo. Toda esta información se almacenará en la SD en un fichero de texto del tipo .txt con el siguiente formato:

dd/mm/aaaa,hh:mm,latitud,longitud,rssi (de la red), tapID (Identificador de la antena),rssi (de la estación base)

29/9/2018,9:02,37.991863,-1.141160,-81,0765,-9C

data.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
5/10/2018,11:01,37.99,-1.14,-82.00,0765,-8E
5/10/2018,11:10,37.99,-1.14,-75.00,0765,-99
5/10/2018,11:17,37.99,-1.14,-84.00,0765,-94
5/10/2018,11:32,37.99,-1.14,-89.00,0765,-93
5/10/2018,11:45,37.99,-1.14,-77.00,0765,-9C
5/10/2018,11:54,37.99,-1.14,-81.00,0765,-99
5/10/2018,12:04,37.99,-1.14,-78.00,0765,-9D
5/10/2018,12:16,37.99,-1.14,-79.00,0765,-9A
5/10/2018,12:27,37.99,-1.14,-81.00,0765,-97
5/10/2018,12:37,37.99,-1.14,-82.00,0765,-99
```

Ilustración 56. Almacenamiento en la SD

Una de las pruebas de campo interesantes, así como una de las posibles aplicaciones para ver el comportamiento del dispositivo es colocándolo en un vehículo en modo automático de forma que vaya enviando y recogiendo datos de cobertura de varios puntos a lo largo de la ciudad. Teniendo así, datos reales de cobertura de la red de Sigfox en diferentes posiciones, pudiendo así hacer un mapa de cobertura.

Ilustración 57. Dispositivo en vehículo

CAPÍTULO 6

PRESUPUESTO

En este capítulo se va a mostrar el presupuesto de coste fijo como son los materiales necesarios para llevar a cabo dicho proyecto son:

	Referencia	Coste (€/und)	Cantidad	Precio	Total
Shields	Arduino MEGA	41,75 €	1	41,75 €	179,93 €
	Módulo GPS NEO6MV2	22,00 €	1	22,00 €	
	Módulo WIFI ESP8266	7,70 €	1	7,70 €	
	Conversor lógico de nivel 3,3-5V	2,77 €	1	2,77 €	
	Cables Conexión Macho/Hembra	1,85 €	1	1,85 €	
	Pantalla táctil 2,8"	50,86 €	1	50,86 €	
	Radio LPWAN Dual	35,00 €	1	35,00 €	
	Xbee shield (adaptador)	18,00 €	1	18,00 €	
PCB	PCB	8,40 €	5	42,00 €	81,92 €
	Transistor MOSFET	0,1750 €	10	1,75 €	
	Resistencia 10K SMD	0,0155 €	20	0,31 €	
	Conector placa a placa de 10 contactos (paso de 2mm)	1,00 €	3	3,00 €	
	Conector placa a placa de 10 contactos	2,60 €	2	5,20 €	
	Conector placa a placa de 8 contactos	2,0900 €	6	12,54 €	
	Conector placa a placa de 6 contactos (2 filas)	3,4245 €	5	17,12 €	
	TOTAL	**261,85 €**			

*La radio incluye un año de conectividad gratuita.

**Por defecto se considerarán de 1 fila y 2,54mm de paso salvo que se indique lo contrario.

CAPÍTULO 7

CONCLUSIONES Y TRABAJOS FUTUROS

7.1. Conclusiones

En este proyecto se ha desarrollado un dispositivo capaz de obtener el nivel de señal devuelto por la trama de Sigfox y acompañarlo de la posición GPS en la que se encuentra el dispositivo. Con esto se ha conseguido hacer un dispositivo capaz de hacer un mapa de cobertura y almacenarlos en una memoria SD.

Como vemos en el apartado 2.2 (Estado del arte de dispositivos de prueba de campo de las redes LPWAN) ya existen dispositivos que realizan estas funciones. Debido a esto, en nuestro proyecto se ha tratado de diseñar un dispositivo que supere a este, dotándolo de ciertas funcionalidades del cual este carece.

Para conseguirlo se ha hecho un estudio de mercado de una gran cantidad de placas de desarrollo microcontrolados eligiendo la que mejor se ajustaba a las especificaciones necesarias para el desarrollo del dispositivo, Arduino MEGA. Acompañado de los otros shields y tras implementarlos individualmente, se ha desarrollado una PCB adaptadora para todos los shields de forma que puedan ser funcionales a la vez implementando el software conjunto. De esta forma, se han conseguido los objetivos iniciales del proyecto. Consiguiendo desarrollar un producto que cumple los requisitos iniciales y con más funcionalidades que los dispositivos ya existentes. Añadiendo una pantalla táctil de mayor tamaño y mayor cantidad de imagen. Además, mediante la interfaz de usuario desarrollada pueden modificarse ciertos parámetros como son la potencia de transmisión, la frecuencia de transmisión y recepción y el intervalo de tiempo en el que se manda una trama sin necesidad de conectar el dispositivo al ordenador. Además de ser mucho más intuitivo y aportando datos que el otro dispositivo no muestra.

7.2. Trabajos futuros

Con todo lo expuesto anteriormente, se podría decir que, partiendo del proyecto realizado y desarrollado en esta memoria, se podrían tomar muchas vías diferentes de desarrollo, es decir, podría haber muchas variantes según las modificaciones, cambios o implementaciones aplicada al proyecto original aquí desarrollado.

Además, lo que se ha desarrollado en este proyecto es un prototipo, por lo tanto, está abierto a una gran cantidad de mejorías hasta llegar a ser un producto final capaz de comercializarse.

Una mejoría significativa a nivel hardware sería la implementación de una PCB con el microcontrolador ATmega2560 del Arduino MEGA y todos los shields integrados en una misma placa, sin necesidad de conectar los shields como se hace en este prototipo. De esta forma se ganaría una gran cantidad de espacio, consiguiendo un tamaño mucho más reducido. Sería mucho más robusto y económico ya que los precios de los shields son más elevados que el precio de los componentes por separado y haciendo la implementación hardware de una PCB, lo que requiere un trabajo y complejidad mucho mayor.

A lo que software se refiere también está abierto a una gran mejoría. Una de las partes a mejorar es la página web donde se envían todos los datos almacenados en la SD, de forma que se puedan visualizar en tiempo real. Debido a la gran cantidad de datos la página web tarda un tiempo considerable en cargarse, por lo que, se deberían cargar únicamente los últimos datos almacenados y poder ir retrocediendo en función de los datos que se quieran visualizar. Una posibilidad sería una flecha que te va llevando a los datos almacenados previamente o con un menú desplegable donde poder seleccionar que datos se desea ver en función de la fecha. Pudiendo así ver todos los datos del último mes, del último día, de la última hora o del último minuto. Consiguiendo así una página web mucha más eficiente y completa. Otra gran mejoría sería que ya que la antena de radio es dual y permite la comunicación con redes Sigfox y LoRa se implementara la parte software de LoRa, pudiendo elegir con que red se desea hacer la comunicación. Pudiendo de esta forma mejorar la información que se expone al usuario. Con los datos que se van recogiendo de cobertura de las redes LPWAN de Sigfox y de LoRa se mostraría al usuario un mapa de cobertura. Incluso la implementación de una app móvil de forma que el usuario pudiera visualizar los datos que se recogen en tiempo real en su aplicación móvil, así como los puntos donde no hay cobertura.

BIBLIOGRAFÍA Y REFERENCIAS

Referencias a artículos científicos

M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "*Long-Range Communications in Unlicensed Bands: the Rising Stars in the IoT and Smart City Scenarios*", IEEE Wireless Communications, Vol. 23, Oct. 2016.

J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "*Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions*" Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, Sept. 2013. http://dx.doi.org/10.1016/j.future.2013.01.010

A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, *"Internet of Things for Smart Cities*" IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22–32, Febr. 2014.

Mekki, Kais, Bajic, Eddy, Chaxel, Frédéric, Meyer, Fernand. (2018). "*Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT*". 2nd IEEE International Workshop on Mobile and Pervasive Internet of the Things. Athens, Greece. March 2018.

Sigfox's ecosystem delivers the worlds first ultra-low cost modules to fuel the internet of things mass market deployment. [Online]. Available: https://www.sigfox.com/en/press/ sigfox-s-ecosystem-delivers-world-s-first-ultra-low-cost-modules-to-fuel-internet-of-things.

Referencias web

Página web de Arduino:
https://www.arduino.cc/

Página web de Sofia2 IoT Platform:
https://about.sofia2.com/2014/11/26/que-es-sigfox/

Página web de Adeunis:
https://www.adeunis.com/en/produit/ftd-868-915-2/

Página web de Naylamp Mechatronics:
https://naylampmechatronics.com/blog/21_Tutorial-ESP8266-Parte-I.html

Página web de Premetec:
https://www.prometec.net/servidor-web-esp8266/

Página web de Nemeus (Hardware Overview):
http://wiki.nemeus.fr/index.php?title=Application_Note:_MM002_Hardware_Overview

Página web de Nemeus (Personalization):
http://wiki.nemeus.fr/index.php?title=MM002-xx-EU_Personalization#SigFox.E2.84.A2_Personalization

Página web de Nemeus (AT Commands & UART configuration):
http://wiki.nemeus.fr/index.php?title=MM002-xx-EU_AT_Commands#UART_configuration

Página web de pycom:
https://pycom.io/product/sipy/