Medidor geolocalizado de pH para control de emergencias medioambientales.

Sergio Pelayo Jimeno
Grado en Ingeniería Informática
Arduino

Consultor: **Antoni Morell Pérez**
Profesor: **Pere Tuset Peiró**

07/Enero/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Medidor geolocalizado de pH para control de emergencias medioambientales.*
Nombre del autor:	*Sergio Pelayo Jimeno*
Nombre del consultor/a:	*Antoni Morell Pérez*
Nombre del PRA:	*Pere Tuset Peiró*
Fecha de entrega (mm/aaaa):	01/2018
Titulación::	*Grado Ingeniería Informática*
Área del Trabajo Final:	*Arduino*
Idioma del trabajo:	*Español*
Palabras clave	*Arduino, GSM,SIM,PH,GPS*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El proyecto quiere abordar los problemas derivados de incidencias con vertidos hídricos en las empresas mediante la monitorización del valor de pH del agua. Éstos vertidos son detectados en general tarde y por tanto difíciles de controlar, con el dispositivo físico base del proyecto y la plataforma web asociada se quiere introducir en las empresas de una manera económica, eficaz y fácil una forma de control para todas las posibles incidencias tanto en alerta temprana como en seguimiento de incidencias graves.

El dispositivo se desarrolla en Arduino y se comunica con servidores web ubicados en Internet (o a través de él) mediante el chip SIM900 integrado en shield. Además se ha dotado al dispositivo de un receptor GPS capaz de mantener geolocalizado el sistema de monitorización, de manera que facilita la lectura y análisis de datos a personas no presentes en las instalaciones.

Todo el sistema se interconecta mediante una placa PCB fabricada ex profeso para ello; igualmente se crea un prototipo real y funcional usando piezas creadas a tal fin mediante impresión 3D.

Con el dispositivo se logra reducir considerablemente los tiempos de respuesta ante emergencias medioambientales a la vez que consigue monitorizar zonas hasta ahora no posibles.

En conclusión, se puede afirmar que la llegada de dispositivos desarrollados con Arduino hacen posible la creación de éste proyecto de manera económica y rápida al contrario de lo que ocurría hasta ahora que eran necesario costosas y voluminosas instalaciones para obtener el mismo (o peor) resultado.

Abstract (in English, 250 words or less):

The project wants to resolve arising problems from incidents with water spills in companies by monitoring the pH value of water. These spills are detected in general late and therefore difficult to control, with the physical device of the project and the associated web platform, the project wants to introduce companies in an economic, effective and easy manner way of controlling all possible incidents: early warning and follow-up of serious incidents.

The device is developed on Arduino and communicates with web servers located on the Internet (or through it) using the SIM900 chip integrated in a shield. In addition, the device has been equipped with a GPS receiver capable of keeping the monitoring system geolocated, in such a way that it facilitates the reading and analysis of data to people not present in the companies.

The entire system is interconnected using a PCB board manufactured in a professional way; Likewise, a real and functional prototype is created using pieces for this purpose applying 3D printing.

The device and system reduce the response times to environmental emergencies while at the same time monitoring areas that have not yet been possible.

In conclusion, we can confirm that the arrival of devices developed with Arduino make possible to create this project economically and quickly, contrary to what happened until now that expensive and bulky facilities were needed to obtain the same (or worse) results.

Contenido

Ilustraciones

Objeto

El presente proyecto tiene por objeto minimizar los costes económicos directos e indirectos y costes medioambientales producidos por las posibles incidencias de cualquier proceso productivo mediante la medición en continuo de los valores de pH de los vertidos; para focalizar el proyecto se ha decidido hacer una implantación en la factoría Acerinox Europa SAU sita en Palmones (Los Barrios – Cádiz).

Descripción

ACERINOX es una de las empresas más competitivas del mundo en la fabricación de aceros inoxidables. Desde su constitución, ha venido realizando un continuo programa de inversiones, con desarrollo de innovaciones tecnológicas propias que, en algunos casos, han constituido un verdadero hito en la tecnología de los aceros inoxidables.

Imagen. Plano aéreo Planta Acerinox Europa SAU.

El proyecto pretende seguir en línea con la responsabilidad social de la empresa en el Campo de Gibraltar para así dotar de medios técnicos de vanguardia en el control de vertidos no deseados.

El equipo objeto del presente proyecto esta implementado con un dispositivo Arduino al que se conectan sensores de pH, Leds de control y un shield de conexión GSM. Dicho equipo es el emisor de la información cuyos valores quedarán registrados en un servidor WEB con PHP cara a poner a disposición a distintos tipos de aplicaciones en la lectura de información. La lectura del registro marcado por el dispositivo se va a plasmar inicialmente en una plataforma web accesible desde cualquier dispositivo con conexión a Internet. En ella se verán marcados los valores de geolocalización del dispositivo así como el valor y hora de lectura del mismo entre otros datos de actividad disponibles y que se explicarán a lo largo del proyecto.
A continuación se van a exponer los principales elementos del dispositivo principal del proyecto.

¿Qué es Arduino?

Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como hobby y para cualquiera interesado en crear objetos o entornos interactivos.
Arduino puede "sentir" el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores y otros artefactos. El microcontrolador de la placa se programa usando el Arduino Programming Language (basado en Wiring) y el Arduino Development Environment (basado en Processing). Los proyectos de Arduino pueden ser autónomos o se pueden comunicar con software en ejecución en un ordenador (por ejemplo con Flash, Processing, MaxMSP, etc.).

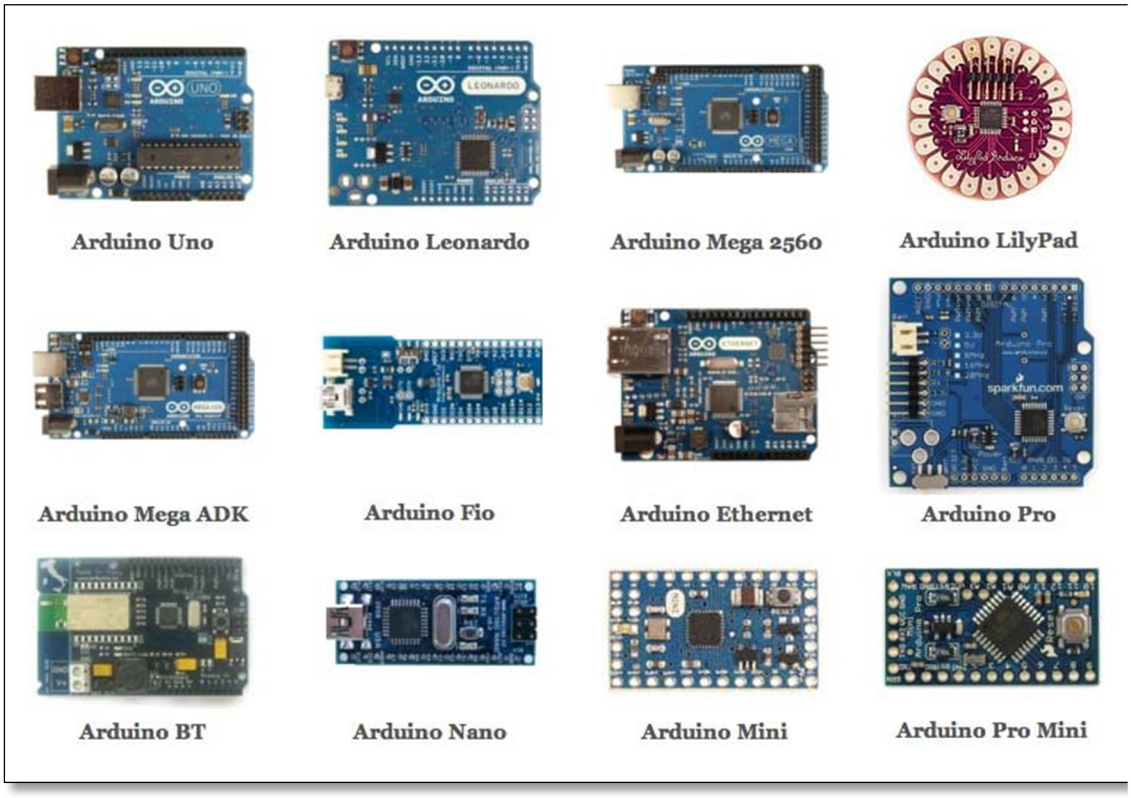
¿Por qué Arduino?

Hay muchos otros microcontroladores y plataformas microcontroladoras disponibles para computación física. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, y muchas otras ofertas de funcionalidad similar. Todas estas herramientas toman los desordenados detalles de la programación de microcontrolador y la encierran en un paquete fácil de usar. Arduino también simplifica el proceso de trabajo con microcontroladores, pero ofrece algunas ventajas sobre otros sistemas:

- **Barato**: Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión menos cara del módulo Arduino puede ser ensamblada a mano, e incluso los módulos de Arduino preensamblados cuestan menos de 50€.
- **Multiplataforma**: El software de Arduino se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux. La mayoría de los sistemas microcontroladores están limitados a Windows.
- **Entorno de programación simple y claro**: El entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para que usuarios avanzados puedan aprovecharlo también..
- **Código abierto y software extensible**: El software Arduino está publicado como herramientas de código abierto, disponible para extensión por programadores experimentados y por tanto podremos ver hasta los últimos detalles de la raíz del sistema.
- **Código abierto y hardware extensible**: El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que podríamos hacer la propia versión del módulo, extendiéndolo y mejorándolo.

Fuente: http://arduino.cl

Gracias a las principales ventajas detalladas anteriormente y a la gran extensión que ha tenido en los últimos años, se ha facilitado a las empresas el acceso al IoT (Internet of Things) a un coste muy por debajo de la inversión que se hacía hasta ahora. La diversificación de formas de conexión necesaria y la autonomía de las mismas hace incrementar el valor total de aplicación en microcontroladoras en general y en Arduino en particular.



Algunos modelos de placas Arduino.

En estudios posteriores se detallará de forma específica el conexionado y la forma de alimentación del mismo así como una justificación más completa de la elección de la placa.

Sensores de entrada y salida de datos.

Como se enuncia en el título del proyecto, el dispositivo posee principalmente los siguientes dispositivos: uno de entrada de datos (medición de pH) , otro de entrada (recepción de datos de Red y posición GPS) y otro de salida (escritura de datos o comunicación con servidores). A parte de los principales harán falta algunos elementos de salida visual básicos como por ejemplo Leds de control.

Medición de pH

El pH es una medida de acidez o alcalinidad de una disolución. El pH indica la concentración de iones hidronio [H3O+] presentes en determinadas sustancias. El sensor debe permitir medir de forma sencilla el pH de un líquido usando un controlador. El controlador debe tener un potenciómetro multivuelta que permita la correcta calibración de la sonda.
Este punto de estudio (posteriormente más a fondo) tendrá en cuenta los esquemas de placa y sensor según las necesidades de la empresa de implantación; dependiendo del tiempo de exposición al medio líquido se optará por un tipo u otro. Actualmente en el mercado se pueden encontrar dos tipos de sensores: sensor para mediciones de corta duración (básico) y sensor para medición de exposición larga (profesional).

Sensores de medición de pH

En el mercado existe un amplio espectro de sondas y controladores de pH, existen algunos de bajo coste (básicos) que permiten mediciones puntuales, sin embargo con un uso prolongado pueden dar valores no correctos; por otro lado existe la posibilidad de usar sondas profesionales que mantienen constante su función en el tiempo y por tanto hacen que sea posible un uso prolongado con total fiabilidad.
Teniendo en cuenta este aspecto junto a las variables de entorno posibles en cada instalación (limpieza del agua, accesibilidad…), se elegirá la sonda para este proyecto.

Sensor analógico de pH (Básico)

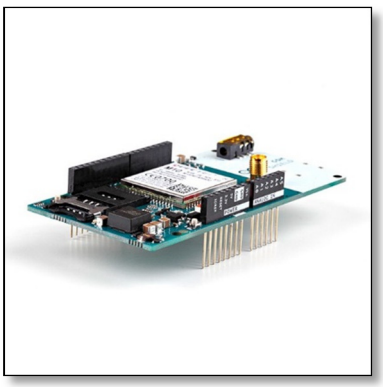
Sensor Profesional medidor de pH

Conexión GPS/GPRS/GSM

Este punto es sin duda, junto al anterior, la base del proyecto; es necesaria una conexión con un servidor (en este caso WEB) para registrar los datos y que puedan ser posteriormente consultados. Igualmente necesitaremos un sensor GPS que dependiendo del sistema final elegido lo haremos independiente o compartido con el propio sistema de conexión al servidor.
Existen en el mercado una gran variedad de sensores y tipos de conexiones, no obstante pensando en las dos opciones planteadas o bien escogemos un sistema que centralice los sensores GPS y conexión GSM o bien se buscan por separado.

Receptor GPS EM-506 (48 canales)

GPS/GPRS/GSM Shield V3.0



Arduino GSM Shield v2

Esquema general

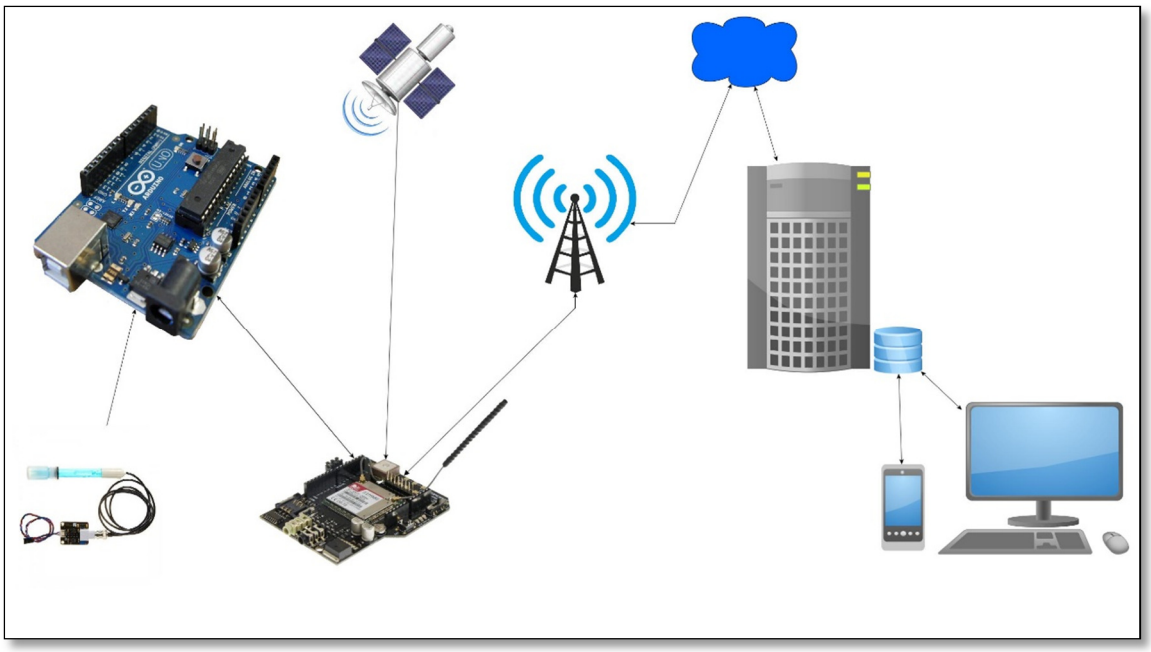


Ilustración de comunicaciones del sistema

El sensor pH tomará las muestras en los distintos puntos que queramos, la placa Arduino se conectará a través del shield de conexión al sistema GSM mediante 3G al servidor WEB que se configure a tal fin, le enviará de forma conjunta el valor medido así como el valor obtenido GPS; el sistema final se presentará vía web igualmente mostrada vía PC, MAC, móvil o Tablet.

Objetivos del proyecto

Tal y como se puede intuir en la descripción del proyecto, el principal objetivo es registrar los valores de pH en un servidor remoto, pero ¿qué ganamos con ello?
En primer lugar cabe enunciar el sistema de recogida de aguas de la empresa destino (en el caso Acerinox Europa SAU).

Tal y como se puede apreciar en la figura 1, la empresa cuenta con 3 colectores internos con salida al exterior y un 4 para recogida de aguas pluviales: A, B, C y D. Cada uno de ellos se encarga de recoger las aguas tratadas o limpias que pueden salir al exterior, igualmente los colectores A, C y D son monitorizados en puntos finales (antes de la salida a la Bahía).

El principal objetivo es mantener un dispositivo móvil geolocalizado capaz de registrar un dato de valor de pH fiable entre 1 y 2 minutos de separación entre muestras.

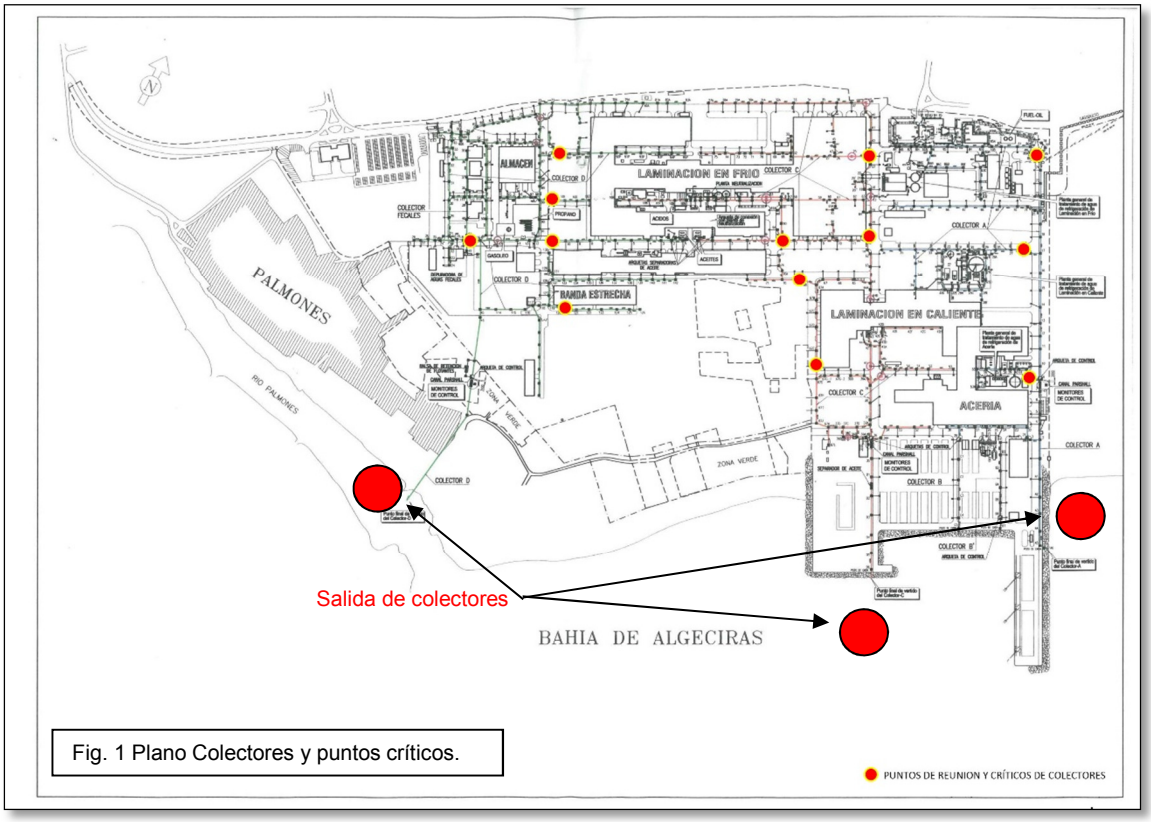


Fig. 1 Plano Colectores y puntos críticos.

Los problemas que aborda el proyecto son aquellos que se producen en algunas de estas situaciones:

1) Vertido ácido o alcalino no controlado en alguno de los colectores internos: cuando se produce este tipo de incidencia los técnicos del servicio de medio ambiente deben realizar un seguimiento manual en todos los puntos críticos (marcados con círculos rojos en la figura 1) de la empresa cara a minimizar el vertido real a la Bahía, dicho de otro modo, deben levantar arqueta por arqueta, tomar muestra, medir y seguir la búsqueda para acelerar el proceso de control. Si el vertido no es continuo puede ser que se tomen muestras con pH normal cuando la realidad es que minutos más tarde el pH vuelve a tener valores indeseados. Si por el contrario activamos el dispositivo creado en este proyecto en las arquetas clave del colector podremos tener la certeza de tener el máximo control durante la emergencia y guiar de manera mucho más efectiva a los cuerpos de intervención, reduciendo costes económicos asociados (tiempo de camiones de alto vacío, cantidad Bicarbonato sódico para neutralizado del vertido, tiempo de personal técnico) y quitando volumen de vertido real.
2) La segunda situación se dá ante la falta de control antes de los puntos finales fijos. La empresa posee tres puntos de monitorización fijos situados en las ramas finales y por tanto no dejan tiempo de maniobra, la premura en la actuación del vertido ya se vio en el punto anterior, pero ¿por qué quedarnos ahí? y ¿por qué no prevenir? La gran ventaja de este proyecto reside en el bajo coste de la creación del dispositivo y su fácil instalación; éstos podrían servir como puntos de medición móviles puestos a voluntad y demanda según las circunstancias de factoría. Un ejemplo de ello se puede producir cuando ante los preparativos de una máquina o línea de producción se prevén vaciar tanques o cubetos existe una mayor probabilidad de vertido; dotar de dispositivos de medición en los puntos de reunión o críticos cercanos sería de gran ayuda dando un

margen mucho mayor que los aportados por los puntos de medición fijos. La versatilidad que aporta un dispositivo de las características de este proyecto hace posible dotar de uno en cada punto crítico.

Fig.2 Sistema de monitorización fija.

3) Situaciones donde los responsables de medio ambiente por razones diversas no pueden asistir a la factoría; haciendo uso de los dispositivos geolocalizados y conectados pueden ver los valores desde cualquier parte con conexión a Internet de modo que pueden dar instrucciones de forma directa e inmediata. La toma de decisiones en este tipo de incidencias es crucial y poder ser asistidos de forma indirecta con valores reales de la emergencia o control del proceso supone una ventaja hasta ahora no implementada.

Estudio de viabilidad

A continuación se exponen los estudios de viabilidad legal, económica y técnica del proyecto.

Viabilidad legal

El dispositivo no cuenta con ningún tipo de impedimento legal para ser llevado a cabo; todos los elementos que lo componen (sensor GPS, shield GSM, LCD, placa Arduino..) no incumplen ningún imperativo legal.

Viabilidad económica

A continuación se detallan los costes asociados a la fabricación del dispositivo. Si bien el realizado físicamente durante este proyecto es un prototipo, los costes económicos de futuras versiones no se prevé que sean más caros (más bien todo lo contrario).

Los costes materiales aproximados de los elementos que intervienen en el dispositivo son:

	CON IVA	SIN IVA
Arduino UNO Rev.3	15,00 €	12,40 €
Sensor analógico de pH (Básico)	15,00 €	12,40 €
GSM Shield V3.0	20,00 €	16,53 €
Batería LiPo 6000mAh / 3.7V	30,19 €	24,95 €

Receptor GPS	12,00 €	9,92 €
Leds de control	0,06 €	0,05 €
Resistencias	0,05 €	0,04 €
Pantalla LCD	4,00 €	3,31 €
Placa PCB fotosensible positiva	15,00 €	12,40 €
Revelador (Parte proporcional)	1,00 €	0,83 €
Atacador (Parte proporcional)	2,00 €	1,65 €
Funda termo retráctil	1,00 €	0,83 €
Botones de configuración	0,10 €	0,08 €
Filamento Impresión Partes de la caja (parte proporcional)	3,00 €	2,48 €
Caja portadora	10,00 €	8,26 €
Interruptor	0,10 €	0,08 €
Adaptador USB-B: Hembra	0,30 €	0,25 €
Tornillería General	0,50 €	0,41 €
Estaño para soldar piezas	0,50 €	0,41 €

Total:	129,80 €	107,27 €

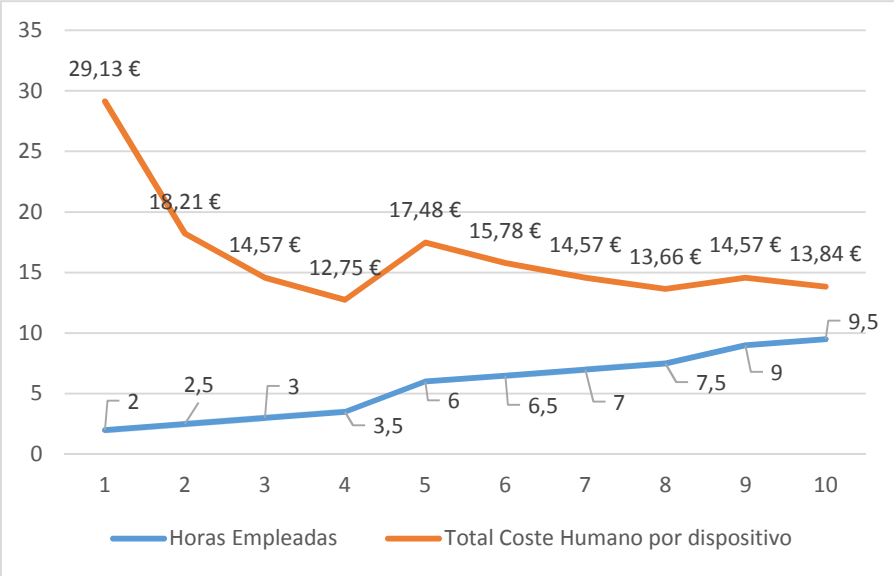
Los costes de contratación de tarjeta SIM para la comunicación y del equipo encargado de ser el servidor WEB de comunicación será encargo de la propia empresa.
Otro dato a tener en cuenta son los costes humanos asociados a la propia fabricación del dispositivo; durante la creación del proyecto se ha detectado un tiempo que oscila entre las 2-3 horas de trabajo para el prototipo. Haciendo un estudio más en detalle y según fuente https://www.indeed.es/salaries/T%C3%A9cnico/a-electronico/a-Salaries para el cálculo de salario bruto anual obtenemos la siguiente tabla de costes humanos:

Cantidad Dispositivos	Horas Empleadas	Total Coste Humano por dispositivo
1	2	29,13 €
2	2,5	18,21 €
3	3	14,57 €
4	3,5	12,75 €
5	6	17,48 €
6	6,5	15,78 €
7	7	14,57 €
8	7,5	13,66 €
9	9	14,57 €
10	9,5	13,84 €

Max horas anuales de trabajo	Salario Bruto anual
1826	20.000,00 €
Seg.Sociales Asociados (Empresa)	6.600,00 €
Porcentaje aportacion S.S. Empresa	33,00%

Coste Hora	14,57 €

Como se puede apreciar el precio coste humano no es lineal y no se duplica por cada dispositivo. Para dicho cálculo se ha tenido en cuenta la duplicidad en medios de creación de modo que un operador puede realizar de forma casi simultánea algunas de las tareas (más en detalle de 4 en 4); de esta manera se consigue un coste aceptable de montaje por dispositivo que ronda la media de 16€ cada uno.



Gráfica Costes: horas empleadas x coste humano.

Por último los costes de mantenimiento de la posible empresa distribuidora y pagadora de impuestos dependerán de su naturaleza; si nos basamos en la creación de una empresa de carácter autónomo, se deberán tener en cuenta las cuotas mensuales y la liquidación de impuestos asociados a beneficios. En este supuesto vamos a considerar una base de cotización mínima (sueldo del empresario) de 893€ y una liquidación del 29,8% dándonos un total mensual a pagar de 266,11€ con carácter fijo y obligatorio.

Base de cotización	893,00 €
% Cotización	29,80%
Total	266,11 €

En resumen obtenemos un cuadro para control del negocio que nos obligaría a vender con un margen del 40% sobre el valor de costes para pagar los gastos fijos:

Coste Material	129,80 €
Coste Montaje	15,99 €
Total gastos por dispositivo	145,79 €
Porcentaje Ganancia dispositivo	40,00%
Ganancia por dispositivo	58,31 €
Total PVP	**204,10 €**

Gastos Fijos Empresa	266,11 €

Dispositivos necesarios para cubrir gastos mensuales	**5**

Teniendo en cuenta que los equipos profesionales de medición oscilan de unos 300€ a 2000€ (sin conexión remota portátil), podemos asegurar que el dispositivo del proyecto es económico y de amortizable instalación en cualquier empresa del sector con las peculiaridades descritas.

Al tener margen suficiente de incremento de valor y no requerir una inversión inicial salvando la necesaria para construir el dispositivo, podemos asegurar que el proyecto tiene vía libre para su construcción en términos económicos.

Viabilidad técnica

En este apartado además de los propios sensores del dispositivo se debe tener en cuenta la infraestructura necesaria que debe tener la empresa; poseer un servidor WEB en Internet o tenerla de forma interna será una decisión básica de este apartado. Cada empresa puede imponer unos requisitos de implantación diferentes y que serán por tanto tenidos en cuenta de cara a la instalación.

Para la implantación total (con todas las características) del dispositivo la empresa deberá aportar un servidor WEB de dominio de la propia empresa y de la contratación (prepago o contrato) de servicio GSM con una operadora que permita la conexión a internet a través de una APN.

Al tratarse de una tarea realizable por el propio departamento de cada empresa este apartado será personalizado a cada petición.

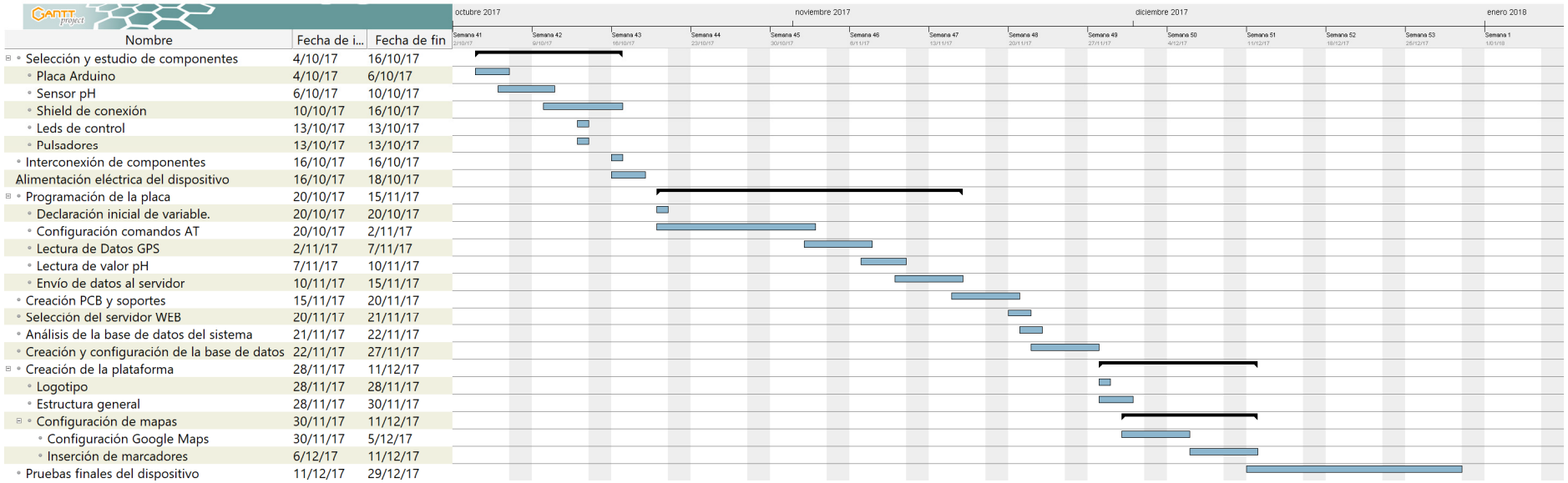
Índice de tareas del proyecto

A continuación se detalla la lista de tareas para la consecución de los objetivos del proyecto.

1. Selección y estudio de compontes.
    - Placa Arduino.
    - Sensor pH.
    - Conectividad red GSM.
    - GPS.
    - Leds, pantalla y pulsadores.
2. Interconexión de componentes.
3. Estudio de alimentación del dispositivo.
4. Programación de la placa.
5. Creación PCB.
    - Diseño.
    - Impresión.
    - Montaje.
6. Soporte del dispositivo.
    - Moldes y soportes.
7. Análisis de la BBDD.
    - Selección del sistema gestor de la BBDD.
    - Elección servidor y alojamiento.
8. Creación de la plataforma.
    - Logotipo.
    - Estructura general.
    - Creación de páginas/servicios.
    - Desarrollo e inserción API Google.
9. Pruebas de funcionamiento.
10. Creación y entrega de memoria.

Diagrama de Gantt

Se adjunta el diagrama que muestra el gráfico de barras horizontales ordenadas por actividades a realizar en secuencias de tiempo concretas para la consecución del proyecto:



Nombre	Fecha de i...	Fecha de fin
Selección y estudio de componentes	4/10/17	16/10/17
Placa Arduino	4/10/17	6/10/17
Sensor pH	6/10/17	10/10/17
Shield de conexión	10/10/17	16/10/17
Leds de control	13/10/17	13/10/17
Pulsadores	13/10/17	13/10/17
Interconexión de componentes	16/10/17	16/10/17
Alimentación eléctrica del dispositivo	16/10/17	18/10/17
Programación de la placa	20/10/17	15/11/17
Declaración inicial de variable.	20/10/17	20/10/17
Configuración comandos AT	20/10/17	2/11/17
Lectura de Datos GPS	2/11/17	7/11/17
Lectura de valor pH	7/11/17	10/11/17
Envío de datos al servidor	10/11/17	15/11/17
Creación PCB y soportes	15/11/17	20/11/17
Selección del servidor WEB	20/11/17	21/11/17
Análisis de la base de datos del sistema	21/11/17	22/11/17
Creación y configuración de la base de datos	22/11/17	27/11/17
Creación de la plataforma	28/11/17	11/12/17
Logotipo	28/11/17	28/11/17
Estructura general	28/11/17	30/11/17
Configuración de mapas	30/11/17	11/12/17
Configuración Google Maps	30/11/17	5/12/17
Inserción de marcadores	6/12/17	11/12/17
Pruebas finales del dispositivo	11/12/17	29/12/17

Diagrama de Gantt

Selección y estudio de componentes.

A continuación se van a detallar las partes seleccionadas para el desarrollo físico del proyecto; así pues y tal y como se adelantó las partes principales corresponden a: placa Arduino seleccionada, sensor pH, receptor GPS, placa GPRS. Igualmente durante el propio desarrollo se van a incorporar otros elementos de control igualmente imprescindibles tales como leds, resistencias, pantalla LCD y botones de configuración/acción.

Placa Arduino

La selección de la placa Arduino va a depender en base a las necesidades del proyecto y por tanto debemos analizar los siguientes puntos del mismo:

1) Necesidad de conexiones externas. Existe la posibilidad de elegir un Arduino, "Arduino Yun" para los casos en los que necesitamos conectividad Wifi. Este tipo de placas es muy recomendable en entornos donde no queremos depender de topologías externas. En nuestro caso la conectividad que queremos tener es usando la red GPRS y por tanto no será conectividad Wifi sino móvil usando Red de telefonía. Posibles segundas versiones podrían usar este tipo de dispositivos siempre y cuando cumpla con los siguientes requisitos.

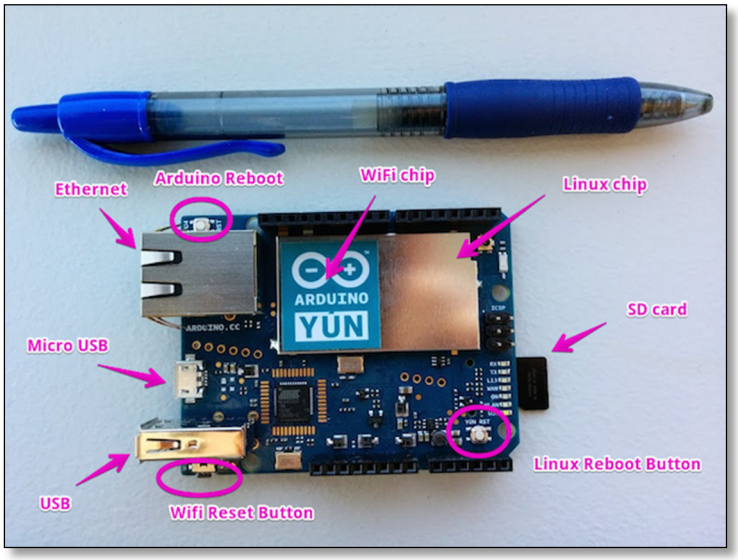
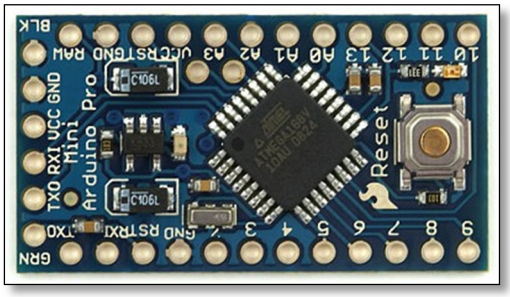


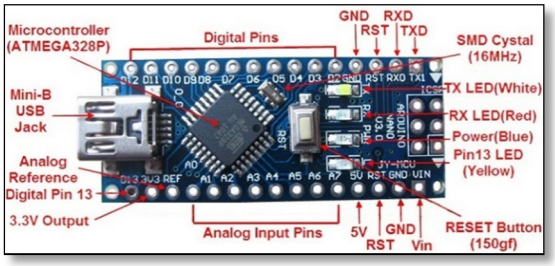
Imagen: Arduino Yun

Por tanto y dado que en la actualidad no existe una placa de Arduino que tenga de forma nativa conectividad móvil, nos decantamos con placas no especializadas.

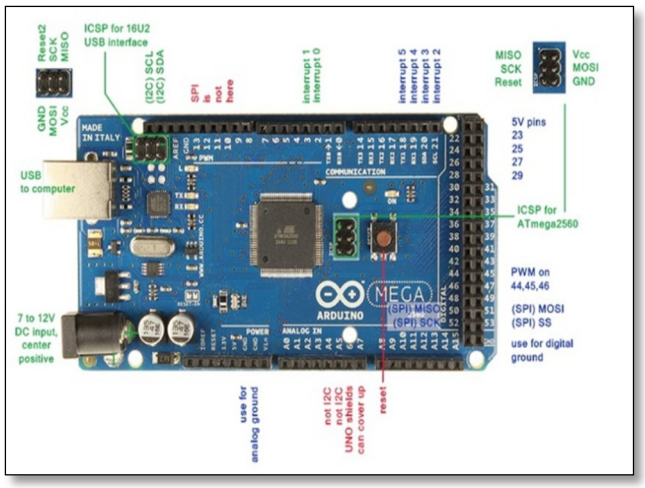
2) Tensión de alimentación: sin duda una parte importante será la forma de alimentación eléctrica que se suministrará. Nuestro proyecto es totalmente autónomo y portátil, por lo que la opción se va a restringir a alimentar de forma especializada la placa, no usará redes eléctricas caseras ni industriales. Necesariamente se deberá crear un sistema de baterías y gracias a ello esta opción no restringirá nuestra elección de placa.
3) Tamaño: el tamaño del dispositivo final será determinante para su usabilidad. Por tanto elegiremos una placa que no exceda mucho de tamaño, ajustaremos el número de pines a usar. Ejemplos de placas reducidas pueden ser Arduino Nano, Mini Pro en caso contrario tenemos Arduino Mega.



Arduino Mini Pro



Arduino Nano



Arduino Mega

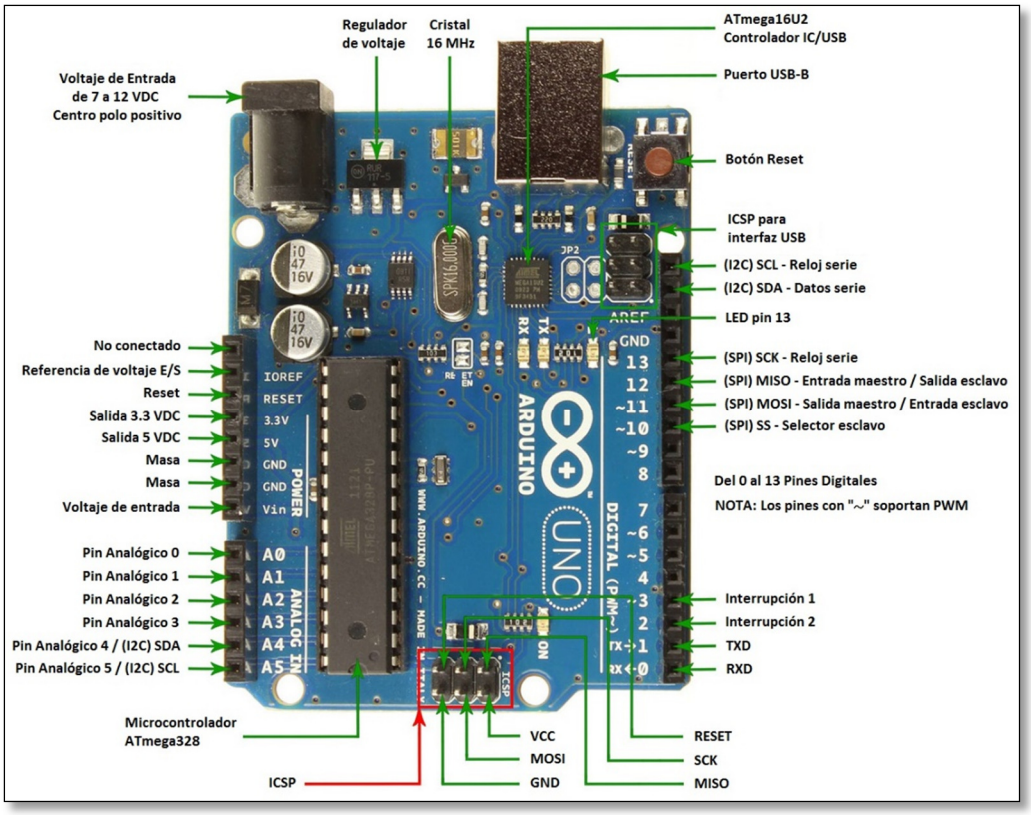
4) Número de entradas y salidas: sin duda este punto está íntimamente enlazado al anterior, se prevé un uso de al menos 9 entradas / salidas digitales, 1 analógica y uso de los pines I2C (que en la mayoría de placas se ubican en dos pines analógicos).
5) Tensiones LOW y HIGH: dependiendo del microcontrolador de la placa escogida las tensiones LOW o HIGH son detectadas por debajo de umbrales diferentes, así pues la familia de controladores ATMEGA32u4 detecta LOW cuando la tensión está por debajo de 1v mientras que ATMEGA328 lo hace por debajo de 3v. Este tipo de especialización no es determinante en el proyecto.
6) Precio de la placa: este punto es especialmente delicado en cualquier tipo de proyecto, ya que si los dispositivos se han desarrollado tanto en los últimos años es gracias a la accesibilidad en precio.

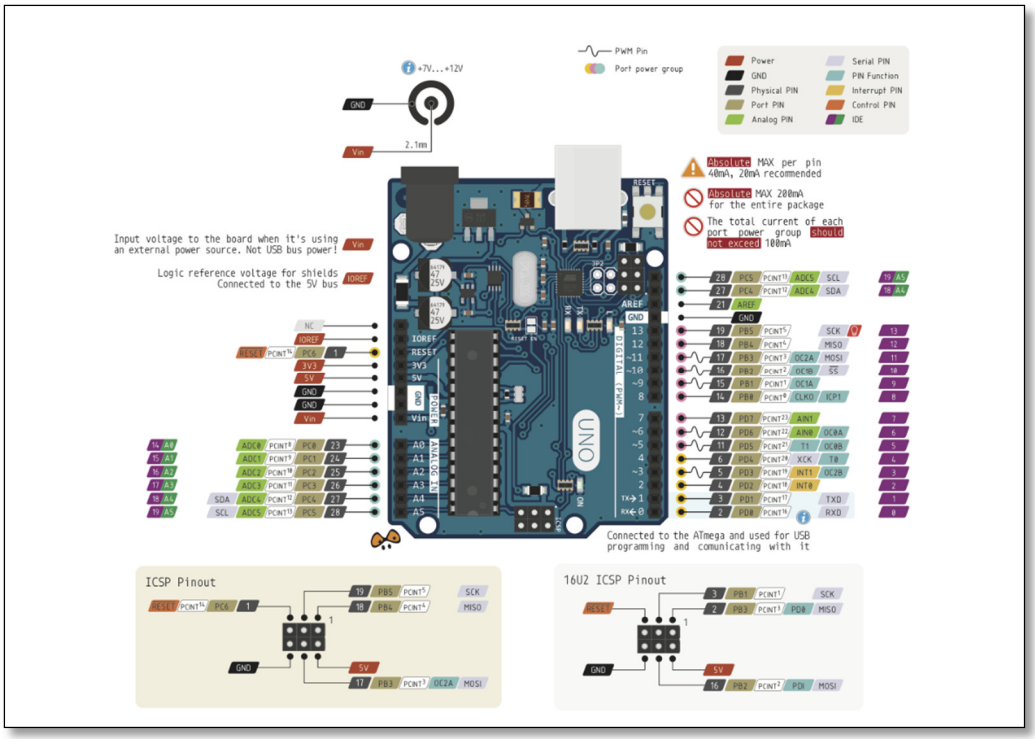
Atendiendo a los requisitos anteriormente expuestos el proyecto va a tener como base la placa Arduino UNO.
Esta placa se basa en el microcontrolador ATmega328 de ATMEL. Tiene 14 Entradas/Salidas digitales (6 de las cuales pueden utilizarse como salidas PWM), 6 entrada analógicas, conector USB, clavija de alimentación hembra tipo Jack, conector ICSP (para programar el microcontrolador sin necesidad de conectar la placa al PC) y botón de reset. Funciona a 16MHz.

Contiene todo lo necesario para el funcionamiento del microcontrolador, solo hay que conectarla al ordenador con un cable USB del tipo A-B o a una batería externa para que funcione como será en nuestro caso.

La versión UNO difiere de las versiones anteriores en que no utiliza el chip convertidor serie a USB FTDI (aunque algunas placas lo siguen manteniendo), en lugar de ello incorpora un ATmega16U2 programado como convertidor serie a USB. Este chip tiene USB nativo y puede ser reprogramado para que el Arduino sea reconocido al conectarlo al PC como cualquier tipo

de periférico USB (por ejemplo el ratón de nuestro PC). Esta característica hará más fácil su programación.





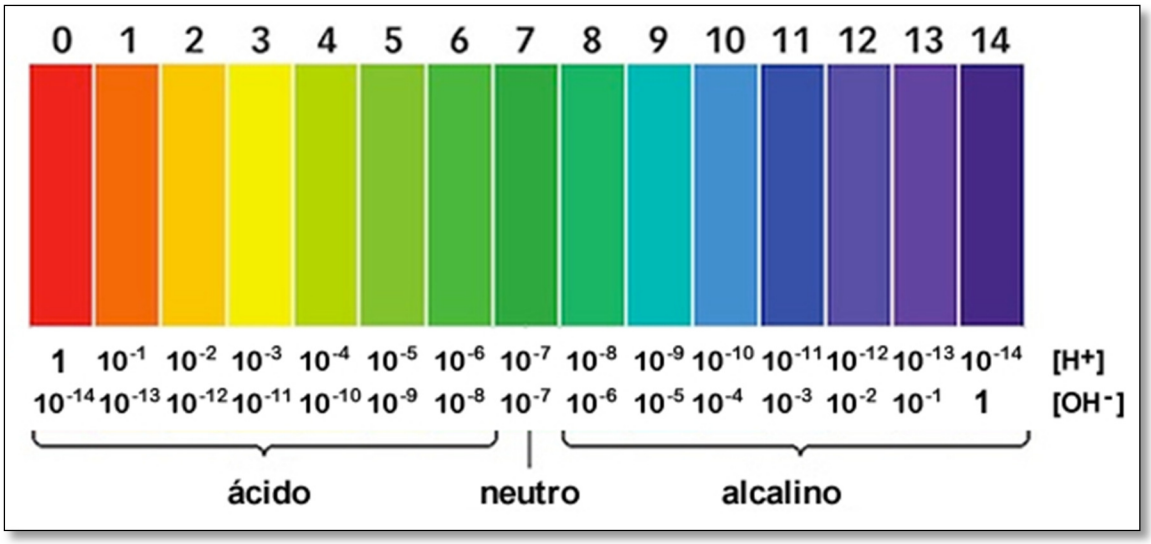
Esquema Arduino UNO Rev.3

Como se puede apreciar cumple con todos los requisitos previos, por tanto ésta será nuestra base del proyecto.

Sensor pH

¿Qué es el pH?

Sin duda alguna para adentrarnos en el mundo electrónico de sensores de pH debemos saber o al menos tener algunas nociones que lo que implica.
El pH es una medida de acidez o alcalinidad de una disolución. El pH indica la concentración de iones hidrógeno [H]+ presentes en determinadas disoluciones.
La sigla significa: potencial hidrógeno o potencial de hidrónes (pondus hydrogenii o potentia hydrogenii; del latín pondus, n. = peso; potentia, f. = potencia; hydrogenium,n. =hidrógeno). Este término fue acuñado por el bioquímico danés S. P. L. Sørensen (1868-1939), quien lo definió en 1909 como el opuesto del logaritmo en base 10 o el logaritmo negativo, de la actividad de los iones hidrógeno.
El término "pH" se ha utilizado universalmente por lo práctico que resulta para evitar el manejo de cifras largas y complejas. En disoluciones diluidas, en lugar de utilizar la actividad del ion hidrógeno, se le puede aproximar empleando la concentración molar del ion hidrógeno.
En disolución acuosa, la escala de pH varía, típicamente, de 0 a 14. Son ácidas las disoluciones con pH menores que 7 (el valor del exponente de la concentración es mayor, porque hay más iones hidrógeno en la disolución). Por otro lado, las disoluciones alcalinas tienen un pH superior a 7. La disolución se considera neutra cuando su pH es igual a 7, por ejemplo el agua.



Escala pH

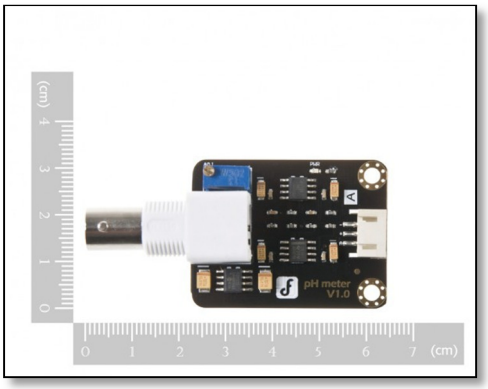
Selección y características

En el mercado hay una gran variedad de sondas y controladoras para conectar a Arduino, la principal característica a tener en cuenta en la selección de la misma radica en relación precisión/precio. A mayor precisión más costosa será la inversión en la sonda y viceversa. Así pues debemos buscar un controlador que se adecue a los medios donde se va a usar, en este caso, un entorno industrial.
La sonda elegida es la siguiente:

Características:

- Alimentación: 5.00V
- Dimensiones: 43x32mm (controlador)
- Rango de medición: 0-14 pH
- Temperatura de medición: 0-60 □
- Precisión: ± 0.1pH (25 □)
- Tiempo de respuesta: ≤ 1min
- Sonda de pH con conector BNC
- Controlador pH 2.0 (3 pines)
- Ajuste de ganancia
- Indicador LED

Imagen. Sonda pH y Controlador.

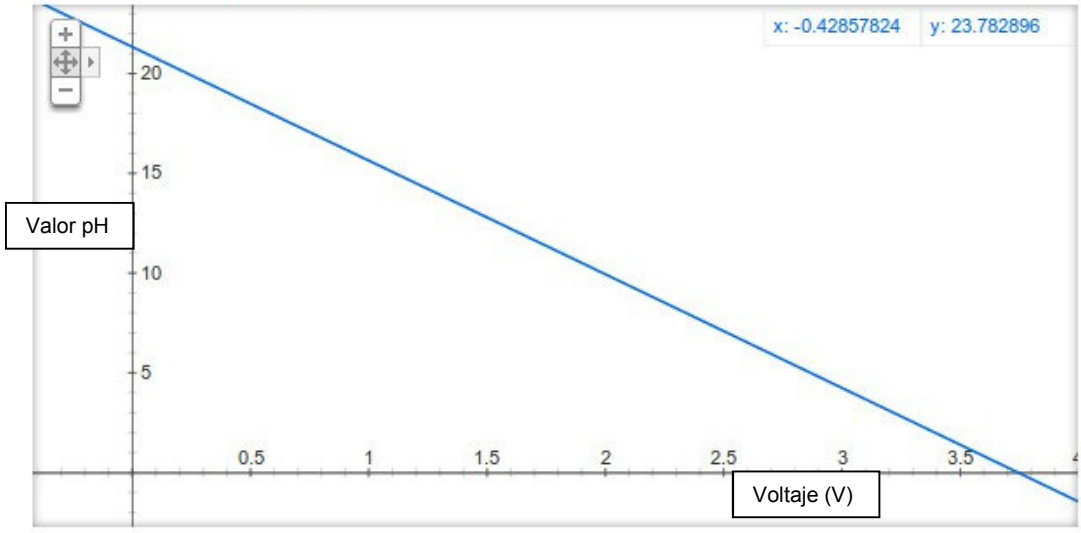


Detalle del controlador de la sonda de pH

Los pines de conexión son: GND, VCC y datos:

El controlador recibe de forma directa un valor analógico de corriente recibida por el electrodo, de forma que con éste dato podemos calcular linealmente el valor de pH de la solución / líquido. Por tanto alimentando el controlador y conexionando a cualquier puerto analógico tendremos la medida.

La siguiente gráfica muestra la correspondencia lineal de los valores medidos con el electrodo y el valor de pH:



Gráfica lineal de correspondencia y=pH / x=voltaje

Como podemos observar se debe aplicar una formula cara a obtener el valor adecuado. Así pues un valor de pH 0 corresponde a un valor medido en zona de 3.75. Para ello se pasa primero el valor medido a milivoltios para posteriormente aplicar el valor de la gráfica (cuyo valor factor es, en el caso de nuestra sonda elegida, 3.5) [valor pH=milivoltios*3.5]. La calibración de este tipo de controlador es bastante sencilla, gracias en parte a la linealidad de la relación entre pH y corriente del electrodo; para ello tan sólo se debe regular con un

pequeño destornillador el potenciómetro que tiene adjunto el controlador tomando como referencia alguna solución de control con un pH ya definido.

Imagen. Sonda pH

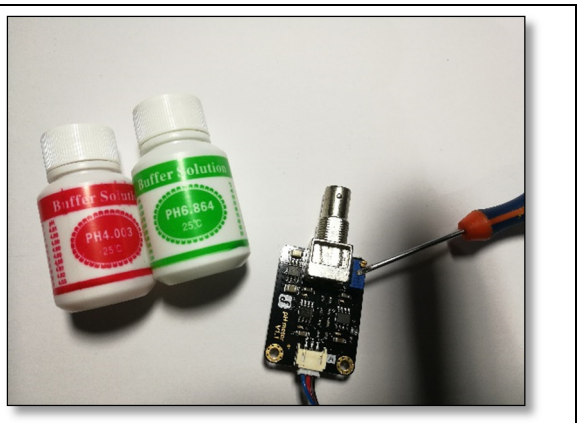


Imagen. Calibración sonda pH

Las características de medición de las principales sondas son:

Electrode type	range	temperature	Zero point	Alkali deviation	PTS	Response time	Internal resistance	Repeatability	Noise
	PH	℃	PH	mV		min	MΩ		mV
65-1	0-14	0-80	7±1	<15	>98	<2	<250	<0.017	
BX-5	0-14	0-80	7X±11	<15	>98	<2	<250	<0.017	
E-201	0-14	0-80	7±0.5	<15	>98	<2	<250	<0.017	<0.5
E-201-C	0-14	0-80	7X±0.5	<15	>98	<2	<250	<0.017	<0.5
95-1	0-14	0-80	7X±0.5	<15	>98	<2	<250	<0.017	<0.5
E-900	0-14	0-80	7X±0.5	<15	>98	<2	<250	<0.017	<0.5

En el caso de uso que se trata en este proyecto, no se medirán líquidos por encima de los 80 grados, igualmente la precisión en todos los casos es más que aceptable. Se debe tener en cuenta que para casos donde la precisión sea milimétrica deberíamos escoger sondas con poco o ningún nivel de ruido.

Conectividad red de telefonía.

Otros de los puntos fuertes del proyecto es la comunicación con el exterior, en este caso, la comunicación con servidores que se ubicarán en Internet. La flexibilidad y potencia que aporta esta característica es su principal atractivo. Por ello se debe escoger bien mirando los requisitos del proyecto:

1) Tamaño de datos a procesar: dependiendo de la cantidad de datos a procesar se deberá optar por un tipo de conexión u otra. Así pues en nuestro caso sabemos que básicamente tenemos que realizar una petición WEB a un servidor pasando como parámetros los valores medidos / tomados no necesitando un gran ancho de banda y por tanto dando la posibilidad de usar una conexión GPRS básica.
2) Flexibilidad: dependiendo la shield escogida, ésta puede tener instalado más hardware adjunto; es obvio que el hardware “extra” que traiga aparte de la propia conexión en sí, deberá cumplir igualmente con los requisitos del propio proyecto.
3) Precio: al igual que nos ocurría en los puntos anteriores, el precio en este punto sí es muy determinante a la hora de elegir un dispositivo u otro. Es muy importante seleccionar en función de las necesidades presentes del proyecto y futuras proyectadas.

Selección de tipología y shield

Sin duda alguna las siguientes tecnologías conexiones a GPRS (como son 3G o 4G) mejoran notablemente las velocidades de transmisión, no obstante, los precios de los dispositivos aumentan notablemente; igualmente como se ha mencionado anteriormente la transferencia de datos de proyecto, si bien es continua, será de poca cantidad en cada transacción por tanto este hecho hace que se decante este proyecto entre los siguientes dos dispositivos descritos en los siguientes apartados.

GPS/GPRS/GSM Shield V3.0

La GPS/GPRS/GSM Shield está basada en un módulo cuatribanda SIM908 que funciona en la frecuencias EGSM 900MHz/DCS 1800MHz y GSM850 MHz/PCS 1900MHz. También incorpora un receptor GPS para recuperar datos de posicionamiento.

La shield se controla con comandos AT e incluye dos antenas: una antena GPS y una antena GSM de alta ganancia.

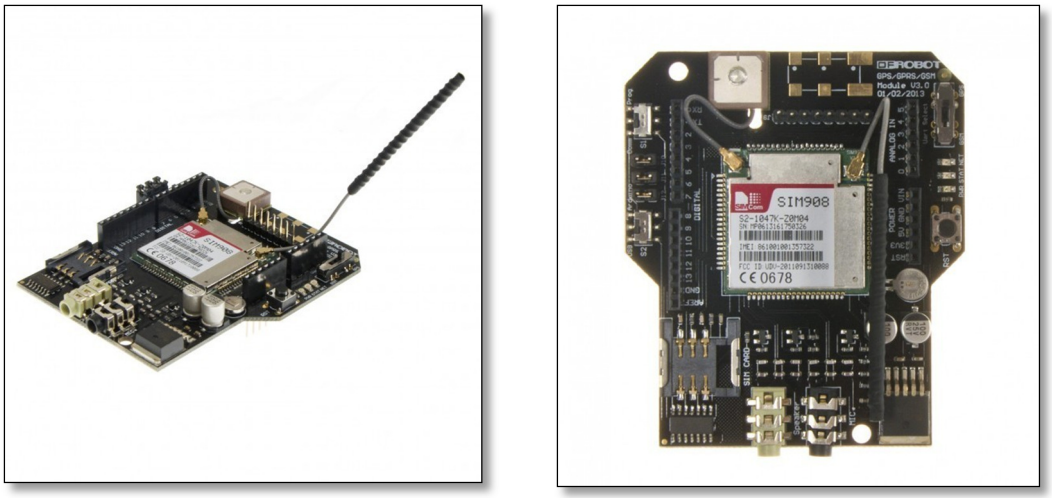


Imagen. Shield V3

Características:

- Alimentación: 6-12v
- Consumo en reposo: (100mA/7v - en modo GSM)
- Quad-Band 850/900/18001900MHz
- GPRS multi-slot class 10
- GPRS mobile station class B
- Soporta GSM phase 2/2+
- Clase 4 (2 W a 850/900 MHz)
- Clase 1 (1 W a 1800/1900MHz)
- Control con comandos AT: GSM07.07 ,07.05 y SIMCOM)
- Receptor GPS incorporado
- Antena SMD integrada para GPS y GSM
- Conexión para teclado 4x4
- Interruptor USB/Arduino
- Indicadores LED para alimentación, estado de la red y estado del módulo
- Interruptores:
  - S1 -- Modo programación/ Modo comunicación
  - S2 -- Interfaz: USB / Arduino
- Interfaces:
  - Zócalo SIM
  - Jack para salida de audio
  - Jack para entrada de audio
- Dimensiones: 81x70mm
- Especificaciones para SMS por GSM / GPRS

- Punto a punto MO y MT
- SMS cell broadcast
- Texto u modo PDU

- Especificaciones para GPS
  - Receptor de 42 canales, GPS L1 C/A code, núcleo STE
  - Sensibilidad:
    - Tracking: -160 dBm
    - Cold starts: -143 dBm
  - Tiempo de captura (primera vez):
    - Cold starts: 30s
    - Hot starts: 1s
  - Margen de error: Inferior a 2.5m CEP
  - Consumo (GSM en reposo): Adquisición 77mA, Tracking 76mA

El precio de mercado de este tipo de dispositivo es de unos 80€.

SIM900 GPRS 850/900/1800/1900 MHz / GSM

El shield basado en el chip SIM900 se comunica con Arduino mediante comandos AT enviados a través del puerto UART (puerto serie).

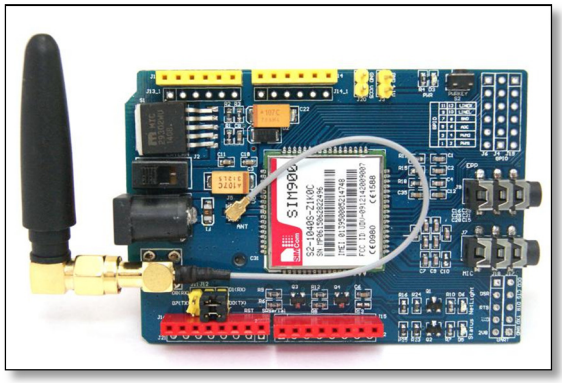
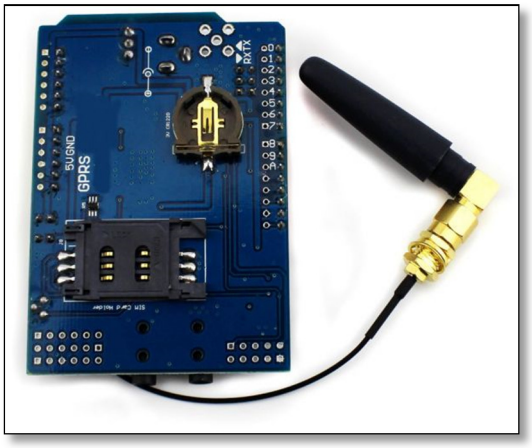


Imagen SIM900



Características:

- Cuatribanda 850 / 900 / 1800 / 1900 MHz (esto hace que funcione en todo el mundo).
- Intervalos Múltiples GPRS clase 10/8.
- Clase de estación móvil GPRS B.
- Compatible con la GSM fase 2 / 2+.
- Clase 4 (2 W a 850 / 900 MHz).
- Clase 1 (1 W a 1800 / 1900MHz).
- Control a través de comandos estándar: 07.07 GSM & 07.05 y comandos mejorados: SIMCOM comandos AT.
- Short Message Service - por lo que puede enviar pequeñas cantidades de datos en la red (ASCII o hexadecimal sin procesar).
- Pila TCP/UDP incorporado (permite transferir datos a un servidor web).
- RTC compatible.
- Puerto serie seleccionable.
- Tomas de altavoz y auriculares.
- Bajo consumo de energía - 1.5mA (modo de ahorro).

Debemos tener en cuenta que este dispositivo a diferencia del anterior no posee chip receptor GPS, por tanto debemos seleccionar un receptor aparte. El precio aproximado de este shield es de 20€.

La diferencia económica aunque no sea tenido en cuenta el receptor GPS es notable, por tanto nos decantaremos por éste último siendo éste dato el tenido en cuenta en el estudio de viabilidad económica anterior. Para seguir cumpliendo los requisitos del proyecto elegiremos un receptor GPS independiente, de este modo podremos mejorar la eficacia e incluso se podrá escalar con mayor versatilidad.

Modulo GPS

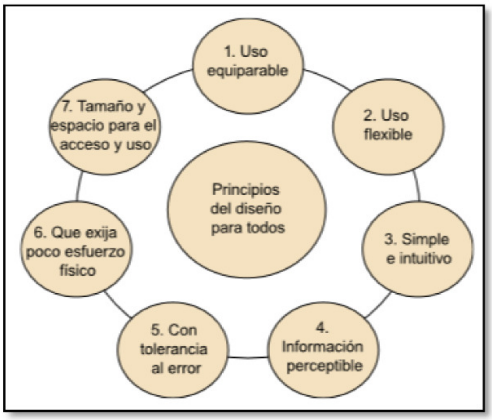
El módulo GPS en su modelo GY-GPS6MV2 viene con un módulo de serie U-Blox NEO 6M equipado en el PCB, una EEPROM con configuración de fábrica, una pila de botón para mantener los datos de configuración en la memoria EEPROM, un indicador LED y una antena cerámica. También posee los pines o conectores Vcc, Rx, Tx y Gnd por el que se puede conectar a algún microcontrolador mediante una interfaz serial. Para que nuestro módulo GPS funcione a la perfección se recomienda hacer las pruebas en un ambiente abierto o cercano a la ventana para una correcta recepción de la señal, al igual que ocurre con la mayoría de los receptores GPS de los móviles.
Cabe destacar la alimentación de éste módulo, 3.3v. El conexionado se verá en los apartados posteriores.

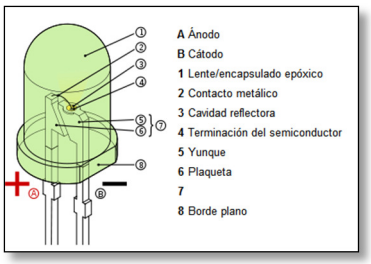
Imagen. Receptor GPS

El precio aproximado es de unos 12€ que junto al módulo GSM (20€) hacen un total de 32€, una cantidad muy inferior al aportado por el primer shield analizado (con las mismas prestaciones finales).

Leds

Cualquier sistema debe cumplir con los requisitos establecidos en los principios del diseño para todos, donde una parte fundamental es la presencia de la información perceptible.
Un diodo emisor de luz (LED por sus siglas en inglés, light-emitting diode, o led, de acuerdo con la Real Academia Española) es una fuente de luz constituida por un material semiconductor dotado de dos terminales. Se trata de un diodo de unión p-n, que emite luz cuando está activado.
En este proyecto se van a usar 3 leds diferentes, de modo que dos de ellos indicará la conectividad GPS (rojo negativa, verde positiva), estos mismos leds (rojo o verde) indicarán el status de encendido del dispositivo junto a la pantalla LCD. Un tercer led indicará la lectura del medidor de pH.





Partes de un Led.

Un led comienza a emitir cuando se le aplica una tensión de 2-3 voltios. En placa se debe usar de manera conjunta a una resitencia asociada cuyo valor vendrá determinado según el led usado y aplicando la Ley de Ohm. En el presente proyecto se van a usar leds de entre 1.8v (led rojo) y 2.1v (verde y amarillo) por lo que las resistencias serán entre 145 Ω y 200 Ω.

Pantalla LCD

En el presente proyecto se va a usar una pantalla LCD de 20 caracteres por cada una de las cuatro líneas que dispone el dispositivo elegido. Existe en el mercado una gran variedad de pantallas, pero al igual que pasa en los periféricos anteriores, los requisitos de precio y flexibilidad hacen que finalmente elijamos la LCD 20x4 PANTALLA DISPLAY 2004. A tener en cuenta para la siguiente parte del proyecto es seleccionarla o prepararla para la conectividad I2C, ya que de otro modo usaríamos un total de 6 pines digitales para controlar la pantalla en vez de 2 que usaremos usando los pines SDA y SCL de nuestro Arduino.

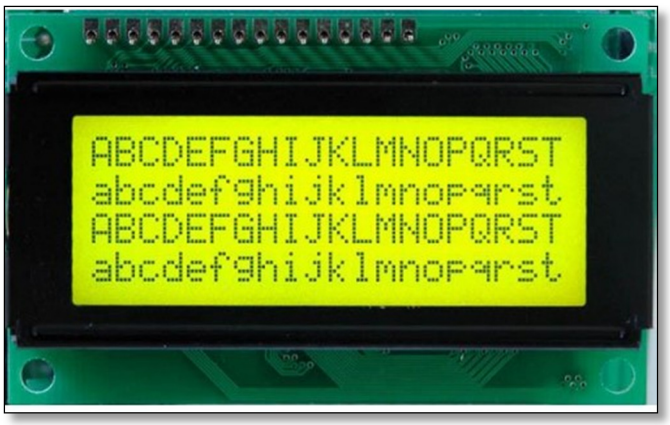
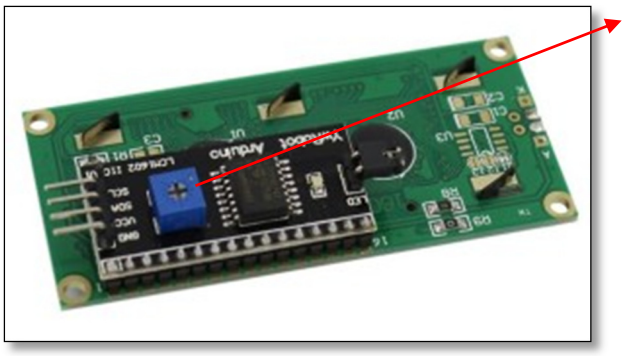


Imagen. Pantalla LCD 20x4

En la parte trasera de nuestra pantalla dispondremos del módulo controlador I2C, donde además se debe ajustar el brillo con un destornillador pequeño.



En la parte trasera de la pantalla podemos ver en azul un potenciómetro para regular la intensidad del brillo de la pantalla que podrá ser ajustada con un pequeño destornillador.

Imagen. Sonda pH y Controlador.

La conectividad I2C nos permite realizar el enlace con la placa Arduino usando tan sólo los pines SDA y SCL de la placa en vez de hacerlo con 6 pines digitales. El bus I2C, un estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de "inteligencia", sólo requiere de dos líneas de señal y un común o masa. Fue diseñado a este efecto por Philips y permite el intercambio de información entre muchos dispositivos a una velocidad aceptable, de unos 100 Kbits por segundo, aunque hay casos especiales en los que el reloj llega hasta los 3,4 MHz.

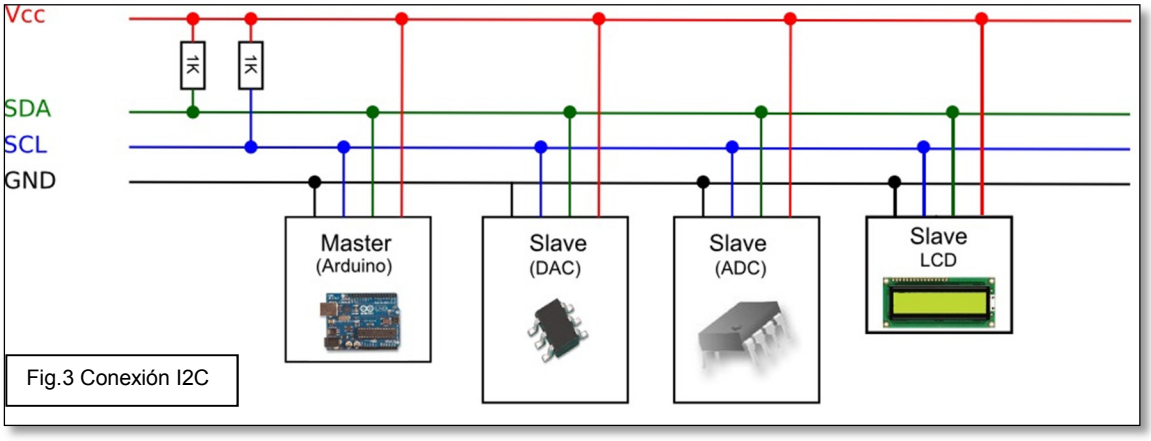


Fig.3 Conexión I2C

Observando la figura 3 podemos ver claramente dos roles en la conexión I2C: master o esclavo. En este proyecto vamos a conectar un solo maestro (podrían conectarse más) que será nuestro Arduino, éste será quien determine el acceso al canal, por otro lado el display elegido tendrá el rol de esclavo y por tanto su acceso es condicionado por el master. Es importante saber los siguientes detalles a tener en cuenta durante el desarrollo del proyecto:

1) Cada dispositivo tiene una dirección "única" que debe ser referenciada para dar acceso al canal, de modo que debemos saber la dirección de nuestro display.
2) El acceso al canal no puede ser simultáneo y es necesario igualmente no saturar el canal cara a no generar problemas de comunicación; este hecho se traduce en tiempos de espera (del orden de milisegundos).

Estos dos datos serán detallados en posteriores apartados.

Botones de conexión.

Éstos nos permiten realizar una acción al pulsarlos; en el proyecto serán usados para la calibración del equipo y para obtener una ubicación GPS "nueva" y así refrescarla, mejorarla o bien actualizar.

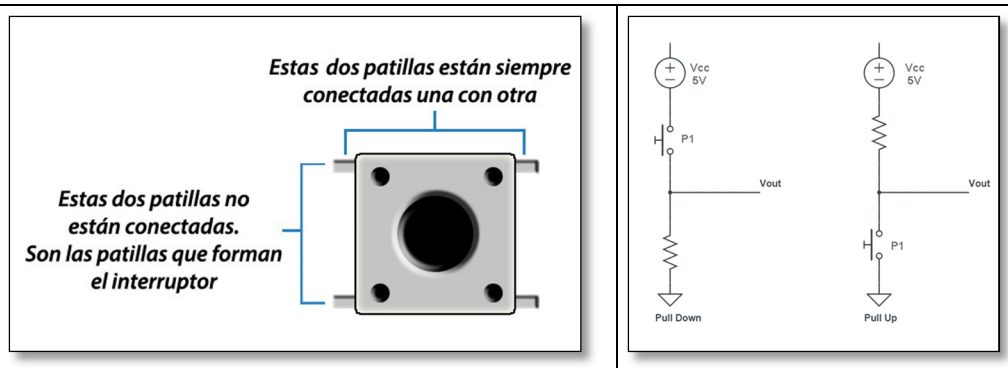


Imagen detalle de un pulsador.

La conexión es simple y cada pulsador usa un solo pin digital con una resistencia pull-down (aparte de estar conectados a toma de corriente 5V).
La principal diferencia entre poner la resistencia en pull-down o pull-up es la lectura de la medida y la eficiencia del uso de energía. Si las ponemos en pull-up nuestra resistencia siempre estará "activa" y por tanto disipando energía, en este caso se detectaría un valor alto (HIGH) en el pin, en cambio al pulsar el botón se leería un valor bajo (LOW). En modo pull-down la lectura es justo al contrario. En la elaboración del proyecto se ha preferido este último caso por razones de eficiencia energética (la resistencia no disipa nada) aunque en términos generales es casi inapreciable.

Interconexión de componentes

Después de haber realizado la selección de componentes y ver el número de pines necesarios, así como ver compatibilidades de cada elemento con las necesidades del proyecto se procede a realiza el ensamblado del mismo. Antes de la asignación de pines cabe destacar la peculiaridad de la shield GPRS. En ella existen varias opciones, así pues si cambiamos los jumpers de la placa de la imagen podremos seleccionar con qué pines de conexión se comunicará el Arduino con ella. En nuestro caso he preferido usar los pines 7 (para TX) y 8 (para RX) (Figura 4). Igualmente para que la placa pueda ser apagada/encendida desde la placa Arduino es necesario puentear la resistencia marcada como R13 (Figura 5).

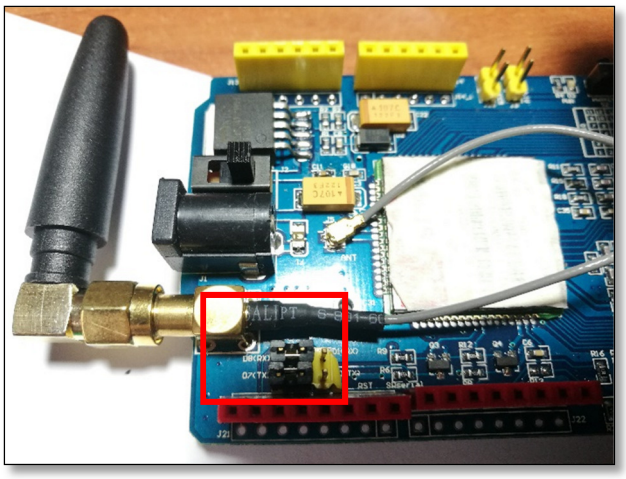


Fig. 4. Jumpers de configuración pines TX-RX

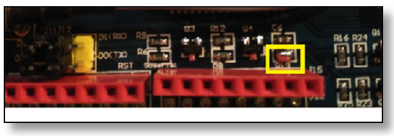


Fig. 5. Detalle de la soldadura en R13

Igualmente se ha decidido compartir la alimentación del Arduino con la placa y por tanto es necesario cambiar el selector del shield para ello (Figura 6).

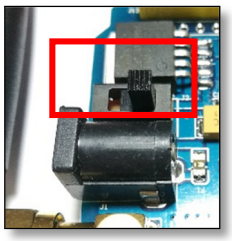


Fig 6. Detalle del selector de corriente.

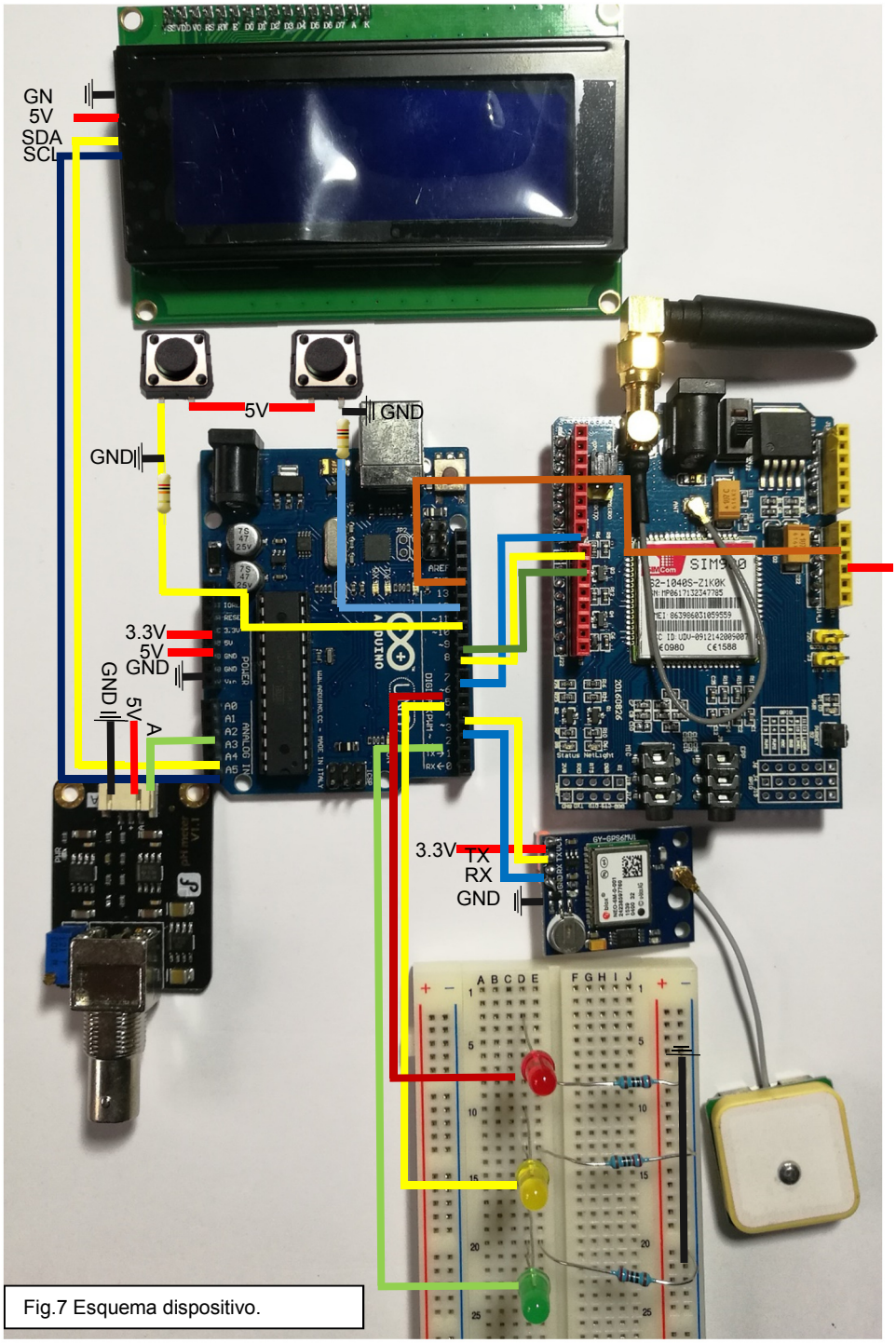


Fig.7 Esquema dispositivo.

En el esquema mostrado en la figura 7 anterior no se ha querido montar el shield empotrado en el Arduino por diversas razones:

- La tarjeta SIM quedará oculta entre ambos dispositivos.
- La entrada de alimentación se encuentran en zonas opuestas. Tal y como se verá en la preparación final de la PCB un esquema lineal nos permitirá alimentar eléctricamente a ambos y quedará un montaje mucho más limpio.

Esquema de conexiones:

Pin Placa Arduino	Elemento	Pin Elemento
A0	-	-
A1	-	-
A2	-	-
A3	Controlador pH	A
A4	LCD	SDA
A5	LCD	SCL
D0	-	-
D1	-	-
D2	Led Verde	Pata positiva
D3	GPS	Rx
D4	GPS	Tx
D5	Led Amarillo	Pata positiva
D6	Led Rojo	Pata positiva
D7	GPRS SIM900	Pin 7
D8	GPRS SIM900	Pin 8
D9	GPRS SIM900	Pin 9
D10	Botón Calibración	Pata
D11	Botón Reubicación	Pata
D12	-	-
D13	-	-

Pin Placa Arduino	Elemento	Pin Elemento
VCC 5V	Controlador pH LCD SIM900 B. Calibración B. Reubicación	VCC VCC 5V Pata Libre Pata Libre
VCC 3.3	GPS	VCC
GND	Controlador pH LCD GPS GPRS SIM900	GND GND GND GND

Alimentación eléctrica del dispositivo

Llegados a este punto donde ya se han interconectado todos los sensores y actuadores del sistema, se procede al detalle de la alimentación eléctrica del dispositivo.
La mayoría de ellos cogen directamente la tensión y tierra de la placa Arduino: pantalla LCD, GPS, controlador sonda pH, leds… sin embargo la placa GPRS es objeto de estudio.
Una característica de la shield GPRS escogida en la posibilidad de seleccionar el origen de la fuente de alimentación, bien desde el propio Arduino o bien a través del Jack de alimentación y compartir el pin de tierra con la placa Arduino. Si contamos con una alimentación fiable en la placa Arduino optaremos por alimentar de forma simultánea ambas placas, sin embargo si prevemos dificultades en la alimentación es recomendable alimentar la placa SIM900 por separado cara a evitar incidencias del sistema.

Posibilidades de alimentación

Existen diversas maneras de alimentar nuestra placa Arduino y la shield GPRS (pilas de 9V, 4 pilas AA de 1.5,5 baterías recargables de 1.2V…), no obstante vamos a centrarnos en aquellas

que nos ofrecen una corriente estable (tanto en intensidad como voltaje) ya que nuestra placa SIM900 dependiendo del estado necesita una intensidad de corriente continua.

2 BATERÍAS DE LITIO 18650 DE 3,7V

Dos baterías de litio 18650 puestas en serie proporcionan 7.4-8.2V, que es un voltaje perfecto para alimentar Arduino. Las baterías de litio 18650 tienen la ventaja de proporcionar una alta capacidad de carga. Las baterías de primeras marcas proporcionan hasta 4800 mAh. Proporcionan una capacidad de descarga de entre 1C – 2C, según modelos. Esto supone una intensidad máxima de hasta 10A, aunque por seguridad no resulta aconsejable drenar más de 2-4 A sin estar muy seguros de la calidad y características de vuestra batería.

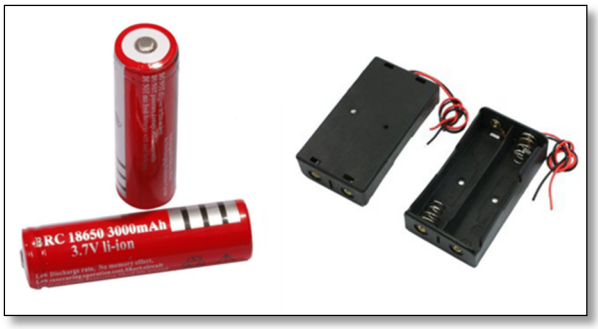


Imagen. Baterías de Ion-Litio 18650.

En nuestro caso esta propuesta se presenta bastante aceptable en términos de voltaje como intensidad, no obstante la autonomía si puede ser un factor importante a tener en cuenta. 4800mAh (máximo) ofrecerían una autonomía aproximada de 8 horas en continuo trabajo.

BANCOS DE BATERÍAS USB DE 5V

Emplear una batería USB, de las que se usan para alargar la batería de los móviles, es una opción interesante para incorporar en nuestros proyectos.

Imagen. Batería USB 5V.

Tienen como ventaja que proporcionan 5V regulados, por lo que podemos alimentar Arduino a través del USB, sin preocuparnos de la necesidad de regular el voltaje.

Muchos de estos bancos, de hecho, incorporan una única batería de litio 18650, más un pequeño circuito que eleva y regula el voltaje a 5V. En estos casos podemos incluso sustituir una batería por otra, empleando la misma caja, mientras cargamos la batería descargada.

El voltaje de 5V es adecuado para alimentar una gran variedad de componentes, como motores DC, servos, así como una gran cantidad de dispositivos (sensores, tiras LED, displays…).

Estos bancos son, por supuesto, recargables. La capacidad de energía es alta, pudiendo encontrar bancos de más de 17.000 mAh. Existe la posibilidad de tener salidas reguladas diferentes en distintas salidas del mismo, por lo que nos podríamos despreocupar de las necesidades individuales de cada dispositivo conectado. Esta opción nos ofrece una gran autonomía e incluso la posibilidad de recarga de forma continua estando en funcionamiento pero tienen por hándicap un mayor espacio de uso.

Cualquiera de las dos últimas opciones presentadas podría ser usada. Por tanto debemos elegir que factor será determinante en nuestro proyecto; por un lado una gran autonomía y por otro espacio. En este caso se ha preferido optar por una mayor autonomía y mayor espacio (hecho que en un prototipo facilitará el entramado de conexiones).

El banco de batería elegido es un “AUKEY Batería Externa 20000mAh Cargador Portátil Fino con 4 Puertos Salida y 3 Entrada”

Especificaciones
Modelo: PB-Y14
Capacidad de Batería: 20000 mAh / 74 Wh
Tipo de Batería: Polímero de Litio
Tiempo de Carga: 10 horas
Entrada Micro-USB: 5 V 2 A
Entrada Lightning: 5 V 1,5 A
Entrada/Salida USB-C: 5 V 3 A

Salida Total: 5 V 3 A Potencia Total de Salida: 15 W Dimensiones: 200 x 96 x 14 mm Peso: 435 g

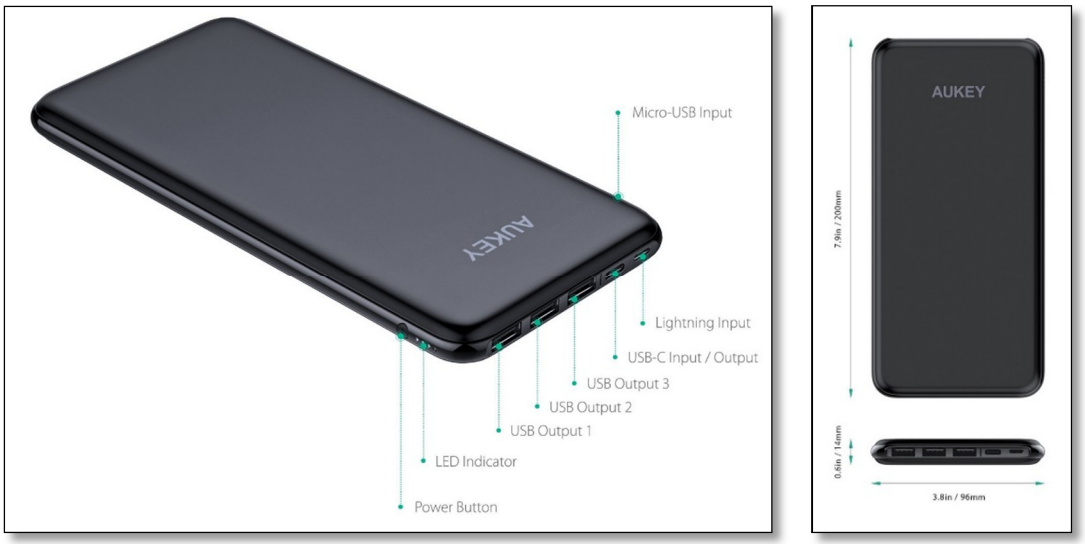


Imagen. Detalle de batería USB 5V.

Tanto las dimensiones, número de salidas como su buena capacidad lo hacen idóneo para alimentar el Arduino y la shield GPRS. El modo de alimentación se hará por USB directamente al bus de 5V y GND del sistema, de modo que se incluye en el esquema un condensador entre ambas pistas cara a dotar mayor estabilidad. En pruebas finales se verá la autonomía final real del dispositivo al completo.

Programación de la Placa

Una vez se interconecte todo, se procede a programar el Arduino; para ello se hace un montaje temporal de todos los elementos y se conecta a través del puerto USB-B del Arduino al ordenador.

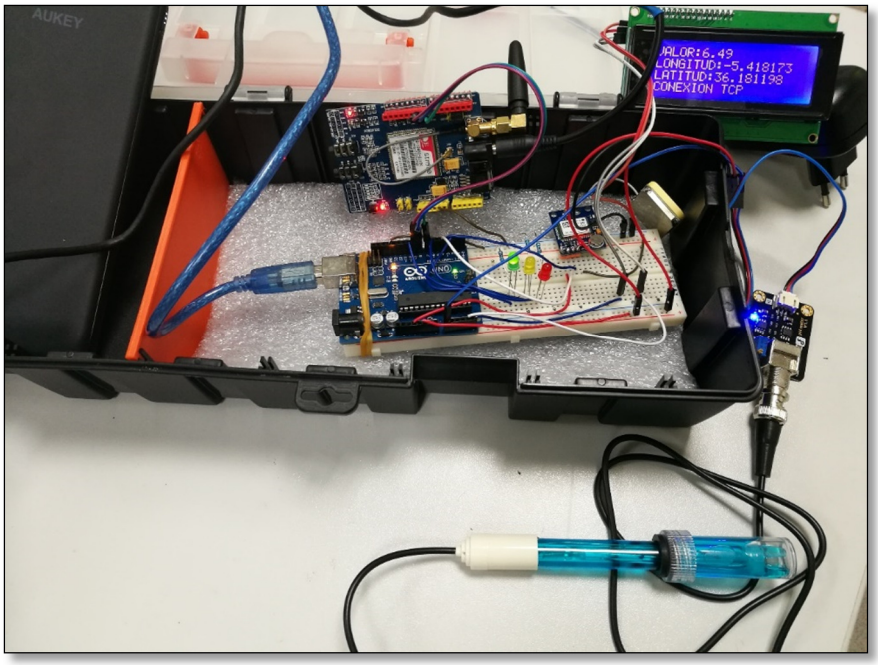
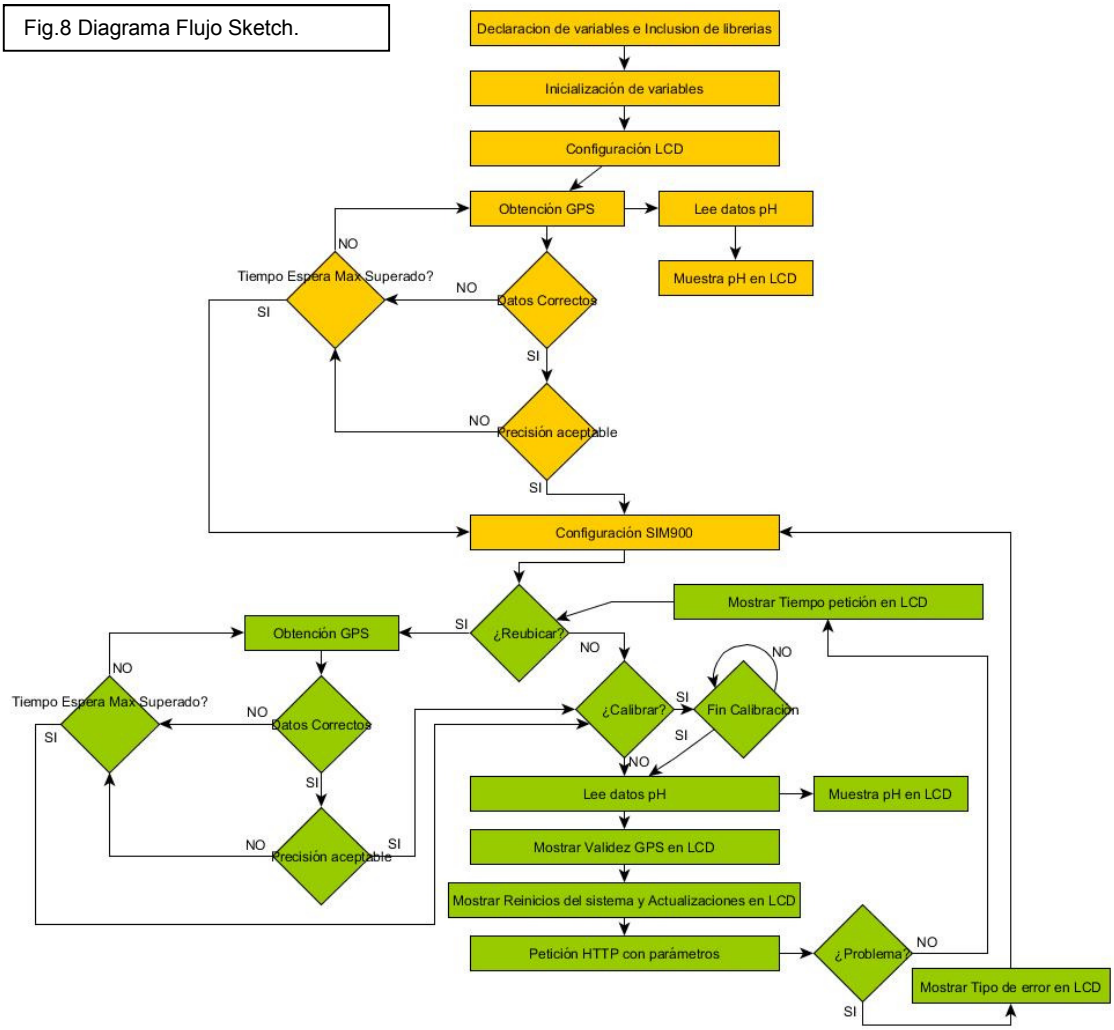


Imagen del dispositivo sin placa PCB

Para facilitar la comprensión general de esta parte del proyecto se crea el siguiente diagrama de flujo TOP-DOWN de la secuencia del sketch; en él se puede observar cómo transcurre todo desde que se declaran las variables hasta que se repite o entra en la secuencia iterativa (Loop):

Fig.8 Diagrama Flujo Sketch.



En el diagrama de la figura 8 mostrada se puede observar la secuencia donde en primer lugar se declaran e inician las variables y librerías; posteriormente se hace la petición GPS donde como se explicará más en detalle se dota de varios mecanismos de control, posteriormente se configura el modulo SIM para pasar al bloque iterativo (en verde). En este bloque iterativo se observa el estado del botón para una posible reubicación (si está pulsado), para seguir con el botón de calibración; terminada ambas tareas de comprobación / configuración pasa a leer el dato de pH del sensor y a mostrar en pantalla los distintos valores resumen (uno de reinicios del sistema y otro de actualizaciones realizadas en la sesión).
Terminados estos procesos se vuelve a repetir todo desde la monitorización del botón para reubicar.
El objetivo principal en la configuración software es conseguir actualizar un valor cada minuto (contando con posibles incidencias de cobertura, interferencias…); en código se fuerzan diversos retrasos necesarios para las esperas de respuesta del módulo SIM (éste hecho será estudiado en la sección de pruebas finales).

A continuación se va a proceder a explicar el código de las partes más importantes del sketch.

Preparación de pantalla LCD para lectura de datos.

Si bien la parte principal del proyecto es la lectura de datos remota en un servidor, se hace imposible la usabilidad sin mostrar en el propio dispositivo los datos o acciones que se están

llevando a cabo. Una vez conectada tal y como se ha mostrado en el esquema de conexiones (Figura 7), se deben incluir las librerías Wire y LiquidCrystal_I2C e inicializar la variable de control. Al usar I2C en vez de pines digitales, necesitamos averiguar la dirección I2C del controlador, en nuestro caso se especifica en las propiedades del producto como 0x3F. No obstante para saber la dirección podríamos conectar únicamente el LCD y hacer uso del siguiente Sketch adjunto donde nos mostraría por el puerto serie la dirección exacta de nuestro dispositivo:

```
#include <Wire.h>
void setup()
{
  Wire.begin();
  Serial.begin(9600);
  Serial.println("\nI2C Scanner");
}

void loop()
{
  byte error, address;
  int nDevices;
  Serial.println("Scanning...");
  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
// El programa usa el error devuelto en el término de la conexión para averiguar si hay algún dispositivo.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknow error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);          // Espera 5 sg y comienza un nuevo escaneo.
}
```

Abriendo la consola serial del propio Arduino se observa que encuentra un dispositivo I2C en la dirección anteriormente citada:

```
Scanning...
I2C device found at address 0x3F  !
done
```

Una vez sabida la dirección I2C podemos inicializar la pantalla:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR 0x3F   // dirección I2C para controlar el LCD
LiquidCrystal_I2C lcd(I2C_ADDR,2, 1, 0, 4, 5, 6, 7); //variable de control LCD y conexiones internas de LCD con el controlador I2C.
int tiempoI2C=20;          //Tiempo de espera necesario para no colapsar el puerto I2C en milisegundos.
void setup(){
            …
      lcd.begin (20,4);    // Inicializar el display con 20 caraceres 4 lineas
      lcd.setBacklightPin(3,POSITIVE);
      lcd.setBacklight(HIGH);

      lcd.home ();                  // inicio
```

```
      lcd.print("ARDUNOX PH");
      lcd.setCursor ( 0, 1 );       // 2ª Linea …
            …
}
```

A partir de este punto se han creado varias funciones de control para la pantalla:

```
void escribirLCD(String msg,int linea){ // se pasa como parámetro el mensage y la linea donde quiero imprimirla
  lcd.setCursor(0,linea);
  delay(tiempoI2C);
  lcd.print("                ");
  delay(tiempoI2C);
  lcd.setCursor(0,linea);
  delay(tiempoI2C);
  lcd.print(msg);
  delay(tiempoI2C);
}
```

En el código de la función cabe destacar los pequeños retrasos que he forzado entre escritura y otra; esto es debido al uso del BUS I2C, llegados a puntos de uso intensivo con tiempo prolongado podría saturarlo y crear “cuelgues” inesperados por lo que se hace necesario la inclusión de un tiempo de espera (en este caso a través de la variable tiempoI2C inicializada a 20 provocando un retraso de 20 milisegundos).

Leds y resistencias

Al igual que la pantalla un elemento que nos indica el estado de forma clara, sencilla y rápida son los Leds, para que éstos no se deterioren debemos instalar una resistencia para establecer la corriente de trabajo, de lo contrario éstos se podrían quemar o no funcionar. Los leds que usamos tienen un voltaje de trabajo de 2.1V y 20mA , el voltaje de la fuente es de 5V por lo que debemos aplicar una resistencia de al menos 145Ω por Led (La Ley de Ohm relaciona el valor de resistencia con el voltaje y la corriente que la atraviesa, de la forma V = I·R, despejando, R = V/I. Tenemos que V es 2,9V (corriente a disipar) y que I es 20mA, por lo que R = 2,9V / 0,02A = 145 Ohm.)

En el sketch debemos configurar los pines digitales de los leds como pines de salida para posteriormente mandar un valor HIGH o LOW al correspondiente que queramos encender o apagar. Cara a mejorar la escalabilidad del sketch se crean variables que indican los números de cada pin configurado.

```
  //variables para controlar los LEDS
const int ledPinVerdeGPS = 2;
const int ledPinLecturaPH = 5;
const int ledPinRojoGPS = 6;
…
void setup(){
…
  pinMode(ledPinRojoGPS, OUTPUT);
  pinMode(ledPinVerdeGPS, OUTPUT);
  pinMode(ledPinLecturaPH,OUTPUT);

    digitalWrite(ledPinRojoGPS, HIGH);
    digitalWrite(ledPinVerdeGPS, HIGH);
    digitalWrite(ledPinLecturaPH, LOW);
…
}
```

Lectura de datos GPS

Una vez detallado el sistema de lectura directa de estado se va a programar la primera parte del dispositivo donde se obtendrá la dirección GPS. Para ello se va a hacer uso de la librería TinyGPS la cual incluye procedimientos para obtener cadenas válidas de los datos recibidos del receptor GPS.

Básicamente el procedimiento de control que hace es el siguiente:

1) Observa los datos recibidos a través de comunicación serial con el receptor.
2) Cuando obtiene una cadena válida lo registra, informa y nosotros lo almacenaremos en una variable de un tipo definida igualmente en la propia librería (tipo gps).
3) Si supera el tiempo de espera y no tiene una posición válida continua el proceso pero sin información GPS.

4) Si encuentra el número predefinido de posiciones válidas, en nuestro caso definido en variable *numPosCorrectas*, termina el proceso con datos de GPS tomadas como válidas.

Los datos que se reciben tienen la siguiente estructura:

```
$GPRMC,135825.00,A,0804.79095,S,07902.70984,W,0.231,,220615,,,A*75
$GPVTG,,T,,M,0.231,N,0.427,K,A*22
$GPGGA,135825.00,0804.79095,S,07902.70984,W,1,033.8-07M,.,M,4D
GPS,,23,1,6,,,,,32,30,10*0$GGV,,,0,3,,4,0,211,4,883,2,301,3*D
$PSV22,7326,24414,6,9,3,1,628,8*4
$PLL004703,,092.010W158500AA*E
```

Siguen el protocolo NMEA (siglas de National Marine Electronics Asociation), las cuales son sentencias estándares para la recepción de datos GPS. Una de ellas y la más usada son las sentencias $GPRMC, las cuales tienen la siguiente estructura:

$GPRMC,044235.000,A,4322.0289,N,00824.5210,W,0.39,65.46,020615,,,A*44

Analizando la trama se observaría:

- 044235.000 representa la hora GMT (04:42:35)
- “A” es la indicación de que el dato de posición está fijado y es correcto. “V” sería no válido
- 4322.0289 representa la longitud (43º 22.0289´)
- N representa el Norte
- 00824.5210 representa la latitud (8º 24.5210´)
- W representa el Oeste
- 0.39 representa la velocidad en nudos
- 65.46 representa la orientación en grados
- 020615 representa la fecha (2 de Junio del 2015)

Para la integración en nuestro sketch de Arduino se deben incluir las librerías TinyGPS y SoftwareSerial.

Se inicializa una variable para el control serial, la mencionad anteriormente para posiciones correctas, otra para almacenar los datos recibidos y dos que se usarán para tener de forma accesible los datos de longitud y latitud; igualmente se crea una variable de control para los casos en que no haya cobertura GPS que salte la búsqueda transcurrido un tiempo determinado.

```
SoftwareSerial ss(4,3);
TinyGPS gps;
float latitud,longitud;
unsigned long tiempoEspera=600000;
```

La rutina en el sketch se inicia en el setup y por tanto se realiza una sola vez (debemos de tener en cuenta que el dispositivo una vez que empiece su uso no cambiará su ubicación hasta que lo apaguemos y lo traslademos a otro lugar); la parte específica de configuración GPS es:

```
void setup(){
…
ss.begin(9600);       //Arrancamos el GPS con velocidad de comunicacion de 9600
                       //La siguiente función determina la posición GPS inicial;
                      //al tratarse de un equipo de movilidad reducida una vez
                       //determinada la posición no la refrescamos a no ser que el cliente quiera.
   //Variable de control GPS inicializada a "no valido"
   valido=0;  //0=posicion NO valida  - 1=posicio valida

 getValidGPSPosition(tiempoEspera);  //Obtener posición con tiempo límite marcado.
ss.end();
…
}
void getValidGPSPosition(unsigned long tiempo){
 unsigned long  tiempoInicio;
    bool newData = false;
  unsigned long chars;
  unsigned short sentences, failed;
  int posiciones;
  posiciones=0;
  ss.begin(9600);  //Arrancamos el GPS con velocidad de comunicacion de 9600
  //La siguiente función determina la posición GPS inicial;
  //al tratarse de un equipo de movilidad reducida una vez
  //determinada la posición no la refrescamos a no ser que el usuario quiera.

tiempoInicio=millis();
while((valido==0 || posiciones<numPosCorrectas) && (millis()-tiempoInicio)<tiempo) //condicion para terminar
{
  // Analizamos datos GPS y palabras clave por 1 sg.
```

```
  for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (ss.available())
    {
      char c = ss.read();
    if (gps.encode(c)) // Nuevo dato encontrado con validez
        newData = true;
    }
  }

  if (newData)
  {
    posiciones=posiciones+1;
    //Mostramos en pantalla el estado y la el numero de posiciones correctas encontradas
     escribirLCD("POS. ENCONTRADA ",3);
     lcd.print(posiciones);
     delay(tiempoI2C);
     lcd.print("/");
     delay(tiempoI2C);
     lcd.print(numPosCorrectas);
     delay(tiempoI2C);

    valido=1;
    float flat, flon;
    unsigned long age;
    gps.f_get_position(&flat, &flon, &age);
    //Guardamos los datos de latitud y longitud
    latitud=flat;
    longitud=flon;

  }
  gps.stats(&chars, &sentences, &failed);

  //No pausamos la lectura de pH mientras busca posición
  valorPHLCD();

}
//Ponemos el estado de validez en los leds de información (Rojo=SIN GPS // verde=GPS OK )
  if(valido==0)
    {
      digitalWrite(ledPinRojoGPS, HIGH);digitalWrite(ledPinVerdeGPS, LOW);
      escribirLCD("SIN GPS",3);
    }
     else
    {
      digitalWrite(ledPinRojoGPS, LOW);digitalWrite(ledPinVerdeGPS, HIGH);
      escribirLCD("GPS OK",3);
    }

 ss.end();
}
```

Al término tenemos almacenado en las variables *latitud* y *longitud* los datos de posición; igualmente tendremos la variable *valido*=1 si es correcta o 0 si no se ha podido identificar la posición.

Lectura de pH

Tal y como se adelantó al presentar el controlador de pH, el valor se obtiene a través de una entrada analógica cuyo valor viene directamente de un electrodo acoplado. Tan sólo hay que aplicar una conversión a milivoltios la cantidad recibida y ésta tiene una relación directa con el valor de pH. Para mejorar el dato recibido se toman N muestras, en este caso según la variable *precisionPH*, para hacer posteriormente las conversiones.

```
float getpH(){
  for(int i=0;i<precisionPH;i++)      //toma de muestras (segun la constante definida precisionPH)
  {
    buf[i]=analogRead(SensorPin);
    delay(10);
  }
  for(int i=0;i<precisionPH-1;i++)      //Ordenar muestras
  {
    for(int j=i+1;j<precisionPH;j++)
    {
      if(buf[i]>buf[j])
```

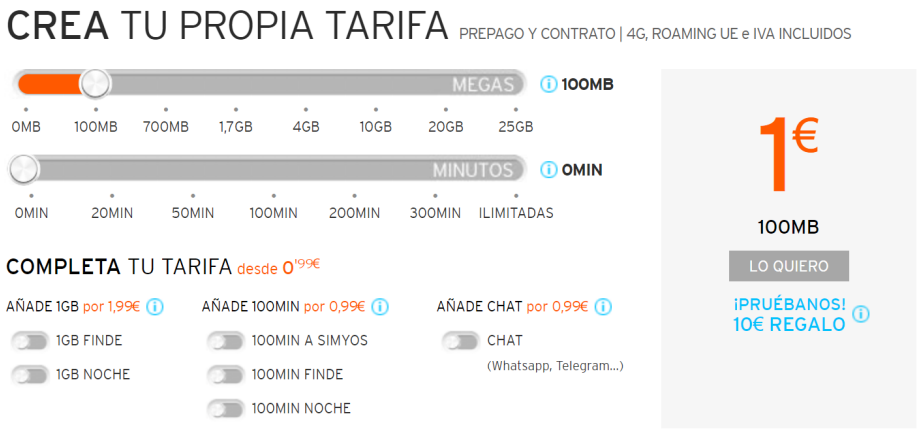
```
    {
      temp=buf[i];
      buf[i]=buf[j];
      buf[j]=temp;
    }
   }
  }
  avgValue=0;
  for(int i=2;i<precisionPH-2;i++)                    //Se hace la media de las muestras intermedias
    avgValue+=buf[i];
  float phValue=(float)avgValue*5.0/1024/(precisionPH-4); //se convierten las mediciones a milivoltios
  phValue=3.5*phValue;                                //se convierten los milivoltios en valor de pH

  digitalWrite(ledPinLecturaPH, HIGH);       //Hago parpadear el led de lectura
  delay(800);
  digitalWrite(ledPinLecturaPH, LOW);

  borrarPantalla(0);                          //Escribo en LCD el valor calculado.
  lcd.print("VALOR:");
  lcd.print(phValue);

  return phValue;
}
```

Configuración y conexión GPRS SIM900

Una vez obtenidas las lecturas de posición y pH debemos crear la conexión con el servidor. Para ello el chip SIM900 usa comandos AT que no son más que instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal modem. De este modo tenemos primero que conectar primero con la red GSM, para posteriormente configurar un nombre de punto de acceso (APN - dado por el proveedor de servicio móvil) y finalmente realizar peticiones “vivas” a un servidor en Internet.
Previa a la explicación del sketch se detalla un servicio GSM válido.

Contratación de servicios GSM y datos de conexión a Internet.

El primer paso requiere primero de la contratación de un servicio móvil. Dependiendo de los servicios que vamos a usar se debe escoger unos servicios u otros, en este caso no necesitamos realizar llamadas ni enviar mensajes, tan sólo una conexión mínima de datos (la cadena que enviaremos con los datos no sobrepasará en ningún caso los 256 bytes).
En el mercado existen varias opciones, una válida es la siguiente contratada en Simyo:



Tarifa para conexión GPRS elegida.

Por 1€ al mes tendremos contratado el servicio de datos suficiente para el proyecto. Una vez tengamos la tarjeta, ya dispondremos del PIN de inicio en red.
Los datos APN nos lo debe facilitar la empresa de servicio móvil, en el caso de SIMYO son:

Crear un punto de acceso nuevo (APN)

- *Simyo Internet*:
  - Nombre: *Simyo Internet*
  - APN: **gprs-service.com**
  - Nombre de usuario: vacío
  - Contraseña: 1234
  - MCC: 214
  - MNC: 19
  - Tipo de APN: default
  - El resto de campos se deja vacío
  - **Puede que algunos modelos se necesite Activar** dos opciones más en el teléfono: **"Itinerancia de datos"** y **"Conexión móvil de datos"**

Con estos datos ya se tiene información suficiente para configurar en el Sketch los datos de conexión donde principalmente usaremos los datos de APN, usuario y contraseña.

Configuración Sketch GPRS SIM900

Para poder pasar los comandos AT se crea una variable de control del tipo SoftwareSerial pasándole como parámetros los puertos especificados en la propia placa a través de los Jumpers de configuración (como se había visto en este caso los pines 7 y 8). Debemos tener en cuenta que el Pin 9 del GPRS está dedicado en exclusiva para el encendido / apagado del mismo, de modo que para encender/apagar el mismo se manda un pulso positivo de al menos 2 sg por el citado pin; en esa línea se crea una función para encenderlo y apagarlo en código.

```
void power_on()
{
  int respuesta = 0;
  // Comprueba que el modulo SIM900 esta arrancado
  if (enviarAT("AT", "OK", 2000) == 0)
  {
    Serial.println("Encendiendo el GPRS...");
    pinMode(9, OUTPUT);
    digitalWrite(9, HIGH);
    delay(1000);
    digitalWrite(9, LOW);
    delay(1000);
    // Espera la respuesta del modulo SIM900
    while (respuesta == 0) {
      // Envia un comando AT cada 2 segundos y espera la respuesta
      respuesta = enviarAT("AT", "OK", 2000);
      SIM900.println(respuesta);
    }
  }
}

void power_off()
{
  digitalWrite(9, HIGH);
  delay(1000);
  digitalWrite(9, LOW);
  delay(1000);
}
```

El encendido se apoya en una función "*int enviarAT(String ATcommand, char* resp_correcta, unsigned int tiempo)*" creada para controlar las respuestas que se reciben tras enviar un comando. Dicha función recibe como parámetros el propio comando, la respuesta esperada y un tiempo máximo de espera de respuesta. En el caso del encendido es notorio su uso, ya que se debe tener presente que si el GPRS se encuentra encendido y se mandan los pulsos usando el pin 9, éste se apagaría y sin un control de respuesta el sketch (el dispositivo) se quedaría en espera.

```
int enviarAT(String ATcommand, char* resp_correcta, unsigned int tiempo)
{

  int x = 0;
  bool correcto = 0;
  char respuesta[100];
  unsigned long anterior;

  memset(respuesta, '\0', 100); // Inicializa el string
```

```
  delay(100);
  while ( SIM900.available() > 0) SIM900.read(); // Limpia el buffer de entrada
  SIM900.println(ATcommand); // Envia el comando AT
  x = 0;
  anterior = millis();
  // Espera una respuesta
  do {
    // si hay datos el buffer de entrada del UART lee y comprueba la respuesta
    if (SIM900.available() != 0)
    {
        respuesta[x] = SIM900.read();
        x++;
      // Comprueba si la respuesta es correcta
      if (strstr(respuesta, resp_correcta) != NULL)
      {
        correcto = 1;
        //Serial.println("RESPUESTA CORRECTA");
      }
    }
  }
  // Espera hasta tener una respuesta
  while ((correcto == 0) && ((millis() - anterior) < tiempo));
  Serial.println(respuesta);

  return correcto;
}
```

Para iniciar en la Red GSM se ha creado una función específica llamada en el propio setup del sketch de modo que lanza los comandos de conexión al encenderse el dispositivo.
El comando para iniciar en red está indicado como "AT+CPIN=XXXX" de modo que si el PIN pasado como parámetro es correcto se debe conectar sin mayor problema.

```
void iniciarRed(){
   int correcto=0;
  escribirLCD("INICIANDO RED GSM",3);

  correcto=enviarAT("AT+CPIN=\"XXXX\"", "OK", 1000); //Introducimos el PIN de la SIM
  if(!correcto) {
    correcto=enviarAT("AT", "OK", 1000);
    if(!correcto)
    {
      escribirLCD("D.Off:REINICIANDO",3);
      reiniciar();

    //Damos dos oportunidades para un inicio correcto
    correcto=enviarAT("AT+CPIN=\"XXXX\"", "OK", 1000);

  }

  }

 //conectando a la RED
  delay (5000);

  //Espera hasta estar conectado a la red movil
  while ( enviarAT("AT+CREG?", "+CREG: 0,1", 1000) == 0 )
  {
  }

 //LLegados a este punto estamos conectados
}
```

El siguiente paso es la configuración APN, implementada igualmente mediante función. En ésta se pasan por comandos AT los datos de configuración anteriormente descritos y se activa el perfil de datos inalámbrico. De nuevo se añade un sistema de control para controlar que los datos se enviarán con garantías. Una peculiaridad del GRPS SIM900 es que no puede controlar más de un socket a la vez, por lo que, si no se controla, podría darse en caso de hacer dos intentos de conexión y por tanto provocar un fallo de sistema.
En el sketch se controla este caso y se cierra la conexión completa para posteriormente reabrirla con las garantías necesarias.

```
void iniciar()
{
  int correcto=0;
  SIM900.print("AT+CGDCONT=1,\"IP\",\"gprs-service.com\"");// Establece parametros PDP
```

```
  SIM900.print((char)13);
  delay(2000);

  escribirLCD("CONECTANDO A APN",3);
  correcto=enviarAT("AT+CSTT=\"gprs-service.com\",\"\",\"1234\"", "OK", 3000); //Definimos el APN, usuario y clave a
utilizar


  if(correcto==0){
    escribirLCD("-SOCKET EN USO-",3);
    cerrarConexion();
    iniciar();
    return;
  }

  escribirLCD("SOLICITANDO CONEXION",3);
  enviarAT("AT+CIICR", "OK", 3000); //Activamos el perfil de datos inalámbrico
  enviarAT("AT+CIFSR", "", 3000); //Activamos el perfil de datos inalámbrico
}
```

Petición HTTP

Por último y como acción bucle del programa, se realizará una conexión “viva” con el servidor seleccionado y se mandarán los datos pasados como query string a una página creada a tal efecto.
En dicha petición se especifican los datos del servidor, puerto, tipo de conexión (TCP), tipo de petición completa (GET + pagina + query string), protocolo usado (HTTP/1.1), host (nombre del servidor), tipo de conexión a mantener (Keep-Alive) y en último lugar el carácter de cierre de comando (emulamos la combinación de teclas CTRL+z mediante el carácter 0x1A).

```
 void PeticionHttp()
 {
 char petDispositivo[20];
   char petValor[20];

   int correcto=0;
   unsigned long tiempoPeticion;
 tiempoPeticion=millis();
 escribirLCD("PETICION PAGINA",3);

 SIM900.print("AT+CIPSTART=\"TCP\",\"www.designea.es\",\"80\"");// Inicia conexion TCP o UDP
 SIM900.print((char)13);

 delay(5000);
 SIM900.print("AT+CIPSEND");// Envias Datos TCP o UDP
 SIM900.print((char)13);
 delay(2000);

 //Archivo donde se enviara el dato y el parametro a enviar
 SIM900.print("GET /sergio/inserta.php?");
 SIM900.print("dispositivo=8");
 SIM900.print("&valor=");
 SIM900.print(valorPH);
 SIM900.print("&latitud=");
 SIM900.print(latitud,6);
 SIM900.print("&longitud=");
 SIM900.print(longitud,6);

 SIM900.print(" HTTP/1.1");

 SIM900.print((char)13);
 SIM900.print((char)10);
 SIM900.print("Host: www.designea.es");//
 SIM900.print((char)13);
 SIM900.print((char)10);
 SIM900.print("Connection: Keep-Alive");
 SIM900.print((char)13);
 SIM900.print((char)10);
 SIM900.print((char)13);
 SIM900.print((char)10);
 SIM900.print("\x1A"); // Finalizacion del comando CTRL+Z

 correcto=enviarAT("AT+CIPCLOSE", "CLOSE OK", 3000);
 if(correcto==0){
    escribirLCD("-FALLO CIERRE-",3);
```

```
    delay(500);
    cerrarConexion();
    reiniciar();
    iniciarRed();
    delay(10000);
    iniciar();
    return;
  }

  escribirLCD("PETICION FINALIZADA",3);
  delay(1000);
      escribirLCD("",3);
      lcd.print((millis()-tiempoPeticion)/1000 );
      delay(tiempoI2C);
      lcd.print(" sg.");
actualizaciones=actualizaciones+1;
delay(1000);
}
```

Reubicar Dispositivo

En caso de querer reubicar (obtener de nuevo la posición GPS) el usuario puede usar el botón instalado. Una vez pulsado al inicio de la secuencia iterativa (no durante la petición u otra parte de la secuencia), se ejecuta la siguiente instrucción:

```
void loop(){
…
// Leemos los botones por si han sido pulsados
  boton_GPS_Estado = digitalRead(boton_GPS);
  boton_Calibracion_Estado = digitalRead(boton_Calibracion);

  if (boton_GPS_Estado==HIGH){  //Si se pulsa el boton GPS se machaca posicion GPS
    escribirLCD("REUBICANDO...",3);
    delay(3000);
    valido=0;
    getValidGPSPosition(tiempoEspera);  //Obtener posición con tiempo límite marcado.
  }

…
}
```

Calibración del dispositivo

Al igual que en el caso anterior en la secuencia iterativa y justo después de comprobar si el usuario quiere reubicar, se comprueba si el usuario quiere calibrar. La secuencia acaba cuando el usuario pulsa de nuevo el botón.

```
void loop(){
…
// Leemos los botones por si han sido pulsados
  boton_GPS_Estado = digitalRead(boton_GPS);
  boton_Calibracion_Estado = digitalRead(boton_Calibracion);

  if(boton_Calibracion_Estado==HIGH){ //Entramos en modo calibracion
    escribirLCD("CALIBRANDO",3);
    delay(3000);
    //Pulsar de nuevo para salir
    while((boton_Calibracion_Estado = digitalRead(boton_Calibracion))==LOW){
      valorPHLCD();  //muestra en pantalla el valor de pH leido
    }
    escribirLCD("FIN CALIBRADO",3);
  }
…
}
```

Como se explicó en la parte hardware del controlador, la calibración se hace con un destornillador ajustando directamente en el propio controlador por lo que la única tarea que hacemos en este punto es mostrar sin retraso el valor de pH medido para que así pueda ajustarlo de una manera más eficiente.

Creación de PCB de componentes.

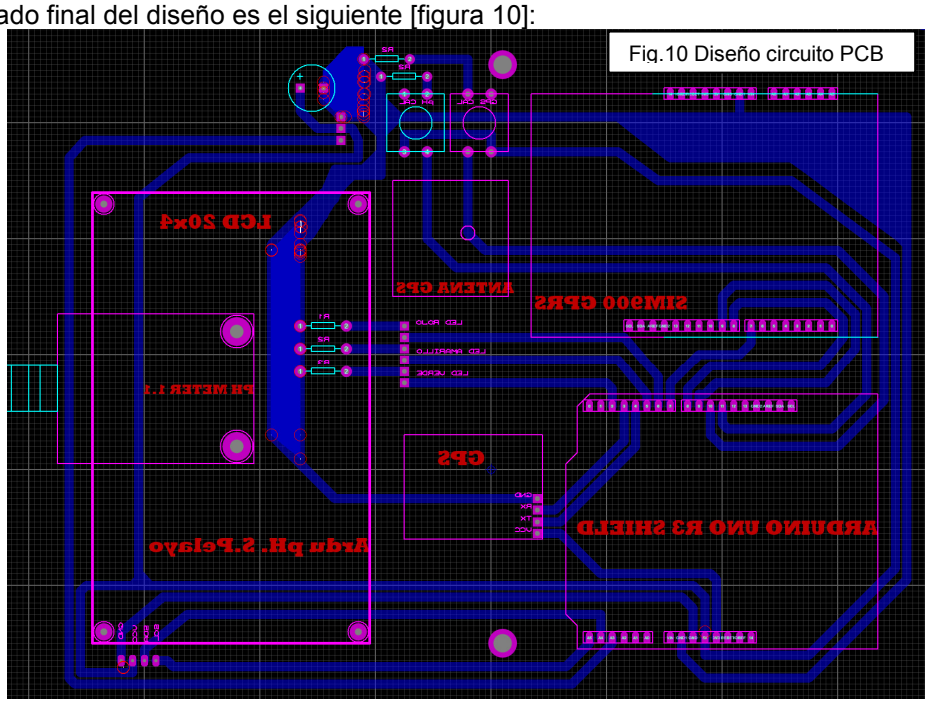
Para obtener un prototipo físico del dispositivo que se está desarrollando se hace completamente necesario desarrollar una placa de circuito impresa (PCB – *"Printed Circuit Board"*). Para ello se deben seguir los siguientes pasos:

1) Diseño de la placa.
2) Impresión de la placa.
3) Inserción de componentes.

Diseño de la placa.

Para hacer un diseño eficaz y sólido, se debe estudiar bien este apartado; la anchura de las conexiones, la separación de corriente y una buena toma a tierra se hacen indispensables para que nuestra placa no falle.
Existen muchos programas de diseño, en este caso se ha usado el programa de diseño "Proteus", donde se ha hecho una transcripción de las conexiones físicas hechas y descritas anteriormente.
El resultado final del diseño es el siguiente [figura 10]:

Fig.10 Diseño circuito PCB

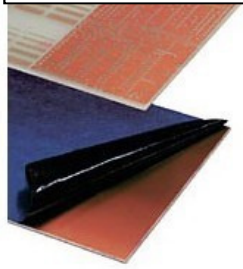


Un detalle a destacar es el tamaño escogido para el diseño. Sin duda éste podría haber sido más compacto, no obstante y como se introdujo en apartados anteriores la autonomía que se requería iba a determinar el alojamiento siendo éste de un tamaño considerado.

Impresión de la placa.

Para la impresión de la placa se debe decidir el tipo de soporte, debido al tamaño del prototipo generado se ha escogido una placa "PKP-104 PLACA FIBRA BUNGARD 1 CARA POSITIVA 150x200mm", el hecho de que sea fotosensible positiva [figura 11] es para poder realizar una impresión con luz y así conseguir un acabado perfecto.

Fig.11 Placa fotosensible

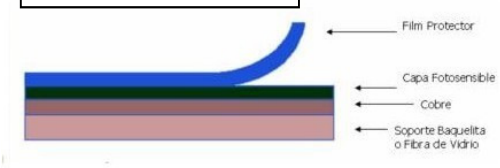


La placa posee distintas capas [figura 12] que se tratarán en distintas fases de la impresión.

- La primera es un film protector de la capa fotosensible, debemos tener en cuenta que si la capa fotosensible se expone a cualquier luz puede deteriorarse y por tanto la impresión final puede ser errónea.

- La segunda es la capa fotosensible en sí, donde plasmaremos nuestro circuito.

Fig.12 Partes de la placa



- La tercera es una lámina de cobre, la cual hará de conductor entre elementos, y cuyas partes no necesarias se deben eliminar.
- Por último posee el soporte del circuito, en este caso de fibra de vidrio.

A continuación se detallan las especificaciones de la placa usada:
- Material: FR4 Epoxy / Fibra de Vidrio laminada
- Espesor del soporte: 1.5 mm.
- Espesor de la Capa de Cobre: 35 micras
- Adhesivo protector azul.
- Dimensiones: 150x200 mm.

Para la impresión en la capa fotosensible se usará una impresión en acetato del circuito creado con Proteus y un insolador [figura 12].

Fig.12 Proceso de insolación con acetato

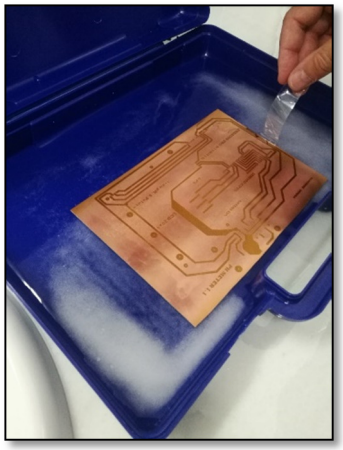


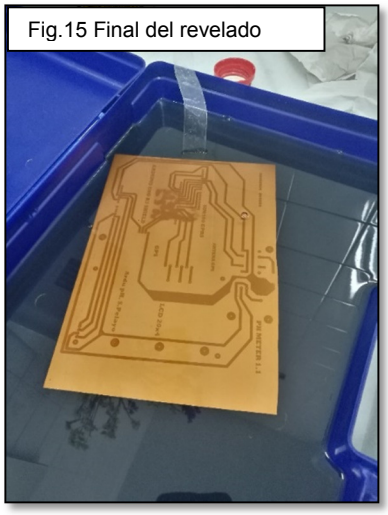
Fig.13 Revelado de la placa

Una vez insolada, deberemos revelar el resultado para poder apreciar como la lámina que ha sido insolada (no cubierta por las pistas), se desprenderá de la placa y por tanto expondrá el siguiente elemento de la placa (en este caso el cobre) [Figura 13].

La placa se revela con movimientos sutiles y aclarado en agua [Figuras 14-15].

Fig.14 Aclarado del revelado

Fig.15 Final del revelado



Una vez aclarada en agua debemos de aplicar un atacador para la lámina de cobre, de modo que éste destruirá todo menos la parte no insolada y por tanto protegida.

El atacador usado ha sido ácido clorhídrico al 33% y una solución base [Figuras 16-17]

Fig.16 Atacador cobre – Ácido clorhídrico

Fig.17 Base a mezclar con el ácido

Al igual que antes para revelar, la placa se atacar con movimientos sutiles y precaución:
Rápidamente la parte expuesta de cobre se destruye (obsérvese como se pone de color verde la solución, signo de la disolución de cobre en ella en la figura 18).
Finalmente obtenemos la placa de cobre inicialmente diseñada con pistas conductoras de cobre:

Fig.18 Proceso de ataque al cobre con la solución

Fig.19 Final del proceso de atacado.

Inserción de componentes.

En último lugar en la creación del a PCB es la inserción de componentes; pin a pin se han ido taladrando cada orificio. Una vez hecha esta parte se insertan y sueldan los componentes [Figuras 20-21].

Fig.20 Proceso de taladrado de pines.

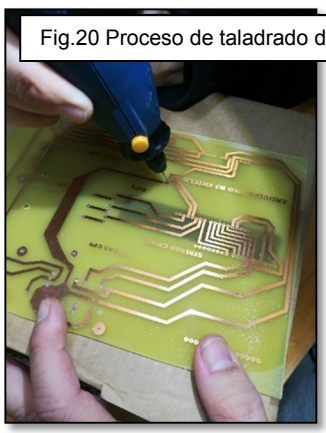


Fig.21 Soldado de componentes.

Los leds de información se han hecho de manera que puedan ser encastrados en pasos posteriores a la tapa de la caja; para ello se han creado cables específicos y crimpados con conectores especiales [Figuras 22-23]

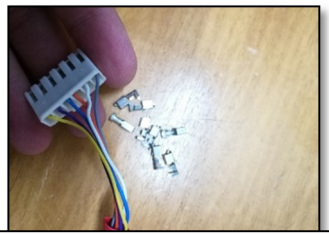


Fig.22 Conector cable Molex de 6 pines

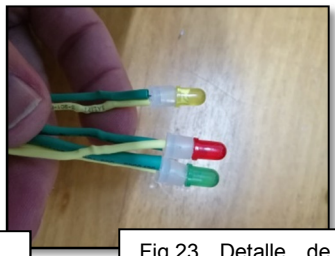
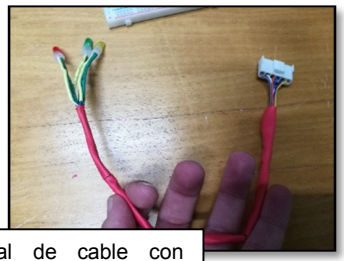


Fig.23 Detalle de final de cable con protectores.



Selección y adaptación del soporte.

Para el encastrado de la PCB así como los demás elementos que lo conforman (sonda, interruptores, puerto de carga…) se ha elegido para el prototipo una caja eléctrica de tamaño proporcional al circuito y sobre todo a la batería elegida.

Fig.24 Caja seleccionada.

Como se puede apreciar la caja [Figura 24] está totalmente limpia y habrá que adaptarla al requerimiento del proyecto.
Por una parte necesita de aperturas exteriores cara a poder mostrar la pantalla LCD, una ranura para la sonda, una ranura para el puerto de carga, tres boquetes para presentar los leds de información, una apertura lateral para ver el nivel de la batería y otro boquete lateral para el interruptor del dispositivo.
En segundo lugar hay que compartimentar el interior para dar cabida a la batería (en un nivel) y a la propia PCB (en otro nivel), de manera que quede aprovechada la altura.
Creación de moldes internos y embellecedores.
El primer elemento que se ha diseñado es el principal interior que divide a la caja en dos niveles: uno inferior para soportar y contener a la batería y otro superior para la propia placa. Tanto para éste como para el resto de embellecedores se ha usado el programa de diseño “Solid Edge V19”, con el cual y tras realizar un diseño en 3D del soporte se exportará mediante el programa “Ultimaker Cura 3.0.4” a una impresora 3D para materializar el proyecto.

Soporte principal interior.

A continuación se muestra el diseño final del soporte tras medir cada parte del interior de la caja y el resultado final [Figuras 25-26]

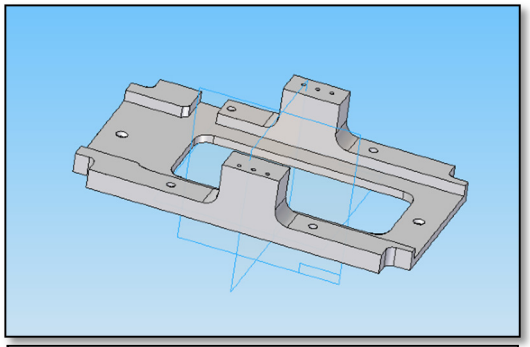


Fig.25 Diseño Solid Edge para el soporte interno.

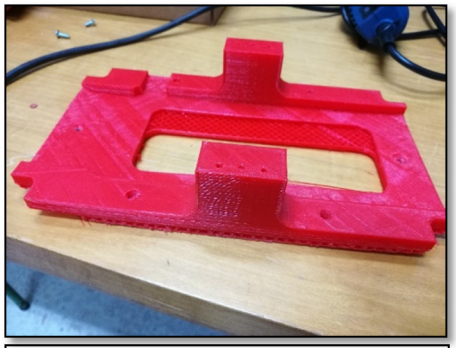


Fig.26 Prototipo del soporte interno.

En la realización de la impresión se han impreso varias carcasas hasta dar con la consistencia requerida, en este caso una densidad del 30% ha sido suficiente para atender a los requerimientos de robustez y rigidez.

Se han realizado varios orificios para sujetar el propio soporte a la caja y para contener a la batería (para ello también se han diseñado dos abrazaderas). El resultado de encajar el soporte en la caja y las dos abrazaderas es:
Para hacer las dos presillas que se ven se ha seguido el mismo procedimiento.

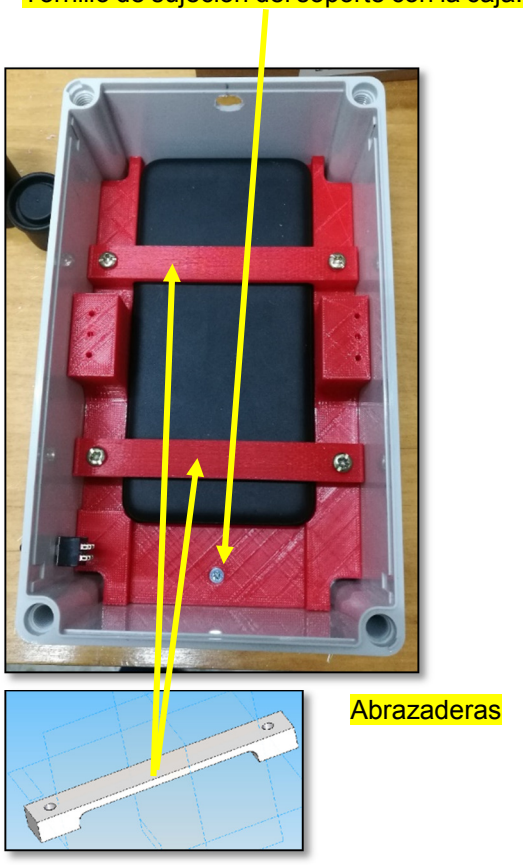


Fig.27 Soporte con batería y abrazaderas.

Fig.28 Soporte, batería y PCB.

Después de asegurar la batería se puede emplazar la placa. Esta vista será la que nos determine la apertura que se debe hacer en la tapa de la caja y su posterior embellecedor [Figura 28].

Apertura en tapa de Caja para LCD y embellecedores.

En la tapa se realiza una apertura 78x28mm y a continuación se diseña e imprime el protector y embellecedor. Éste se realiza en dos partes para así dejar un protector de metra quilato en su interior. [Figuras 29-30-21]

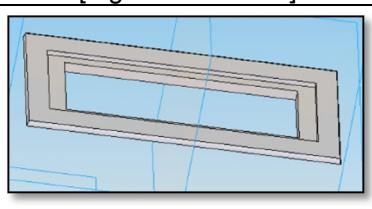


Fig.29 Diseño parte baja

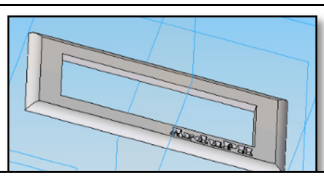


Fig.30 Diseño parte alta embellecedor.

Fig.31 Impresión embellecedor LCD

Justo en el lateral de la pantalla se disponen los embellecedores Led (éstos si han sido comprados) y un protector traslucido de éstos (fabricados ex profeso).

El diseño final de la tapa de la carcasa es el siguiente:

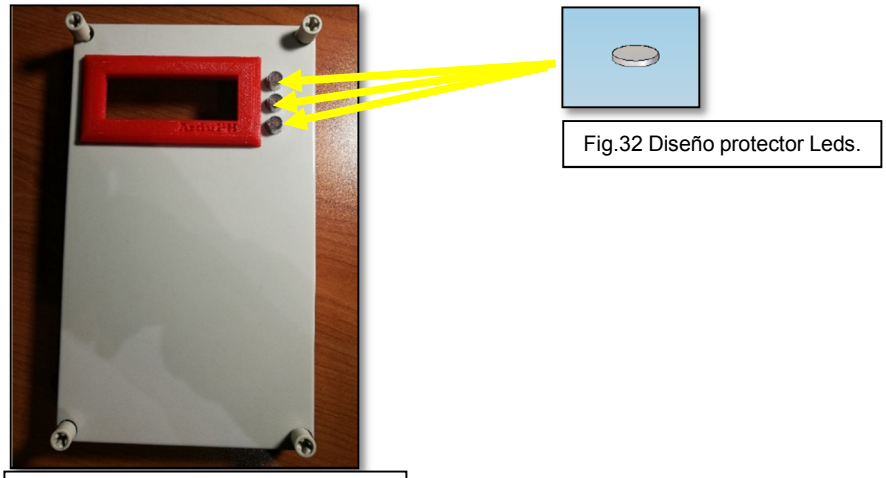


Fig.32 Diseño protector Leds.

Fig.33 Resultado final tapa de la caja.

Embellecedores y elementos laterales.

En los laterales se hacen los siguientes orificios: puerto de carga, boquete para interruptor, orificio para ver nivel de carga y boquete para sonda.
Todos excepto el boquete de la sonda de pH han sido cubiertos con algún embellecedor.

Embellecedor conector de carga

EL puerto carga tipo USB-B se ha instalado para ampliar el nivel de carga de la batería sin tener que abrir la caja, igualmente se usaría para suministrar alimentación externa continua al dispositivo para así dotar de una autonomía mayor (o perenne). [Figura 34]

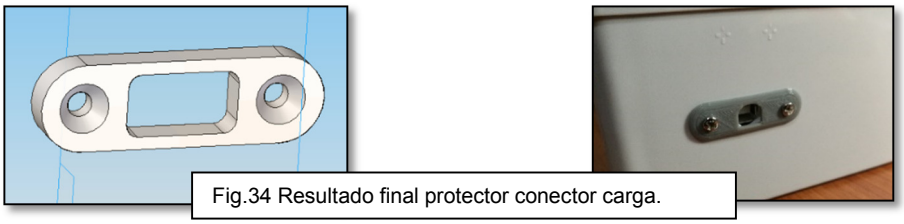


Fig.34 Resultado final protector conector carga.

Orificio para visualizar nivel de batería.

En el lateral se ha realizado una ventana a la caja para poder ver cuanta autonomía le queda [Figura 35].

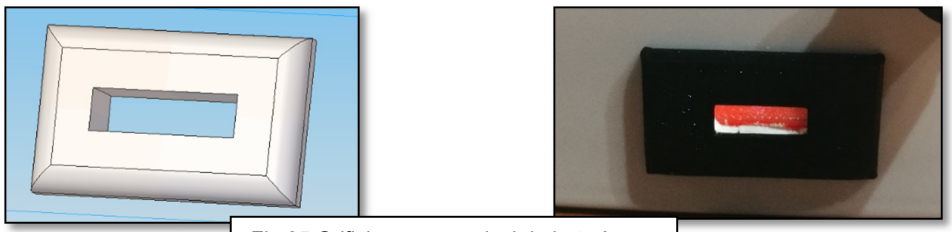


Fig.35 Orificio para ver nivel de batería.

Interruptor del circuito.

Justo en la parte superior de la ventana para ver la batería se ha instalado un interruptor tipo "llave" para dar corriente al dispositivo [Figura 36]. En vez de realizar la toma de corriente desde alguno de los puertos que posee el Arduino se ha optado por insertar 5v directamente en placa,

Fig.29 Ventana para ver batería.

por lo que se ha soldado en los pines de 5v y GND un conector molex de 2 pines [Figura 37]. Este se conecta directamente a GND y VCC del USB-A de la batería; para dar corriente o no el interruptor interrumpe la señal VCC de la batería. Igualmente se ha fabricado una presilla para que el cable USB no sufra daños con el tiempo.

Fig.37 Interior interruptor y alimentación de placa

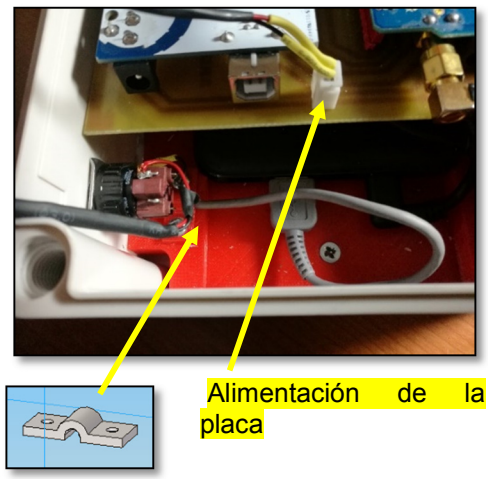


Fig.36 Detalle del interruptor usado.

Asa y retenedor cable de la sonda

La caja en sí tiene mucha dificultad al agarre por lo que se ha creado un asa que sirve para sujetar la caja en sí así como para alojar el cable dela sonda. Un dato a tener en cuenta es sin duda la longitud del cable de la sonda; éste estará limitado por la profundidad que tengan las arquetas y por tanto deberá ser suficientemente largo como para llegar sin necesidad de ninguna extensión. Para el escenario objetivo se ha escogido un cable de 3 metros de largo [Figura 38].

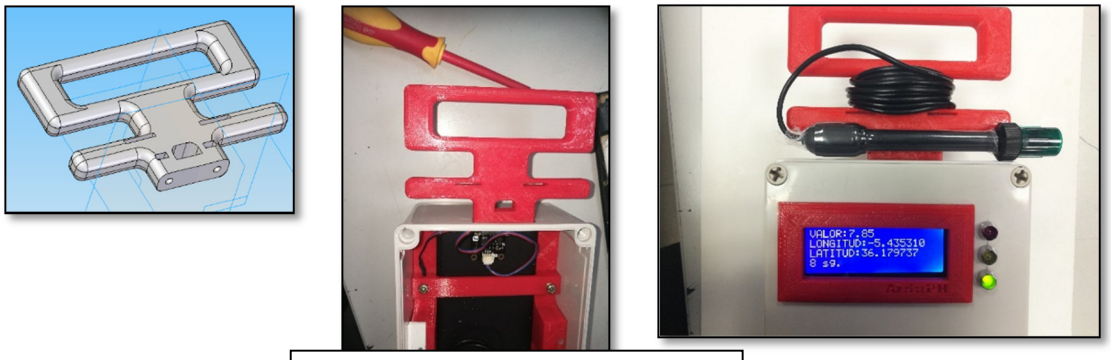


Fig.38 Montaje asa/sujeta cables.

Terminada esta parte se procede a la elaboración del servicio cliente donde se deberá escoger el sistema gestor de base de datos y servidor como elementos principales.

Análisis de la base de datos del sistema

El sistema en sí deberá ser capaz de almacenar los datos de valor que le mande el dispositivo, igualmente deberá registrar valores de tolerancia numérica (que datos son válidos y cuales no), gestionar alarmas y tener la capacidad de recopilar datos de diferentes dispositivos ubicados en distintos puntos del planeta.
Un análisis válido es el que se muestra en el siguiente esquema entidad-relación:

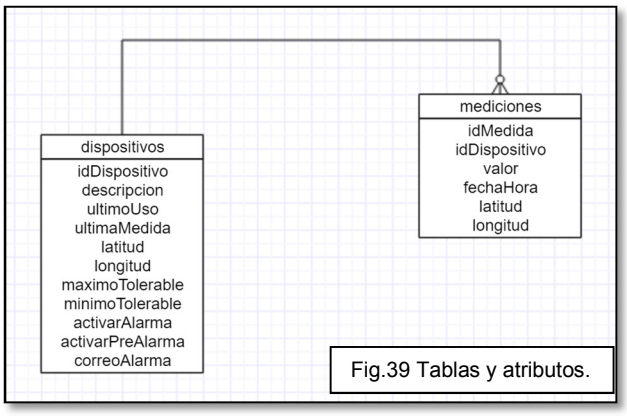


Fig.39 Tablas y atributos.

Sistema gestor de base de datos escogido.

El sistema gestor de base de datos escogido es MySQL por tanto una estructura válida es la siguiente:

Tabla “dispositivos”:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**idDispositivo**	int(11)			No	*Ninguna*
2	**descripcion**	text	utf8_general_ci		No	*Ninguna*
3	**ultimoUso**	datetime			No	*Ninguna*
4	**ultimaMedida**	decimal(10,2)			No	*Ninguna*
5	**latitud**	decimal(10,6)			No	*Ninguna*
6	**longitud**	decimal(10,6)			No	*Ninguna*
7	**maximoTolerable**	decimal(10,2)			No	8.20
8	**minimoTolerable**	decimal(10,2)			No	6.20
9	**activarAlarma**	tinyint(1)			No	1
10	**activarPrealarma**	tinyint(1)			No	0
11	**correoAlarma**	varchar(120)	utf8_general_ci		No	sergiopelayo@gmail.com
12	**ultimaNotificacion**	datetime			No	*Ninguna*
13	**intervaloNotificaciones**	int(11)			No	5

Tabla “mediciones”:

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**idMedida**	float			No	*Ninguna*
2	**IdDispositivo**	int(11)			No	*Ninguna*
3	**valor**	decimal(10,2)			No	*Ninguna*
4	**fechaHora**	timestamp			No	CURRENT_TIMESTAMP
5	**latitud**	decimal(10,6)			No	*Ninguna*
6	**longitud**	decimal(10,6)			No	*Ninguna*

Servidor y alojamiento.

Una vez sabemos los requisitos del sistema gestor podemos seleccionar el servidor. En nuestro caso queremos que los datos sean accesibles desde cualquier lugar, ya sea desde dentro de la corporación como desde fuera. Para el proyecto se ha decido escoger un servidor alojado en Internet y por tanto ya de por sí accesible. Los datos del servidor importantes son que soporte PHP 7.0 y configuración del servidor MySQL.

Se aprovecha el servidor alojado en Hostytec para el dominio “designea.es” para las pruebas del dispositivo.

En la sección de base de datos se crea una base de datos llamada “TFG_PH” con acceso para el usuario “ACX_USER”. Los datos de acceso se guardan en un archivo en el propio servidor, de modo que no exponemos el usuario y contraseña en cada página que se desarrollará. Igualmente se dan de altas las tablas fruto del análisis anterior.

Conexión con base de datos.

Con la finalidad de no exponer en cada página los datos de acceso, se ha creado el archivo de conexión “funciones.php” donde se introducen los datos de acceso así como algunas funciones que serán de utilidad en el desarrollo de la plataforma.

Plataforma WEB

A continuación se detallas las distintas partes de la plataforma de control de los dispositivos, tanto para ver los propios dispositivos como para su gestión.
La página al completo se puede visitar en la web http://designea.es/sergio

Estructura general de la plataforma.
La plataforma se estructura de forma simple y clara, de modo que toda la funcionalidad de la misma se basa en las dos partes base del proyecto: la geolocalización y el control de los valores en tiempo real.
La plataforma se organiza del siguiente modo [Figura 40]:

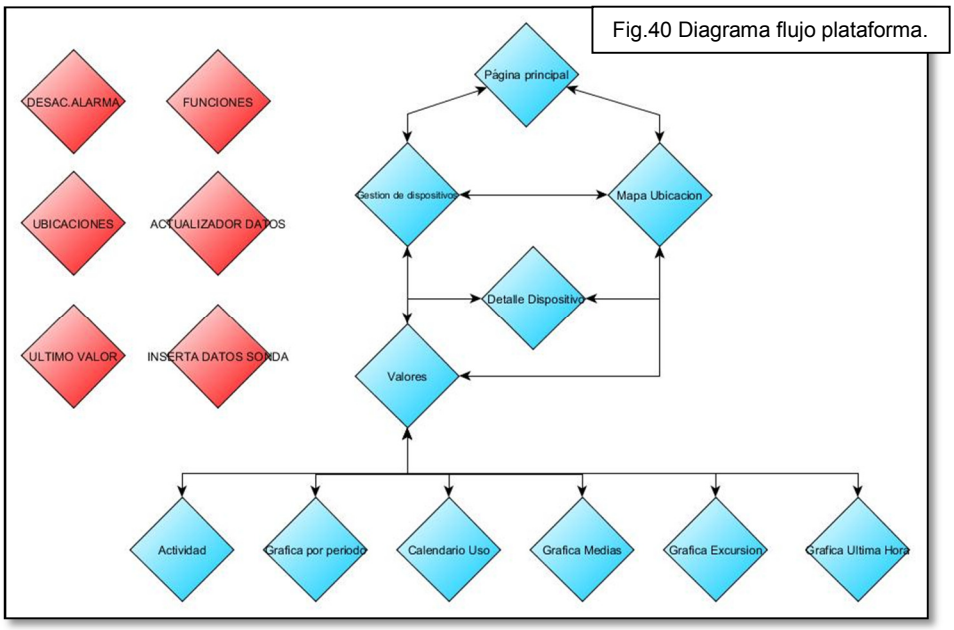


Fig.40 Diagrama flujo plataforma.

Se diferencian (en color rojo) varias páginas que no son visibles para el usuario y que sin embargo son las encargadas de ejecución de funciones, conexión de base de datos, actualizar datos de los dispositivos, gestión de alarmas, inserción de valores, control de avisos de las posibles alarmas y de dar respuesta con los últimos valores insertados.
Como se aprecia en el diagrama la página no posee demasiada profundidad, de modo que cualquiera de ellas es accesible con un máximo de 3 clics.
Imagen de la marca y dispositivo.
En todas las páginas se ha incluido un diseño de marca que se incluirá en éste y dispositivos próximos y se añade el específico del propio proyecto; para ello se ha creado ex profeso el logotipo mostrado en la figura 41.

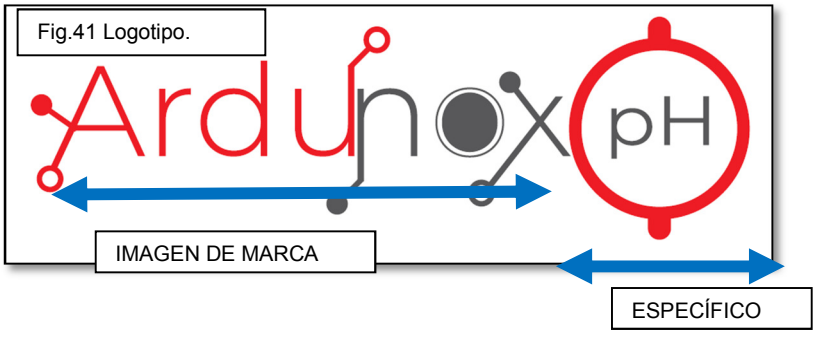


Fig.41 Logotipo.

Página de inserción de datos de sonda.

Archivo físico: “inserta.php”

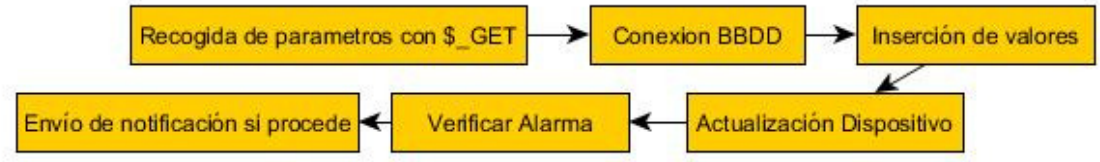
Esta página no es accesible directamente desde ningún vínculo de la plataforma, sin embargo es la base de todo ya que se encarga de la introducción de los datos procedentes de cada dispositivo. La página se llama “inserta.php” y recibe como parámetros de entrada el identificador del dispositivo, la latitud, longitud y el valor leído por la sonda. También es la encargada de enviar la alerta (que se hace por medio de correo electrónico) si el dispositivo lo tiene configurado.

Igualmente mantiene los últimos datos actualizados en la tabla dispositivos, de manera que de un solo vistazo recorriendo la tabla “dispositivos” podemos ver la última secuencia insertada en la tabla “mediciones”.

Se ha implementado en ella la funcionalidad del envío de correos en caso de alarmas [Figura 42], de modo que cuando un dispositivo que tenga las alarmas activas inserte un valor que sobrepase los límites (inferior o superior) y no se haya enviado una notificación en los anteriores minutos (intervalo definido en gestión del dispositivo) enviará un correo con el aviso:

Fig.42 Correo de alarma

Diagrama página:



La función de conexión con la base de datos se implementa en el archivo funciones.php ya que será de uso en todas las páginas desarrolladas. Igualmente se han comentado en código todos los detalles propios del lenguaje de programación PHP (versión 7.0).

Del código desarrollado se destacan las siguientes partes:

```
<?php
include "funciones.php";  //Inclusión de fichero con claves de acceso y funciones comunes
$conexion=conectarse();//Conexión con la base de datos desarrollada en funciones.php
…
$dispositivo=$_GET["dispositivo"]; //Recogida de datos pasados en URL mandado con el Arduino
$latitud=$_GET["latitud"];
$longitud=$_GET["longitud"];
…
$crite="INSERT INTO mediciones (idDispositivo,valor,latitud,longitud) VALUES
(".$dispositivo.",".$valor.",".$latitud.",".$longitud.")"; //Consulta inserción
…
mysqli_query($conexion,$crite)or die("<br/>Problema en CONSULTA");; //Ejecución consulta
….
if ($row['minimoTolerable']>$valor || $row['maximoTolerable']<$valor) //Coprobación de valor si sobrepasa los límites
…
//Para no saturar al usuario con emails se programa un tiempo mínimo entre ellos, de esta manera se controla
Sumaperiodo se desarrolla en funciones.php
if (sumaperiodo($row['ultimaNotificacion'],"+".$row['intervaloNotificaciones']." minute")<date("Y-m-d H:i:s
…
//Cuando se envía se actualiza el campo de ultima notificación de la tabla dispositivos.
$sqlUpdate="UPDATE `dispositivos` SET `ultimaNotificacion`="".Date("Y/m/d H:i:s"). "" where
idDispositivo=$dispositivo;";
mysqli_query($conexion,$sqlUpdate);
…
?>
```

Página de funciones y conexión.

Archivo físico: “funciones.php”.
En ella se especifican los datos de acceso al servidor, así como las funciones que se usarán en el resto de páginas, entre ellas la más importante de todas, la propia conexión con la base de datos.
Cabe destacar la función de conexión y cierre con la base de datos:

```
########   CONEXIONES:  ##################################################

function conectarse()
{
        $server = "localhost";
        $user = "ACX_USER";
        $pass = "xxxxxx";
        $bd = "TFG_PH";

        //Creamos la conexión
        $conexion = mysqli_connect($server, $user, $pass,$bd)
        or die("Ha sucedido un error inesperado en la conexion de la base de datos");
        mysqli_set_charset($conexion,'utf8');

        return $conexion; //devolvemos la conexión
}

#########################################################################
function cerrar($conexion)
{
if (!mysql_close($conexion)) {echo "<script>window.location='error.php?mensaje=8'</script>";}
}
#########################################################################
```

Actualizador de datos.

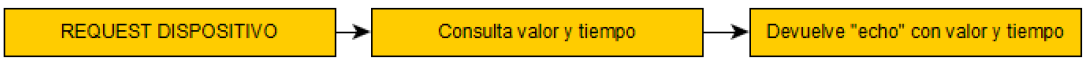
Archivo físico: “pelayo_guardar_configuracion.php”.
Encargado del guardado de datos disponible en el detalle de la gestión de dispositivos.

A diferencia del archivo anterior la recogida de datos se ha desarrollado con el comando $_REQUEST. El resto de código es técnicamente igual que en el apartado anterior.

```
…
extract($_REQUEST);          #con esta orden capturo en variables separadas todos los valores del formulario
…
```

Ultimo valor.

Archivo físico: “pelayo_ultimo_valor.php”.
Este archivo es esencial para cargar de manera dinámica los valores más actuales en la página de valores de cada dispositivo. Gracias a ella no se tendrá que cargar la página de valores al completo sino desde la página de valores y usando AJAX como base se solicitará cada un intervalo de tiempo definido ésta para obtener siempre el último valor inserto y el tiempo transcurrido entra la lectura y la hora actual.




```
<?
…
//Detalle de la consulta para obtener el ultimo valor y el tiempo que ha pasado desde que se actualizó
$sql="SELECT valor,TIME_TO_SEC(TIMEDIFF(NOW(),fechaHora)) AS transcurridos FROM mediciones
WHERE idDispositivo='$dispositivo' ORDER BY fechaHora DESC LIMIT 1";
…
if($datos['transcurridos']>0){ //Con este dato tengo en cuenta los dispositivos que no han sido actualizados nunca
        $transcurridos=$datos['transcurridos'];
}
echo $datos['valor']."~".$transcurridos;
//Devolvemos con un echo el valor y el tiempo transcurrido separado por el carácter ~
?>
```

Ubicaciones.

Archivo físico: “pelayo_mapas_marcas.php”.
Página no accesible directamente que crea la estructura de datos necesaria en XML para que sea leída desde la página “pelayo_mapa.php”.
Ejecutamos la página directamente nos devuelve una página con formato XML con los valores de dispositivos [Figura 43].

```
<markers>
  <marker name="1" valor="5.86" lat="36.180135" lng="-5.420674" ultimoUso="2017-12-16
  12:26:54" icono="img/icon_rojo_dispositivo.png" descripcion="Primer Dispositivo Creado
  36.180135, -5.420674"/>
  <marker name="8" valor="0.00" lat="36.147541" lng="-5.457543" ultimoUso="2017-12-19
  08:41:53" icono="img/icon_rojo_dispositivo.png" descripcion=" Arqueta 43 Colector
  C234534543"/>
</markers>
```

Fig.43 Formato de salida de la página

Es la encarga de devolver también el valor de actividad, de modo que si el dispositivo está activo se usará un icono en mapa de color verde y si no lo está devolverá un icono en rojo (previamente diseñados e incluidos en la carpeta “img” del servidor.

Recogida datos $_GET → Consulta datos del dispositivo → Creación con "echos" datos en formato XML

```
$dispositivo=$_GET["dispositivo"];
…
//Si no le pasamos el dispositivo se ejecuta una consulta para ver todos los dispositivos dados de alta.
if (strcmp($dispositivo,"")==1)
    $sql = "SELECT * FROM dispositivos ";
…
//Bucle para recorrer los dispositivos y crear con “echo” el formato XML
while ($row =mysqli_fetch_assoc($res)){
  // Add to XML document node
  echo '<marker ';
  echo 'name="' . parseToXML($row['idDispositivo']) . '" ';  //Esta función elimina los caracteres especiales
  echo 'valor="' . parseToXML($row['ultimaMedida']) . '" ';
  echo 'lat="' . $row['latitud'] . '" ';
  echo 'lng="' . $row['longitud'] . '" ';
  echo 'ultimoUso="' . $row['ultimoUso'] . '" ';
…
//Se verifica el tiempo de actividad para poner un led u otro en mapa (Rojo o verde según actividad)
if ($datos2['tiempoInactivo']<120 && $datos2['tiempoInactivo']>0)
    echo 'icono="img/icon_verde_dispositivo.png" ';
…
```

Página de desactivación de alarmas.

Archivo físico: “desactiva_alarma.php”.
Los enlaces a esta dirección sólo se lanzan en los correos de alerta enviados desde la página “inserta.php”. El único cometido es desactivar el valor de “activar Alerta” del dispositivo que está generando la alarma, de modo fácil y rápido. Otra manera indirecta es desde la propia web en la gestión del dispositivo.

Recogida datos $_GET → Actualización Campo ALERTA → Muestra mensage de nuevo estado


```
<?php
include "funciones.php";
$conexion=conectarse();
$dispositivo=$_GET["idDispositivo"];
$sqlUpdate="UPDATE `dispositivos` SET
`activarAlarma`=0 where idDispositivo=$dispositivo;";
echo "Alarma del dispositivo DESACTIVADA";
mysqli_query($conexion,$sqlUpdate);
?>
```

Fig.44 Pagina desactivar alarma

Pantalla principal de acceso

Archivo físico: “index.html”.
Desde ella se podrán acceder básicamente a las dos funcionalidades de mayor importancia:

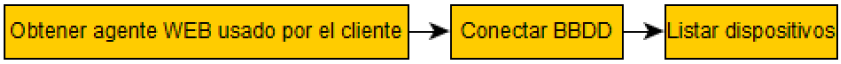
1) Gestión de dispositivos.
2) Ver un mapa general con los datos últimos de valor de todos los dispositivos.

Fig.45 Página principal.

Pantalla gestión de dispositivos.

Archivo físico: “Pelayo_gestion.php”.
En ésta se listan todos los dispositivos actuales dados de alta en el sistema. Igualmente ya se puede desde este apartado ver un mapa específico mostrando de manera centrada el dispositivo con el valor actual, acceder a los valores actualizados del dispositivo así como editar los campos del mismo [Figura 46].

Al principio del código se ha usado la variable obtenida con “*$_SERVER['HTTP_USER_AGENT']*” en varias condiciones para determinar que cliente WEB se utiliza y así poner un estilo de diseño u otro para adaptar mejor el contenido.



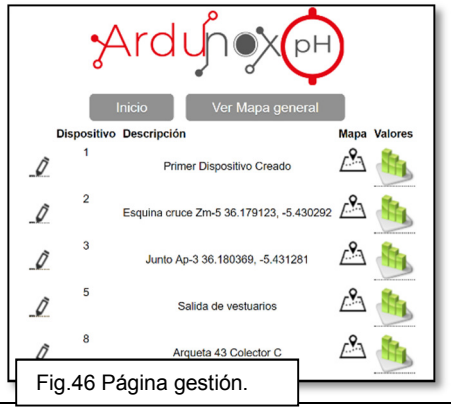


Fig.46 Página gestión.

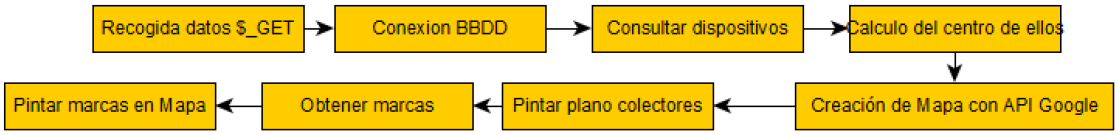
```
if (preg_match('/(tablet|ipad|playbook)|(android(?!.*(mobi|opera mini)))/i', strtolower($_SERVER['HTTP_USER_AGENT'])))
…
if (preg_match('/(up.browser|up.link|mmp|symbian|smartphone|midp|wap|phone|android|iemobile)/i', strtolower($_SERVER['HTTP_USER_AGENT']))) {
…
if ((strpos(strtolower($_SERVER['HTTP_ACCEPT']),'application/vnd.wap.xhtml+xml') > 0) or ((isset($_SERVER['HTTP_X_WAP_PROFILE']) or isset($_SERVER['HTTP_PROFILE'])))) {
…
//Condiciones para averiguar el cliente WEB usado.
```

Pantalla mapa general.

Archivo físico: “Pelayo_mapa.php”.
La página muestra un mapa de Google con la ubicación de los dispositivos, igualmente se han integrado un sistema de colectores de prueba para ver realmente la integración posible en un área real usando poli líneas. Técnicamente las marcas de los dispositivos se han integrado desde la página externa llamada “pelayo_mapas_marcas.php” explicada anteriormente.

Fig.47 Mapa con detalle de los colectores y dispositivos (Rojo: desconectado - Verde: Activo).

Diagrama de la página



A continuación se detallan los puntos clave:

```
… //Creación del mapa y centrado según centro calculado con etiquetas php; zoom igualmente personalizado
function initMap() {
        var map = new google.maps.Map(document.getElementById('map'), {
          center: new google.maps.LatLng(<?echo $centroLat;?>, <?echo $centroLong;?>),
          zoom: <?echo $zoom;?>,
                        mapTypeId: google.maps.MapTypeId.SATELLITE
        });
…
//Creación de la polilinea y puesta en mapa , una variable por cada colctor pintado…
var colectorA = [
{lat: 36.183251, lng:-5.426278  },//Inicio
{lat: 36.183516, lng:-5.425521   },//Barrera parkin oficina
{lat: 36.183956, lng:-5.425762  },//entradilla
{lat: 36.183516, lng:-5.425521   },//Barrera parkin oficina
…
];

var colectorA_mapa = new google.maps.Polyline({path: colectorA,geodesic: true,strokeColor:
'#0000FF',strokeOpacity: 1.0, strokeWeight: 5 });
 colectorA_mapa.setMap(map);
…
//Descarga de marcas, se hace la llamada a la página creada, descargamos y pintamos las marcas en el mapa
 var pagina='http://designea.es/sergio/pelayo_mapas_marcas.php?dispositivo=<?echo $dispositivo;?>';
    downloadUrl(pagina, function(data) {
var xml = data.responseXML;
 var markers = xml.documentElement.getElementsByTagName('marker');
Array.prototype.forEach.call(markers, function(markerElem) { //por cada marca se pinta en mapa un marcador
…
```

La función initMap inicializa y agrega el mapa cuando se carga la página web. Usa una etiqueta script para incluir JavaScript propio que contenga la función initMap.

```
    <script async
    src="https://maps.googleapis.com/maps/api/js?key=MIKEYDEGOOGLE&callback=initMap">
    </script>
```

Analizando en detalle la línea podemos observar varios parámetros:

- El elemento script carga la API desde la URL especificada.
- El parámetro callback ejecuta la función initMap cuando la API se carga por completo.
- El atributo async permite al navegador seguir procesando el resto de la página mientras se carga la API. Si quisiéramos que parase pondríamos el atributo “defer”.

- El parámetro key contiene la clave API. Esta clave es necesaria para el uso de la misma, podría usarse una clave pública o bien generarse desde la consola de desarrollador de Google.

Detalle de dispositivo.

Archivo físico: “Pelayo_configurar.php”.
En esta página se muestran todos los campos editables del dispositivo además de aquellos valores que se han integrado por último. Se podrán editar los elementos de alarma, valor máximo tolerable, mínimo tolerable, correo de alarma e intervalo en minutos entre notificaciones. Cuando pulsemos el botón “Guardar” hará una llamada al archivo antes explicado “pelayo_guardar_configuracion.php” [Figura 48].

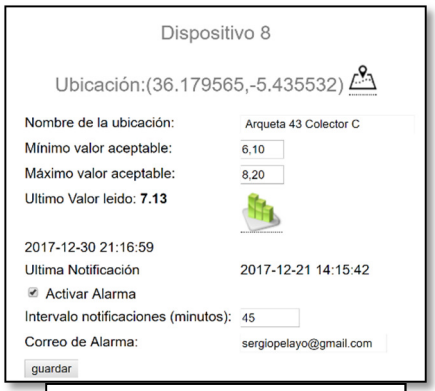
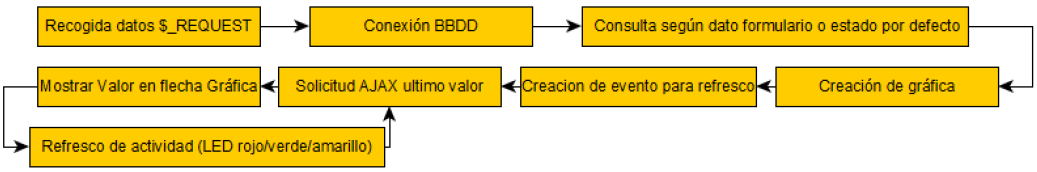


Fig.48 Página edición dispositivo.

Control principal de valores del dispositivo.

Archivo físico: “Pelayo_principal.php”.



Esta página es la encargada de mostrar los datos leídos por el dispositivo; igualmente muestra el estado de actividad por medio de una imagen (LED ROJO/VERDE/AMARILLO). Desde esa misma sección se han puesto cuadros de lista y periodos de rango de fechas (predefinidos o totalmente elegibles por el usuario). Cada uno de ellos hace cargar en el gráfico inferior los datos de valores en el intervalo.

Fig.49 Selección de periodos de valores

Fig.50 Gráfica Valores

Parte Verde
valores admitidos
Parte Anaranjada
Tolerancia 10%
Parte Roja
Fuera de límites
Abanico Azul
Valores que ha tenido en el intervalo (en este caso todos los valores del 0-14)
Flecha
Indica Valor Actual

Ésta y todas las gráficas generadas se han hecho usando funciones con soporte en JavaScript creadas por Fusión Charts (https://www.fusioncharts.com). El código completo de la gráfica es extenso por lo que a continuación se muestra la rutina para obtener los valores principales de la gráfica y para la actualización del valor instantáneo, el resto está comentado en el propio código del archivo de la página que se entregará junto a la presentación.

```
//Variables para personalización de la gráfica
/// consulta de configuración de este dispositivo:
$sql="SELECT * FROM dispositivos WHERE idDispositivo='$dispositivo'";
$res=mysqli_query($conexion,$sql);
$datos=mysqli_fetch_assoc($res); # registro único

$nombre=$datos['descripcion'];
$minimo_aceptable=$datos['minimoTolerable'];
$maximo_aceptable=$datos['maximoTolerable'];
$margen=($maximo_aceptable-$minimo_aceptable)*0.1; //10%
$minimo_plus=$minimo_aceptable+$margen; // 10% antes de llegar para color amarillo
$maximo_plus=$maximo_aceptable-$margen; // 10% antes de llegar para color amarillo


/// CONSULTA DEL RANGO:
/* se mostrará el rango de valores del periodo seleccionado en azul en el medidor instantáneo*/
#mínimo:
$sql="SELECT min(valor) as minimo FROM mediciones WHERE idDispositivo='$dispositivo'
AND fechaHora>='$desde'
AND fechaHora<='$hasta'";
$res=mysqli_query($conexion,$sql);
$datos=mysqli_fetch_assoc($res); # registro único
$minimo=$datos['minimo'];

#máximo:
$sql="SELECT max(valor) as maximo FROM mediciones WHERE idDispositivo='$dispositivo'
AND fechaHora>='$desde'
AND fechaHora<='$hasta'";
$res=mysqli_query($conexion,$sql);
$datos=mysqli_fetch_assoc($res); # registro único
$maximo=$datos['maximo'];

//Creación de la gráfica y paso de valores:
<script>
FusionCharts.ready(function () {
    var cSatScoreChart = new FusionCharts({
                        id: 'medidor',
        type: 'angulargauge',
        renderAt: 'chart-container',
        width: '500',
        height: '300',
        dataFormat: 'json',
        dataSource: {
            'chart': {
                'caption': 'Dispositivo $dispositivo <br>".$nombre_periodo[$periodo]."',
                'subcaption': '$descripcion',
                'plotToolText': 'Valor pH actual: $ultimo_valor',
                'theme': 'fint',
                'chartBottomMargin': '50',
                'showValue': '1',
                                                'valueBelowPivot': '1',
                                                'valueFontSize': '12',
                                                'valueFontBold': '1'
            }
…

//Creo el evento en la gráfica (javascript), éste obiene de un div oculto el valor que tiene cada 3 sg.
…
events: {
            'rendered' : function (evtObj, argObj){
                var intervalVar = setInterval(function () {
                    var chartIns = evtObj.sender,
                    valor=document.getElementById('nuevo_valor').innerHTML.split(\"~\")[0]
                                                        chartIns.feedData('value='+valor);

                }, 3000);
            }
        }
…
```

```
//Llamada a la función AJAx y rutina cada 3 sg.
ajaxFunction();
setInterval('ajaxFunction()',3000); // ejecuta el script que recoge el último valor cada 3 segundos
…
// crea la función que recibirá los datos enviados por el servidor y los meterá en el div "nuevo_valor":
  ajaxRequest.onreadystatechange = function(){
            if(ajaxRequest.readyState == 4){
                        var ajaxDisplay = document.getElementById('nuevo_valor');
                        ajaxDisplay.innerHTML = ajaxRequest.responseText;
            }
  }
…
//Llegados a este punto también actualizamos el led de control: amarillo lectura de valor, rojo inactivo, verde activo.
…
//Cuando el evento de la gráfica cargue el valor del div oculto éste estará actualizado gracias a la rutina anterior.
```

Gráfica última Hora

Archivo físico: “grafica_ultima_hora.php”.
Muestra los datos del dispositivo en la última hora contando la actual. Muy útil durante una emergencia donde los valores del momento anterior son relevantes para saber la tendencia.

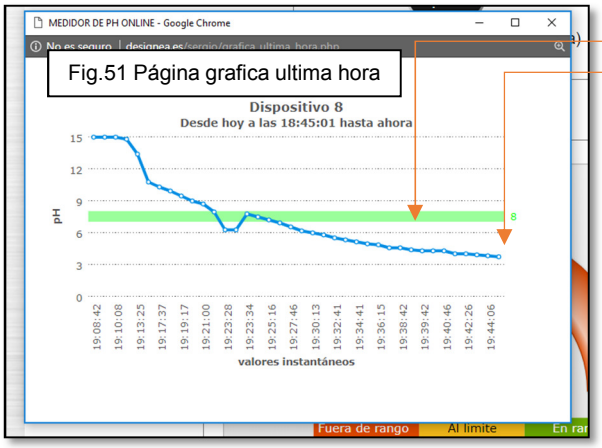


Fig.51 Página grafica ultima hora

Intervalo definido como válido
Valores que tenía en dispositivo.

En la gráfica de la figura 51 se observa como la tendencia del pH es bajista y por tanto no permisiva; se debe actuar de manera inmediata.

A continuación se detalla el código para obtener el resultado anterior.

```
/// consulta de configuración de este dispositivo:
$sql="SELECT * FROM dispositivos WHERE idDispositivo='$dispositivo'";
$res=mysqli_query($conexion,$sql);
$datos=mysqli_fetch_assoc($res); # registro único

$nombre=$datos['descripcion'];
$minimo_aceptable=$datos['minimoTolerable'];
$maximo_aceptable=$datos['maximoTolerable'];
$margen=($maximo_aceptable-$minimo_aceptable)*0.1; //10%
$minimo_plus=$minimo_aceptable+$margen; // 10% antes de llegar para color amarillo
$maximo_plus=$maximo_aceptable-$margen; // 10% antes de llegar para color amarillo
$sql="SELECT fechaHora as fecha ,valor FROM `mediciones` WHERE idDispositivo='$dispositivo' AND
fechaHora>='$desde' AND fechaHora<='$hasta'";
…
//definidas la variable $desde y $hasta obtenemos las distintas gráficas para posteriormente guardarlos en las
variables que se usaran para los datos de la gráfica.
/// cadena de datos para graficas :
$categoria=""; // una cadena con fechas para grafico de medias diarias
$data=""; //una cadena con valores para gráfico de medias

while($datos=mysqli_fetch_assoc($res))
{
            $categorias.="{'label' : '".substr($datos['fecha'],11,8)."'},";
            $data.="{'value' : '".$datos['valor']."'},";
}
//Creación de la variable para la etiqueta de la gráfica
if(substr($desde,0,10)==substr($desde,0,10))
{
            $subCaption="Desde hoy a las ".substr($desde,11,8)." hasta ahora";
}
else
```

```
{
        $subCaption="Desde ayer a las ".substr($desde,11,8)." hasta ahora";
}
```

Para la creación de ésta y el resto de gráficas se ha seguido el mismo procedimiento:

```
<script>
FusionCharts.ready(function(){
   var salesChart = new FusionCharts({
      type: 'scrollline2d',
      dataFormat: 'json',
      renderAt: 'chart-container',
      width: '100%',
      height: '50%',
      dataSource: {
         'chart': {
            'caption': 'Dispositivo <?=$dispositivo?>',
            'subCaption': '<?=$subCaption?>',
            'xAxisName': 'valores instantáneos',
            'yAxisName': 'pH',
            'yAxisMaxValue': '14',
            'showValues': '0',
            'numberPrefix': '',
            'showBorder': '0',
            'showShadow': '0',
            'bgColor': '#ffffff',
            'paletteColors': '#008ee4',
            'showCanvasBorder': '0',
            'showAxisLines': '0',
            'showAlternateHGridColor': '0',
            'divlineAlpha': '100',
            'divlineThickness': '1',
            'divLineIsDashed': '1',
            'divLineDashLen': '1',
            'divLineGapLen': '1',
            'lineThickness': '3',
            'flatScrollBars': '1',
            'scrollheight': '10',
            'numVisiblePlot': '100',
            'showHoverEffect':'1',
                                        'drawAnchors' : '1',
                                        'anchorRadius' : '2',
                                        'canvasBgColor' :'#FFFFFF',
                                        'canvasBgAlpha' :'30'
         },
         'categories': [
            {
               'category': [<?=$categorias?>]
            }
         ],
         'dataset': [
            {
               'data': [<?=$data?>]
            }
         ],
      'trendlines': [{
         'line': [{
            'isTrendZone': '1',
            'startvalue': '<?=$minimo_aceptable?>',
            'endValue': '<?=$maximo_aceptable?>',
            'color': '#00FF00',
            'valueOnRight': '1',
            'alpha': '40'
         }]

      }]
      }
   }).render(div1);
});
</script>
```

Gráfica periodo seleccionado.

Archivo físico: “grafica_valores_periodo.php”.
Al igual que la anterior gráfica muestra los datos pero esta vez en el periodo seleccionado en la página “Pelayo_principal.php”; más en concreto mostraría los valores que ha tenido el dispositivo y que se habían plasmado en el “abanico” azul de la gráfica.

El periodo seleccionado en la figura 52 ha sido los últimos 30 minutos.

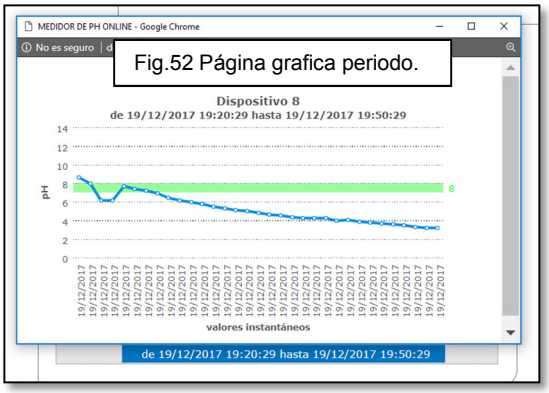


Fig.52 Página grafica periodo.

Gráfica valores medios horarios.

Archivo físico: “grafica_valores_medios.php”.

Este tipo de gráficos es mejor interpretarlo post-emergencia, tal y como se veía en la gráfica anterior la tendencia era bajista y por tanto no permisiva, sin embargo tal y como se ve en la gráfica (durante la emergencia) el dato de las 19h es superior (caso que puede dar confusión), debido a la existencia en las primeros minutos de las 19 de datos con valores muy altos.

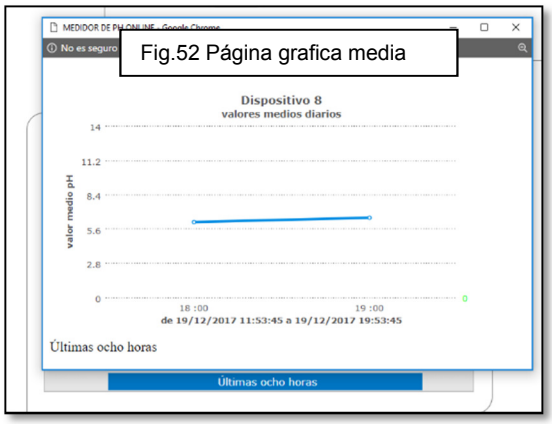


Fig.52 Página grafica media

Gráfica excursión de pH.

Archivo físico: “grafica_excursion_ph.php”.
Esta gráfica nos indica los valores máximos y mínimos obtenidos en el periodo seleccionado en la pantalla de control principal de valores. Para interpretar los valores tenemos que fijarnos en la parte de color (azul), ésta zona será en la que se han movido los valores del dispositivo en el tiempo.
Los datos de la gráfica ejemplo son erráticos, vemos como oscilan entre 14 y 0 con mucha facilidad en los días seleccionados (en este caso ha oscilado por pruebas con el mismo).
La consulta para obtenerla es un poco más compleja que en los casos anteriores, no obstante siguen el siguiente patrón según quiera verlos por minutos, horas o días (a continuación para minutos):

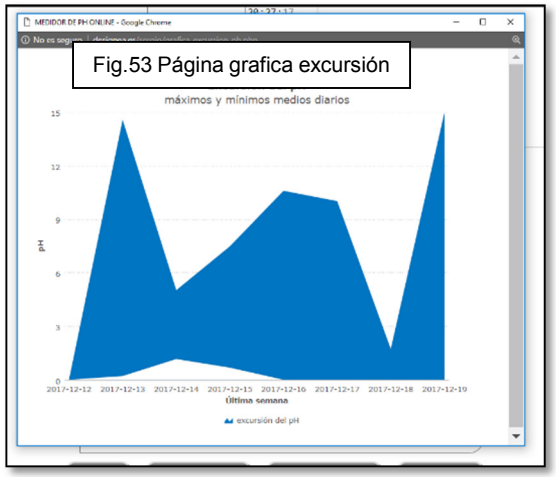


Fig.53 Página grafica excursión

```
…
//Con esta consulta calculo los minimos agrupados por fechas (incluidos minutos), de modo que esta misma consulta cambiando MIN por MAX obtengo los máximos.
$sql_por_minutos="SELECT CONCAT(DATE(fechaHora),' / ',HOUR(fechaHora),':',MINUTE(fechaHora),' h.') as momento,MIN(valor) as valores FROM `mediciones` WHERE idDispositivo='$dispositivo' AND fechaHora>='$desde' AND fechaHora<='$hasta' GROUP BY DATE(fechaHora),HOUR(fechaHora),TIME(fechaHora) ORDER BY `fechaHora` ASC";
…
…
//se guardan los valores en variables del momento se hace dos veces…(uno para mínimos y otro para los máximos)…
while($datos=mysqli_fetch_assoc($res))
{
        $fechas.="{'label' : '".$datos['momento']."'},";
```

```
		$minimos.="{'value' : '".$datos['valores']."'},";
}
…
//Con los datos en las variables mínimos, máximos y fechas pintamos la gráfica.
```

Página calendario de mediciones.

Archivo físico: “Pelayo_concentracion.php”.
Este apartado está disponible desde la página de control principal de valores del dispositivo mediante un icono tipo calendario. Desde aquí se pueden ver los días que se ha utilizado y qué valores se han obtenido por días.

Fig.54 Página Calendario

Si pulsamos sobre cada día, se abre la gráfica del día en concreto.

Para la generación del calendario no se ha usado ningún complemento ni objeto externo; se ha creado mediante código usando tablas y variables ($mes,$dia y $ano). Se calculan los días de cada mes de cada año seleccionado. Anteriormente se guardan en un vector de elementos los datos de mínimos y máximos, para poder seleccionar un color u otro a medida que vamos recorriendo los días de cada mes.

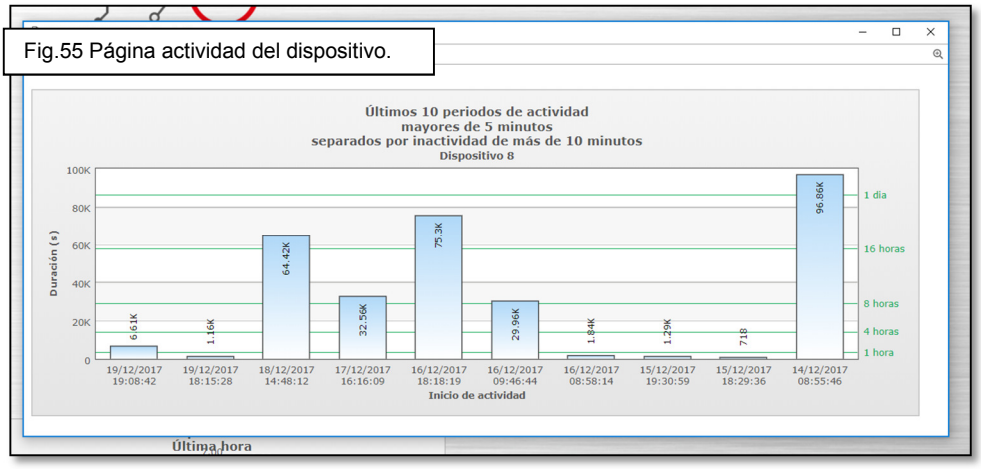
Ejemplo para los mínimos:

```
//minimos: construción de array con una sola consulta:
$sql="SELECT DATE(fechaHora) as fecha ,min(valor) as minimo FROM `mediciones`
WHERE idDispositivo='$dispositivo'
GROUP BY DATE(fechaHora)";
$res=mysqli_query($conexion,$sql);
$minimos=array(); // plural, asociativo [fecha]=valor
while($datos=mysqli_fetch_assoc($res))
{
		$minimos[$datos['fecha']]=$datos['minimo'];
}
```

Página actividad.

Archivo físico: “grafica_periodos_actividad.php”.
La página actividad se ha creado para ver los periodos de actividad del dispositivo, no sólo en valores sino en segundos de actividad. Viendo la gráfica generada se puede ver de un solo vistazo el tiempo de funcionamiento del dispositivo. Accesible desde el mismo lugar que el propio calendario en la página principal de valores.

Fig.55 Página actividad del dispositivo.



El periodo de la gráfica al igual que en las anteriores es el periodo seleccionado en la página de valores.
La creación de la gráfica en sí sigue el mismo formato que las anteriores gráficas (cambiando el tipo), no obstante la obtención de los periodos de actividad es algo más complejo que los casos anteriores:

```
$sql="SELECT fechaHora FROM mediciones WHERE idDispositivo='$dispositivo' ORDER BY fechaHora DESC";
$res=mysqli_query($conexion,$sql);

$limite_periodos=10; //Total periodos máximos a mostrar en la gráfica.
$hueco=600; // segundos a considerar como inactividad
$duracion_minima=300; //Tiempo para considerar otro periodo de actividad (5 minutos)

$res=mysqli_query($conexion,$sql);
$fh=array(); // cada fechaHora
$inicio=array();
$duracion=array();
$datos=mysqli_fetch_assoc($res);  # muchos datos
$fin[0]=$datos['fechaHora']; #cojo el primero que es el "hasta" con seguridad
$fh[0]=$fin[0]; // lo meto como primer valor del array.
$i=1; // siguiente
$p=0;
while($datos=mysqli_fetch_assoc($res)) #el resto de datos
{
        $fh[$i]=$datos['fechaHora'];
        // calculo el tiempo como la diferencia con el anterior:
        $t= strtotime($fh[($i-1)]) - strtotime($fh[$i]); // anterior-actual
        if($t>$hueco)  // hay hueco de mas de 5 minutos: cojo el anterior y listo.
        {
                $inicio[($p)]=$fh[($i-1)];
                $p++;
                $fin[$p]=$fh[$i];
                $d=strtotime($fin[($p-1)]) - strtotime($inicio[$p-1]); // duración
                $duracion[$p-1]=$d;
                if($d<$duracion_minima)
                {
                        $p--; //retrotraigo el índice del periodo
                        $fin[$p]=$fh[$i];
                }
        }
        $i++;
        if($p>=$limite_periodos){break;}
}
$n_periodos=count($inicio)-1;
for ($p=0;$p<=$n_periodos;$p++)  //Orden inverso
{
        $category.="{ 'label': '".espanola($inicio[$p])."' },";
        $data.="{ 'value': '".$duracion[$p]."' },";
}
//Llegados a este punto ya tengo en las variables $category y $data los datos necesarios para la gráfica.
```

Pruebas de funcionamiento.

Las pruebas se van a dividir en varios sectores de control:

1) Persistencia: tiempo máximo de funcionamiento, control de errores y recuperaciones del sistema.
2) Consumo: cantidad de datos usado.
3) Fiabilidad: comprobación de datos con equipos homologados y posicionamiento.
4) Tiempos de muestreo.

Persistencia

La herramienta de periodos de actividad creada en la fase de creación de la plataforma nos va a servir de utilidad para verificar este campo. Para ello se ha completado una carga completa de la batería y se ha dejado en funcionamiento por largos periodos de trabajo, en la gráfica de la figura 56 comenzaron las pruebas el día 22/12/2017, dejándose éste en funcionamiento por tiempo superior a 1 día (90.520 sg); seguidamente se dejó el día 24/12 por tiempo similar al anterior (91.210sg) y los días 25 y 26 con tiempo de 47.650sg y 46.350sg respectivamente.

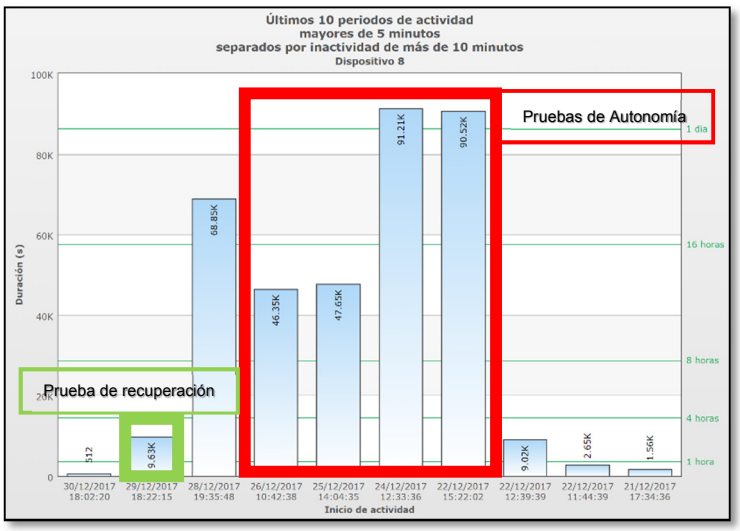


Fig. 56. Gráfica de actividad pruebas.

Sumados todos los intervalos hacen una suma de 275.730 sg (76.59h = 3,19días). Durante ese tiempo no ha habido fallo crítico salvo el apagado del dispositivo al consumir la batería; este hecho ha sido posible gracias a la parte de la programación del sketch dedicada a la recuperación del sistema por lo que ante eventuales fallos de la operadora es capaz de recuperarse.
Dicha comprobación (recuperación ante cuelgues-desconexiones de la operadora) se hizo el día 29/12 desconectando el sistema de recuperación y obteniendo un tiempo de trabajo de 9630 sg (2,67h) dejando por tanto en buen lugar al sistema de recuperación.

Consumo

Durante el periodo de pruebas se ha estado monitorizando el consumo de datos del dispositivo, de modo que podemos medir con exactitud la cantidad de kilobytes consumidos por unidad de tiempo.
Como se puede apreciar en la figura 57 (obtenida de la plataforma cliente de la operadora) el consumo máximo diario obtenido ha sido de 4,69MB; teniendo en cuenta que ese día fue uno de pruebas y sabemos que estuvo 91210 segundos activo obtenemos un consumo de $\frac{4{,}69x1024}{91210} = \mathbf{0.05}\ \boldsymbol{Kbytes/sg}$ (éste dato no ha tenido en cuenta aquellos periodos menores de 5 minutos en ese día). Cualquier plan de datos actual cubre las necesidades del dispositivo.

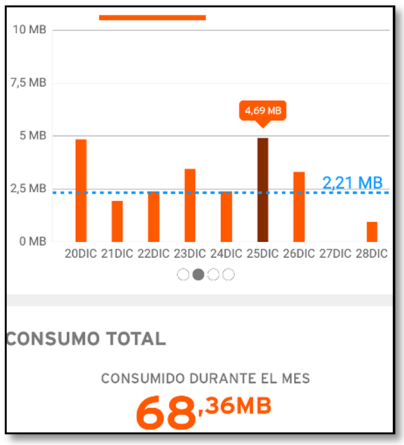
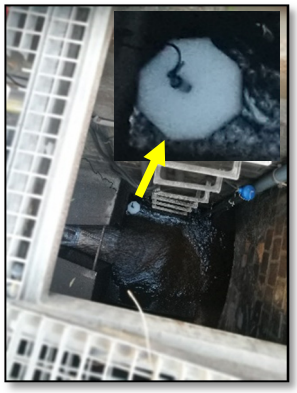


Fig. 57. Gráfica de consumo de datos.

Fiabilidad

Para probar la fiabilidad del sistema he desplazado el dispositivo al mismo lugar de medición de uno de los medidores fijos de la factoría; éste posee un software que monitoriza los datos recogidos. Se he agregado un soporte a la sonda para que flote en el medio líquido.

Fig. 58. Pruebas reales.



Se ha medido por tiempo entre las 17:35h y las 18:00h del día 21/12 y se han obtenido las gráficas de la figura 59.

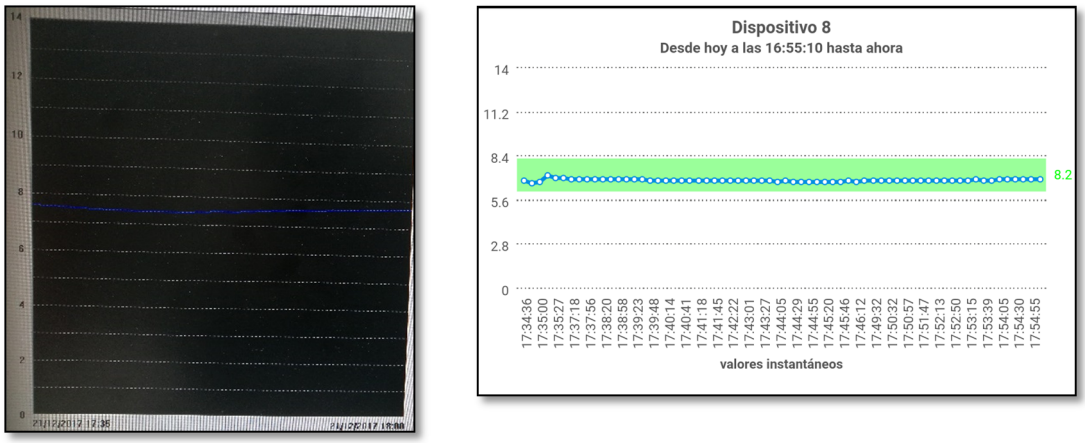


Fig. 59. Comparativa de resultados.

El resultado es apto y se da por buena la prueba.

Para las pruebas de posicionamiento se ha encendido el dispositivo en un lugar conocido y se ha comprobado en el mismo Google Maps la precisión obtenida.

Fig. 60. Prueba de posicionamiento.

La referencia de la prueba era la intersección de la nave y el centro de la calle (marca en suelo en la figura 60) obteniendo un resultado de desviación respecto a la marca de 1,5 metros aproximadamente.

Tiempos de muestreo.

Cara a cumplir con el objetivo inicial de registrar un dato cada 1-2 minutos se observan los registros dados de alta en las pruebas realizadas.

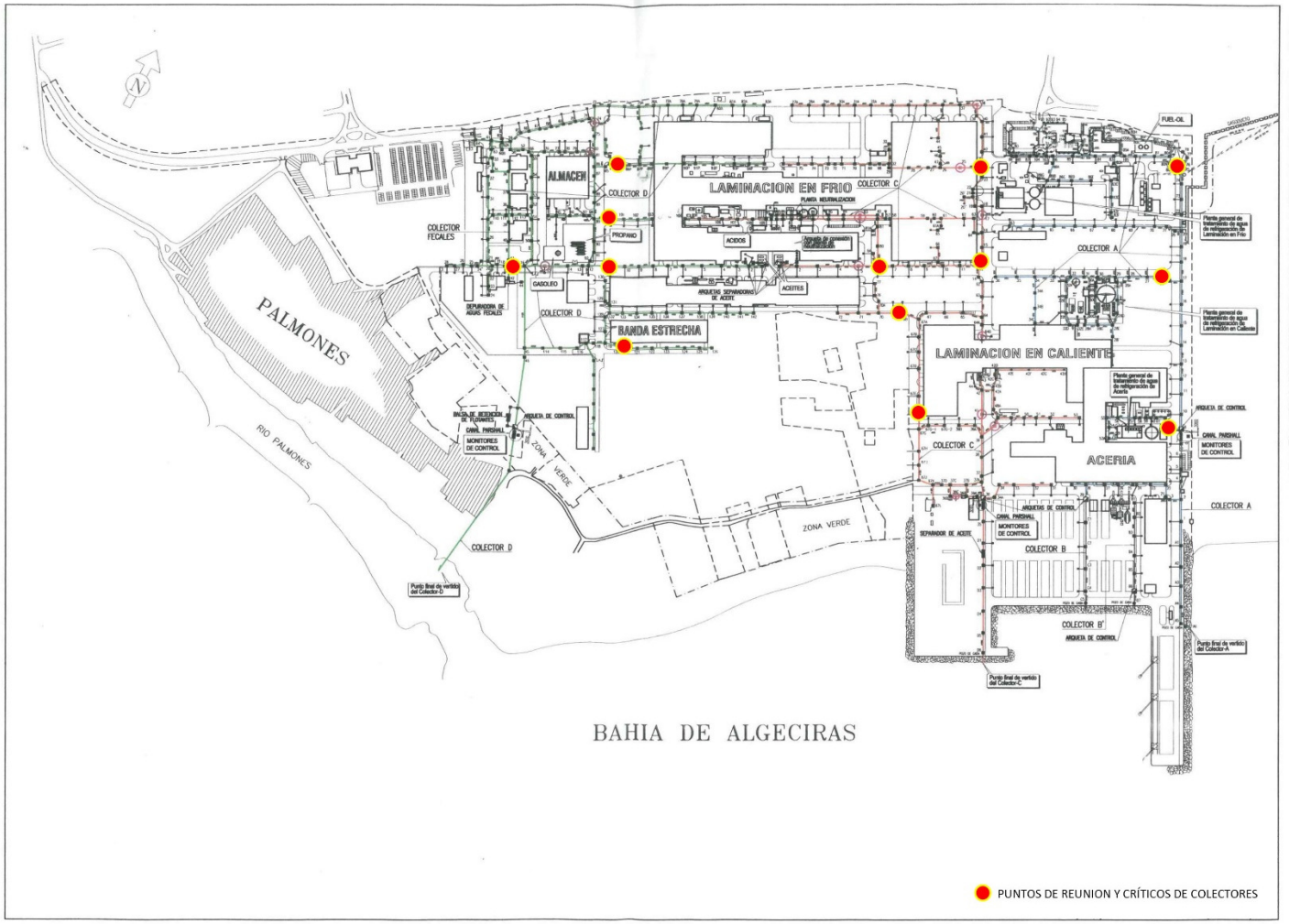
idMedida	idDispositivo	valor	fechaHora	latitud	longitud
78835	8	7.13	2017-12-30 21:16:59	36.179565	-5.435532
78834	8	7.12	2017-12-30 21:16:32	36.179565	-5.435532
78833	8	7.12	2017-12-30 21:16:18	36.179565	-5.435532
78832	8	7.11	2017-12-30 21:16:04	36.179565	-5.435532
78831	8	7.11	2017-12-30 21:15:51	36.179565	-5.435532
78830	8	7.10	2017-12-30 21:15:37	36.179565	-5.435532
78829	8	7.18	2017-12-30 18:10:52	36.179599	-5.435582
78828	8	7.18	2017-12-30 18:10:38	36.179599	-5.435582
78827	8	7.18	2017-12-30 18:10:25	36.179599	-5.435582
78826	8	7.18	2017-12-30 18:10:11	36.179599	-5.435582
78825	8	7.17	2017-12-30 18:09:58	36.179599	-5.435582
78824	8	7.17	2017-12-30 18:09:31	36.179599	-5.435582
78823	8	7.17	2017-12-30 18:09:18	36.179599	-5.435582
78822	8	7.17	2017-12-30 18:09:05	36.179599	-5.435582
78821	8	7.17	2017-12-30 18:08:51	36.179599	-5.435582

Fig. 61.Registro datos en el servidor

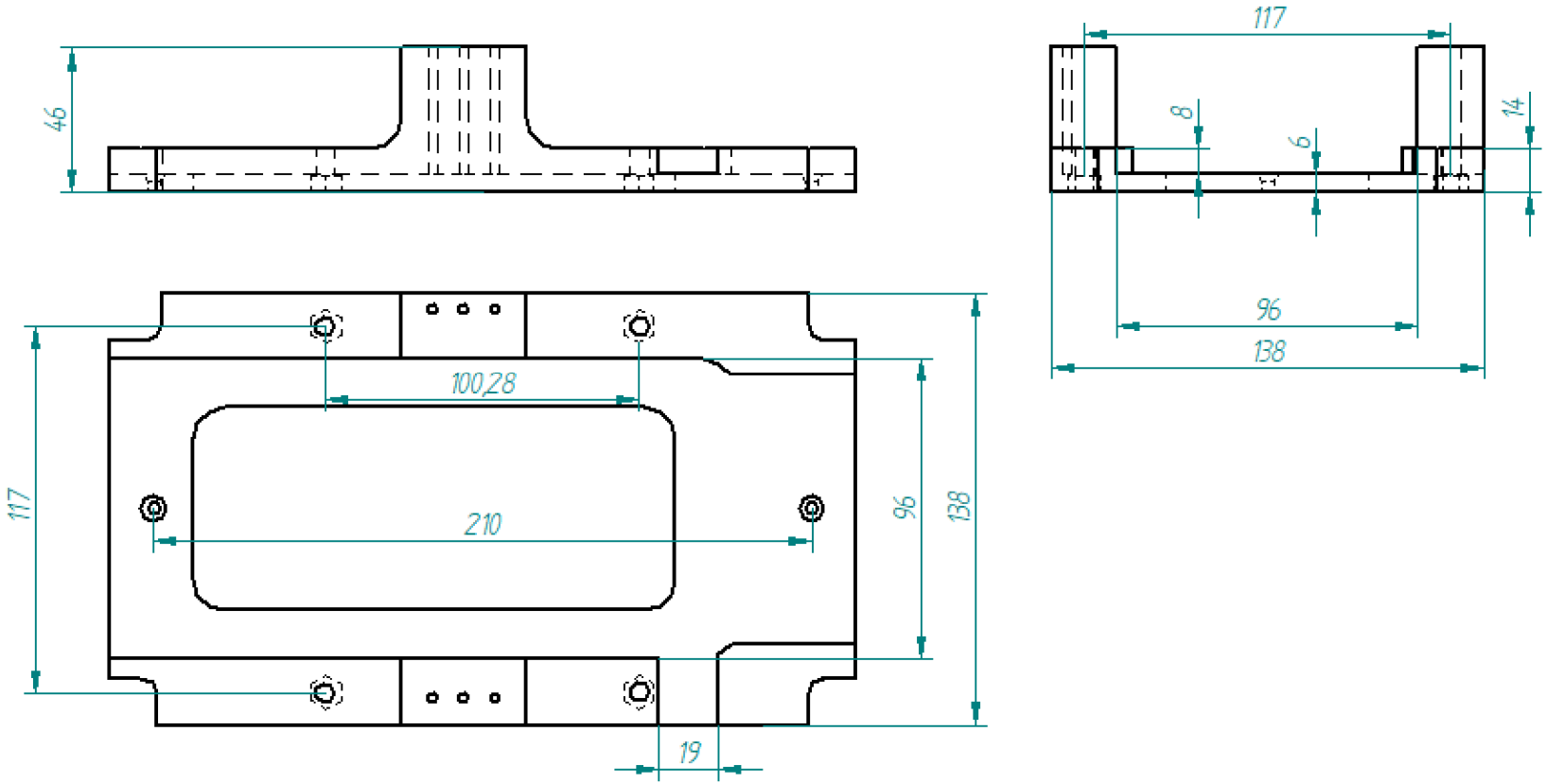
Si nos fijamos en los datos de sesión marcados en la figura 61 podemos observar como el tiempo de separación entre muestras es inferior a 1 minuto y por tanto cumple con los objetivos de tiempos de muestreo.

ANEXOS.

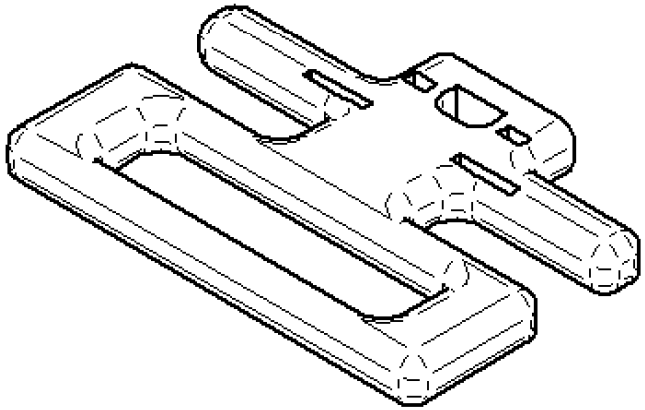
Plano de puntos de reunión críticos.

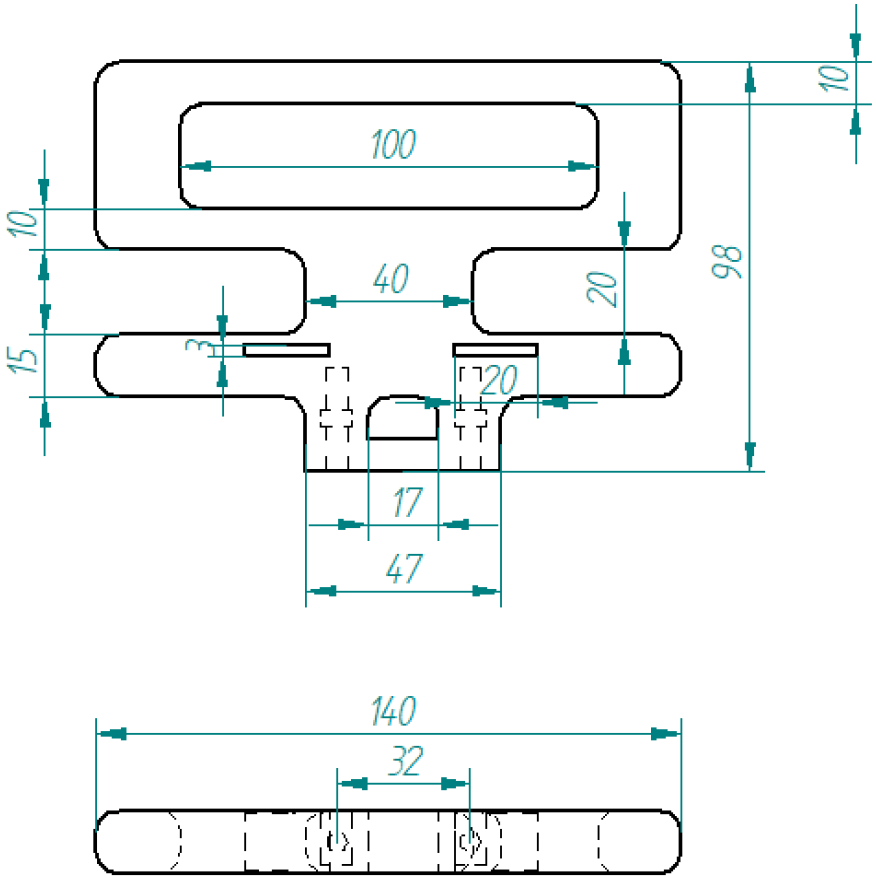


Plano soporte principal de la carcasa.

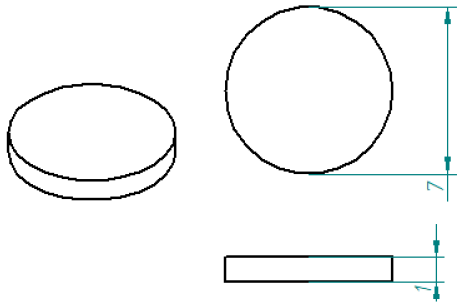


Porta cables / Asa.



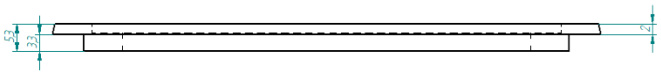


Protector Led.

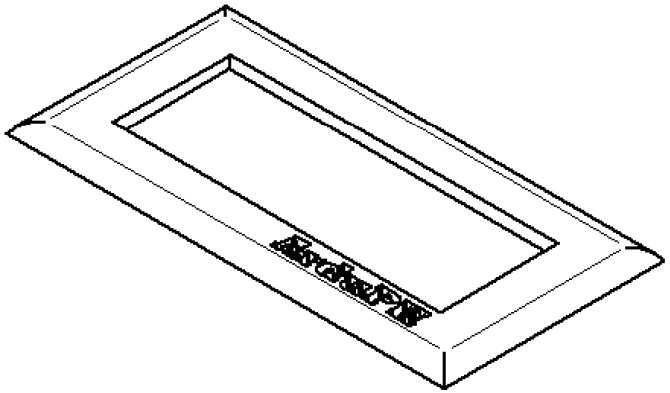


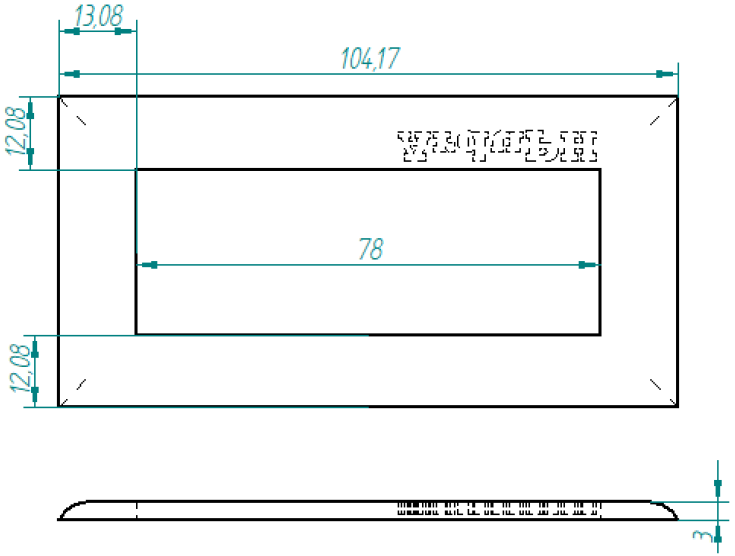
Embellecedor LCD

Parte Baja

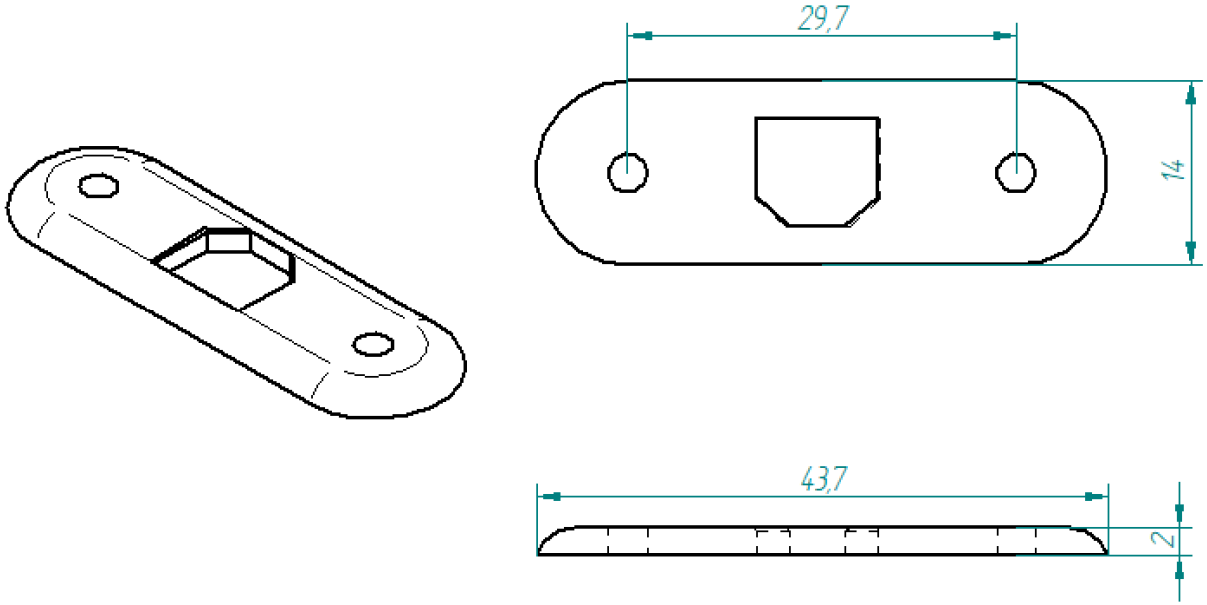


Parte Alta

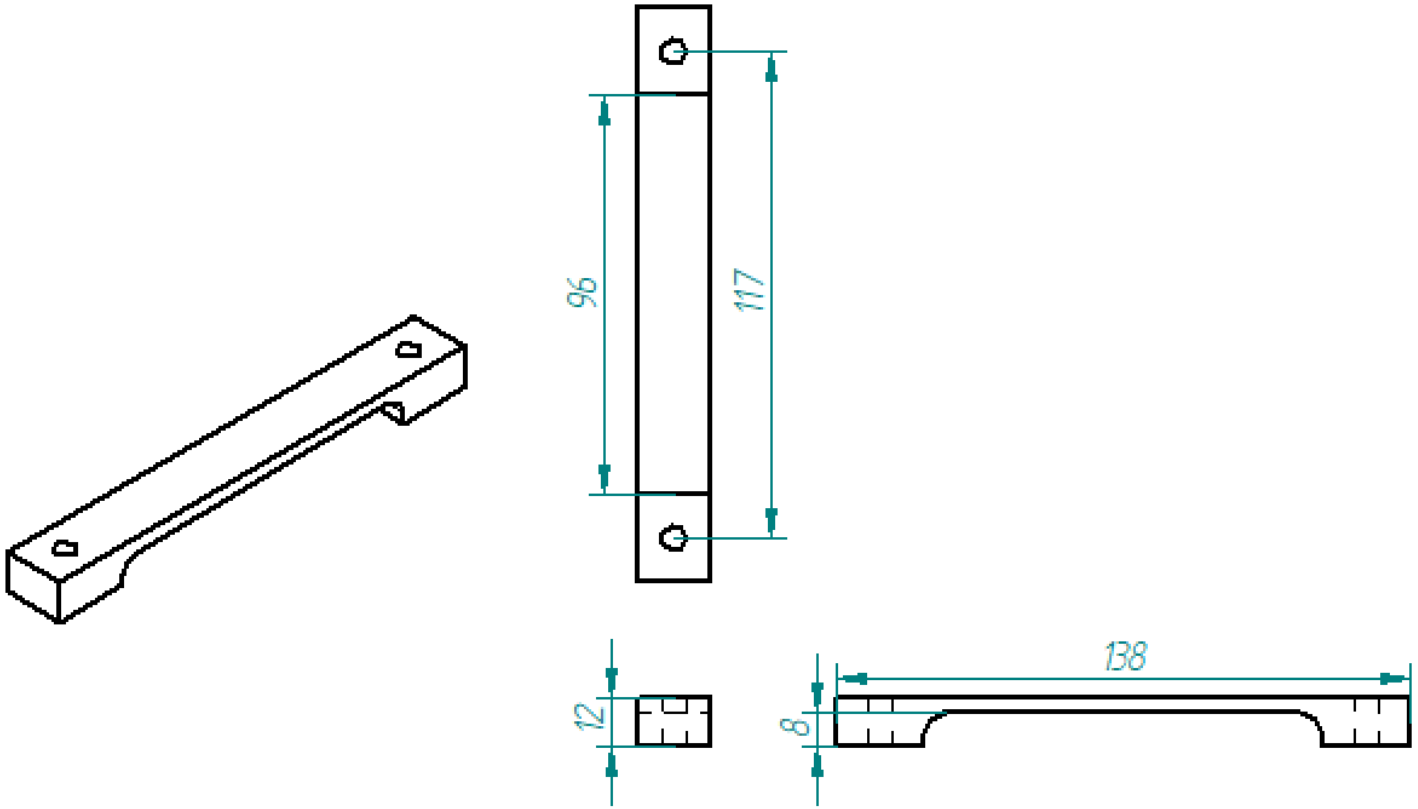




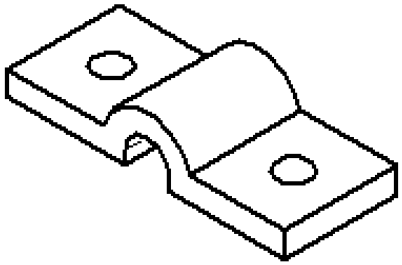
Embellecedor conector USB-B

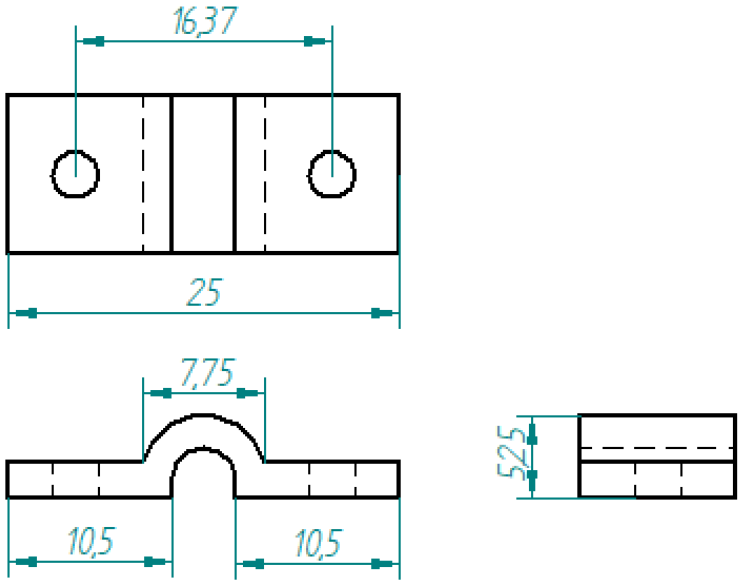


Presilla batería.

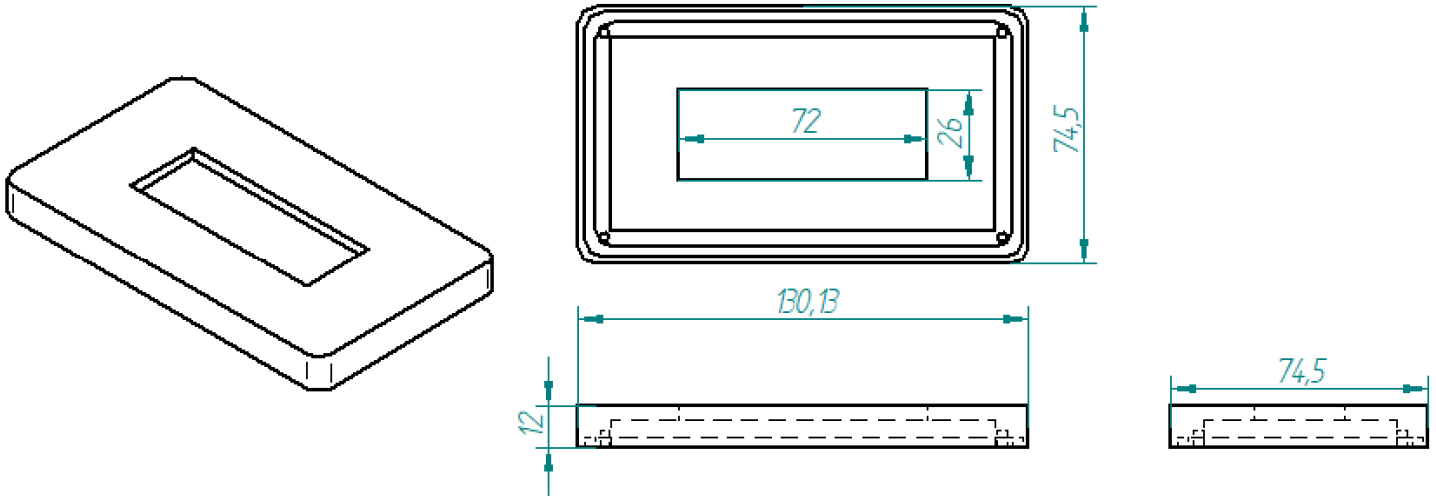


Prisionero Cable.



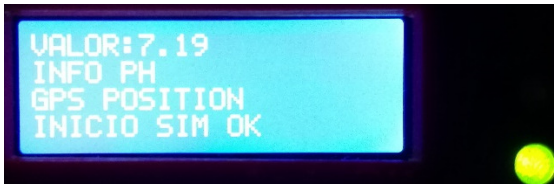


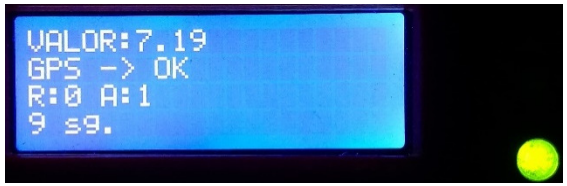
Visor para Batería.



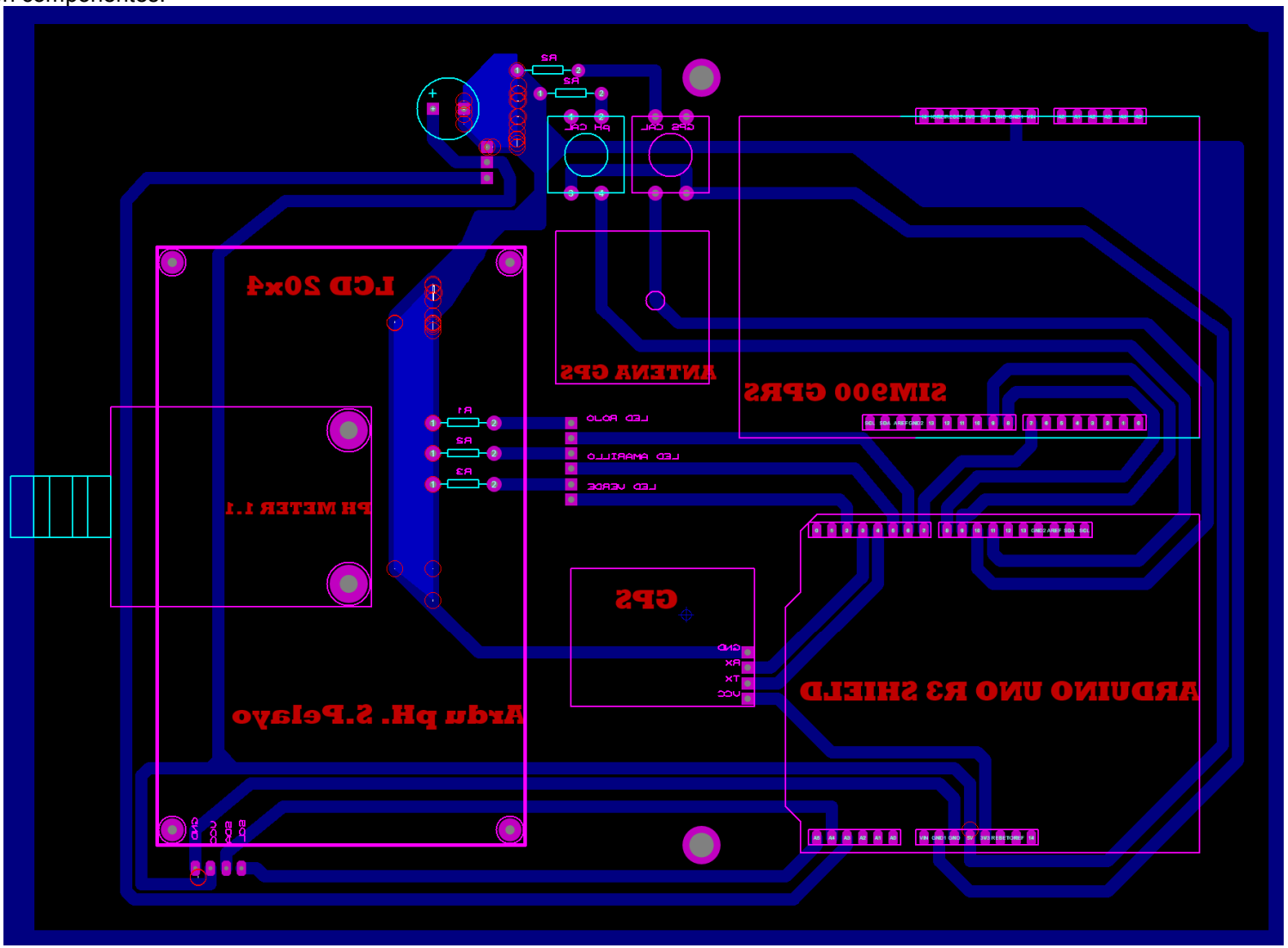
Vista general y detalles del dispositivo.

Secuencia de funcionamiento del dispositivo.

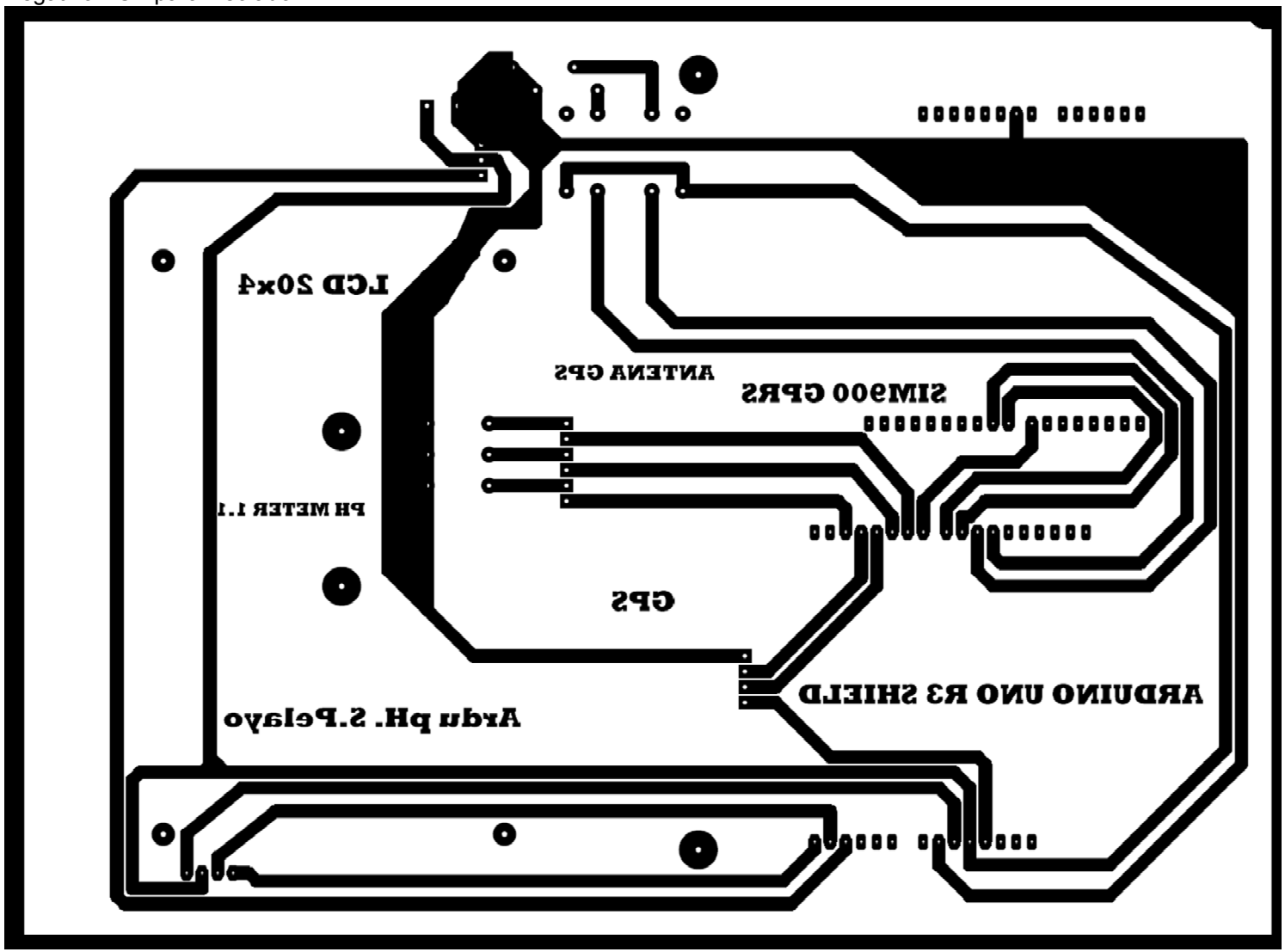




Circuito PCB con componentes.



Negativo PCB para insolado.



Bibliografía

S/N. (2016). Arduino Avanzado. -, de - Sitio web: https://aprendiendoarduino.wordpress.com/cursos/curso-arduino-avanzado-2016/

Wikipedia. (2010). Comandos Hayes. -, de Wikipedia Sitio web: https://es.wikipedia.org/wiki/Conjunto_de_comandos_Hayes

Prometec. (2017). Comandos AT para GSM/GPRS. -, de Prometec Sitio web: https://www.prometec.net/comandos-at-gsm-gprs-gps/

Mikal Hart. (2013). A Compact Arduino GPS/NMEA Parser. -, de - Sitio web: http://arduiniana.org/libraries/tinygps/

Glenn Baddeley. (2001). GPS - NMEA sentence information. -, de - Sitio web: http://aprs.gids.nl/nmea/#rmc

Prometec. (2017). El Bus I2C. -, de - Sitio web: https://www.prometec.net/bus-i2c/

PunBB. (2014). Curso / Tutorial de Solid Edge en español. -, de - Sitio web: https://www.arquiparados.com/t497-curso-tutorial-de-solid-edge-en-espanol-aprende-desde-cero

Google. (2017). Google maps API. -, de - Sitio web: https://developers.google.com/maps/documentation/javascript/adding-a-google-map

Aprende a programar. (S/N). Tutorial básico del programador web: Ajax desde cero. -, de - Sitio web: https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=912:informacion-basica-curso-qtutorial-basico-del-programador-web-ajax-desde-ceroq-cu01201f&catid=83&Itemid=212