El propósito principal de este proyecto es la implementación de producto de medición de ruido acústico y variables ambientales. A lo largo del documento se muestra el análisis, diseño e implementación de los requisitos del cliente: la empresa Heimdal Sound Control. El producto en conjunto incluye un dispositivo electrónico con funcionalidad de medición y conexión y un sistema de gestión basado en tecnologías web alojado en Internet capaz de tratar las mediciones realizadas por diferentes dispositivos, así como de gestionar las entidades implicadas en el sistema en base a diferentes roles y permisos.

Pablo Garrido Sánchez nació en Cádiz, España, en mayo de 1992. Con este trabajo finaliza el Máster en Ingeniería de Telecomunicación de la Universidad de Granada tras haber terminado satisfactoriamente el Grado de Ingeniería de las Tecnologías de Telecomunicación con mención en Telemática en junio de 2014 en la misma universidad.

Andrés María Roldán Aranda es el profesor ingeniero a cargo del presente proyecto, así como el tutor del alumno. Actualmente es profesor del departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

TRABAJO FIN DE MÁSTER

Desarrollo de un sistema de medición de ruido acústico

Pablo Garrido Sánchez

INGENIERÍA DE TELECOMUNICACIÓN

2015/16

UNIVERSIDAD DE GRANADA

Máster en Ingeniería de Telecomunicación

TRABAJO FIN DE MÁSTER

Desarrollo de un sistema de medición de ruido acústico

Pablo Garrido Sánchez

Año académico 2015/2016

Tutor: Andrés María Roldán Aranda

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

"Desarrollo de un sistema de medición de ruido acústico y variables ambientales"

CURSO: 2014/2015

Pablo Garrido Sánchez

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

"Desarrollo de un sistema de medición de ruido acústico y variables ambientales"

REALIZADO POR:

Pablo Garrido Sánchez

DIRIGIDO POR:

Andrés María Roldán Aranda

DEPARTAMENTO:

Electrónica y Tecnología de los Computadores

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Máster de D. Pablo Garrido Sánchez,

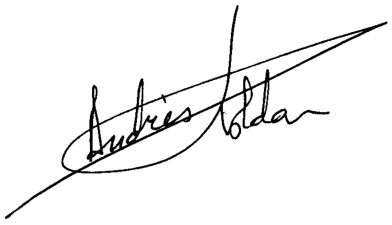
Informa:

que el presente trabajo, titulado:

"Desarrollo de un sistema de medición de ruido acústico y variables ambientales"

ha sido realizado y redactado por el mencionado alumno bajo nuestra dirección, y con esta fecha autorizo a su presentación.

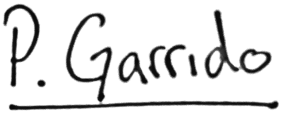
Granada, a 15 de julio de 2016

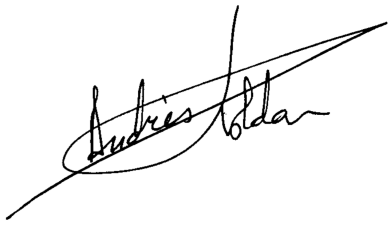


Fdo. Andrés María Roldán Aranda

Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Máster se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 15 de julio de 2016





Fdo. Pablo Garrido Sánchez

Fdo. Andrés María Roldán Aranda

Desarrollo de un sistema de medición de ruido acústico y variables ambientales

Pablo Garrido Sánchez

PALABRAS CLAVE:

Ruido acústico, Variables ambientales, Sensor, Internet of Things, Diseño de PCB, Electrónica, Altium, Raspberry Pi, Full Stack JavaScript, Node.js, Python, JavaScript.

RESUMEN:

El propósito principal del presente proyecto es desarrollar un producto completo capaz de realizar mediciones ruido acústico y de variables ambientales, así como de transmitirlas hacia un sistema de gestión de datos con interfaz de usuario desde donde se puedan consultar dicha información y gestionar el sistema.

La metodología de desarrollo propuesta está enfocada a producto; de esta manera, tras la introducción del documento, se presentan los requisitos del cliente que ha contratado los servicios de desarrollo para después aplicar un proceso de análisis, diseño e implementación basado en las directrices de la ingeniería de producto.

El trabajo realizado se presenta como Trabajo Fin de Máster de la titulación de Máster en Ingeniería de Telecomunicación de la Universidad de Granada, por tanto se considera interesante otorgarle al trabajo un enfoque multidisciplinar en el que se utilicen, dentro de lo posible, todos los conocimientos adquiridos durante la formación.

Bajo esta última consideración, se pretende que las tecnologías utilizadas sean nuevas y emergentes para aportar valor al factor académico del trabajo. Este hecho hace que a lo largo del trabajo se presenten sistemas construidos a partir de programas de diseño electrónico utilizados en la industria; se apliquen técnicas avanzadas de prototipado, como son la utilización de placas electrónicas metalizadas, de doble cara y fresadas mediante control numérico o la utilización de dispositivos SMT; por otra parte el sistema de gestión de los datos recopilados que se presenta hace uso de novedosas tecnologías web dado el reciente crecimiento de las posibilidades y la popularidad de las mismas, de esta manera se obtendrá un sistema de gestión seguro y acorde a los modelos actuales de programación y una interfaz multiusuario y multiplatataforma con fuerte orientación a la experiencia de uso.

El sistema completo desarrollado, tal y como se explica a lo largo del documento, cumple con los requisitos y necesidades del cliente, implementa un producto funcional en una fase de prototipo avanzado y supone un reto multidisplinar para el alumno.

Development of an acoustic noise and environmental variables measurement system

Pablo Garrido Sánchez

KEYWORDS:

Acoustic noise, Enviromental parameters, Sensors, Internet of Things, PCB desing, Electronics, Altium, Raspberry Pi, Full Stack JavaScript, Node.js, Python, JavaScript.

ABSTRACT:

The main purpose of this project is the development of a product that can measure acoustic noise and environmental variables which it is able to transmit this information to a data management system with an user interface where the measured information can be visualized and the system can be managed.

The methodology proposed for this development is focused on product. Thus, following the introduction of the document, the requirements of the customer who has engaged the services of development are exposed. Afterwards, a process of analysis, design and implementation based on product engineering's guidelines is applied.

The work is presented as a Master Degree Project for the degree of Master in Telecommunications Engineering of the University of Granada, therefore it is interesting to give the work a multidisciplinary approach in which, as far as posible, all the competences acquired during the formation are used.

Following the latter consideration, it is intended that the development uses cutting-edge technologies in order to enhance the academic value of the project. Due to this fact, during the development of the product, industrial electronic computer aid desing softwares were used; advanced prototyping techniques were applied, such as the use of electronic milling double sided boards by Computer Numeric Control or the use of Surface Mount Technology devices during the design process. Furthermore, the management system presented makes use of innovative web technologies, who's possibilities and popularity have recently grown, in such a way that it has been designed according to an up-to-date programming paradigm. Using the same paradigm, a multi-user and multi-platataform user interface with strong orientation to the experience of use have been developed.

The system developed, as explained throughout the document meets the customer's requirements, implement a functional product in an advanced prototype stage and it is a multi-disciplinary challenge for the student.

A la memoria de Pablo Garrido

Agradecimientos:

En primer lugar, me gustaría agradecer a mis padres Pilar y Antonio su apoyo durante mi periodo universitario, y en general, durante toda mi vida. Hubiese sido imposible conseguir lo que he conseguido hasta ahora sin su incondicional apoyo y compañía en cada momento. Con este proyecto, cierro mi etapa universitaria, pero abro otra en la que seguiréis siendo los principales protagonistas.

También quiero agradecer a mis compañeros y amigos de la facultad, quiénes me han ayudado en cada momento que lo he necesitado en mis años universitarios.

Por último, agradezco a mi tutor, Andrés María Roldán Aranda su tiempo dedicado y comentarios instructivos, no sólo para este proyecto, sino para mi futura vida profesional.

Acknowledgments:

Firstly, I would like to thank my parents Pilar y Antonio for their support during my university period and, in general, my whole life. It would have been impossible to achieve what I achieved without their unconditional support and company in every moment. With this project, I close my university stage, opening up a new one in which you will also be main characters.

I want to thank to my colleagues and friends too, who help me in every moment that I need in my university years.

Finally, I would like to thank to my advisor, Andrés María Roldán Aranda for his provided timely and instructive comments, not only for this project, but for my future professional life.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE VIDEOS

ÍNDICE DE TABLAS

GLOSARIO

Altium Paquete *software* EDA capaz de asistir el diseño de esquemáticos electronicos y PCB, así como la simulación de sistemas electrónicos.

API HTTP REST Siglas de Application Programming Interface, HiperText Tranfer Procotol y REpresentational State Transfer. Este concepto hace referencia a un tipo de interfaz web hacia el lado del cliente basada en HTTP que utiliza los verbos comunes del protocolo HTTP (GET, POST, DELETE...) para acceder y gestionar datos e información existente en el lado del servidor.

Bit-banging Técnica de comunicación digital que utiliza salidas digitales de propósito general para simular por *software* lo que haría un módulo *hardware* dedicado a dicha comunicación.

Daemon Se conoce como servicio o proceso *daemon* a aquellos procesos que se ejecutan en segundo plano y no tienen interacción con el usuario dentro del conjunto de procesos de un sistema operativo. En programación multihebrada en Python, se conoce como hilo *daemon* a aquellos hilos secundarios que se ejecutan en segundo plano y que terminan junto con el hilo de ejecución principal, sin necesidad de tener que gestionar su estado desde el mismo.

formato PEM Privacy-enhanced Electronic Mail es un formato abierto para el almacenado de claves RSA. Utiliza codificación Base64 y está mantenido por la IETF.

Full Stack JavaScript Tambien conocido como MEAN Stack, es un nuevo paradigma de programación web que está sustituyendo al famoso LAMP. Se basa en el uso de

JavaScript y JSON desde la interfaz de usuario hasta el servidor y la base de datos. Pretende unificar el lenguaje de programación web así como la notación de objetos.

HTTP Websockets Tecnología que permite generar una comunicación *full-duplex* sobre el protocolo de aplicación HTTP similar a la utilizada por TCP en la capa de transporte de la pila OSI.

Modelo-Vista-Controlador Paradigma de arquitectura de *software* que separa la lógica del programa del conjunto de datos a manejar, así como de la representación de los mismos. Tal como indica el propio nombre, los tres pilares de eta arquitectura son el modelo de datos, el controlador o lógica de programa y la vista o interfaz de usuario.

Raspberry Pi HAT Nombre común de las extensiones *hardware* de la comunidad de la plataforma Raspberry Pi. Hace referencia a *Hardware Attached on Top* y juega con el doble sentido de la palabra *hat* o sombrero en inglés.

Responsive Web Design Filosofía de diseño web orientada a hacer diseños adaptables al dispositivo en el que se muestre la web en cuestión. Se tienen en cuenta tanto el tamaño, resolución y ratio de aspecto de la pantalla como el tipo de interfaz de interacción de usuario (ratón, táctil...).

RSA RSA (Rivest, Shamir y Adleman) hace referencia al primer y más común sistema criptográfico de clave pública. Mediante un par de claves, una pública y otra privada, es posible encriptar y firmar digitalmente información. Se basa en la idea de que todo lo que se cifre con una de las claves solo podrá ser descrifrado con la otra, y en que la clave privada únicamente la conocerá el dueño de las claves, mientras que la clave pública será accesible por cualquiera interesado.

Ruido acústico Se considera ruido acústico toda aquella potencia emitida en forma de onda acústica que no es deseada en un entorno concreto para un cierto grupo de receptores.

Sensor En el contexto de este proyecto, se considera un sensor a todo aquel dispositivo capaz de medir el estado de las Variables ambientales en un instante concreto.

Variables ambientales En el contexto de este proyecto, se consideran variables ambientales todas aquella características que describen el estado de un sistema ambiental, entendiendo por sistema ambiental un entorno urbano, rural, un local o un emplazamiento público. Son variables ambientales la temperatura, la humedad, la luminosidad o la velocidad del viento o el ruido acústico, entre otras.

ACRÓNIMOS

ACID Atomicity, Consistency, Isolation and Durability.

ADSL Asymmetric Digital Subscriber Line.

ARM Advanced RISC Machine.

CNC Computer Numeric Control.

CSS Cascade Style Sheet.

CSV Comma Separated Values.

EDA Electronic Desing Automation.

EMC ElectroMagnetic Compatibility.

FR4 Fire Retardant 4.

GNU GNU is Not Unix.

GPIO General Purpose Input Output.

GPRS General Packet Radio Service.

GPS Global Positioning System.

GUI Graphical User Inteface.

HFC Hybrid Fiber Coaxial.

HTML HyperText Markup Language.

HTTP HyperText Tranfer Protocol.

HTTPS HyperText Transfer Protocol Secure.

I2C Inter-Integrated Circuit.

IETF Internet Engineering Task Force.

IP Internet Protocol.

IR InfraRed.

JSON JavaScript Object Notation.

LAMP Linux + Apache + MySQL + PHP.

LCD Liquid Crystal Display.

LED Light-Emitting Diode.

MEAN MongoDB + Express + Angular.js + Node.js.

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor.

NoSQL Not Only SQL.

NTP Network Time Protocol.

OSI Open System Interconnection.

PCB Printed Circuit Board.

PKCS1 Public Key Cryptography Standard 1.

PWM Pulse Width Modulation.

RH Relative Humidity.

RISC Reduced Instruction Set Computer.

RTC Real Time Clock.

SBC Single Board Computer.

SHA-2 Secure Hash Algorithm 2.

SMA SubMiniature version A.

SMD Surface Mount Devices.

SMT Surface Mount Technology.

SOIC Small Outline Integrated Circuit.

SOT Small Outline Transistor.

SPI Serial Peripheral Interface.

SQL Structured Query Language.

SSH Secure SHell.

SSL/TLS Secure Socket Layer / Transport layer Security.

TCP Transmission Control Protocol.

THT Through-Hole Technology.

TTL Transistor to Transistor Logic.

UART Universal Asynchronous Receiver-Transmitter.

UGR Universidad de Granada.

ULDTP Universal Limiter Data Transmission Protocol.

URL Uniform Resource Locator.

USB Universal Serial Bus.

VNC Virtual Network Computing.

WiFi Wireless Fidelity.

WSGI Web Server Gateway Interface.

XML eXtensible Markup Language.

CAPÍTULO

1

INTRODUCCIÓN

El presente Trabajo Fin de Máster se presenta como muestra final de los conocimientos adquiridos durante la realización del Máster en Ingeniería de Telecomunicación. El principal objetivo del mismo es la implementación de un sistema integral de monitorización ambiental enfocado a los parámetros de ruido acústico más relevantes.

El proyecto parte de necesidad de integrar todas las funcionalidades que son necesarias para las tareas de recolección de datos ambientales y acústicos en un solo sistema. Este sistema habrá de ser flexible y acorde a las corrientes tecnológicas actuales.

Para ello se tendrán en cuenta todos los escenarios posibles, desde la monitorización y control del aislamiento acústico en locales hasta la recolección de datos medioambientales de las vías públicas. Por otra parte, y como se detallará más adelante, se requerirá un sistema de transmisión de información, así como un sistema de gestión de dichos datos

Todo ello conformará un sistema completo, en el que diferentes usuarios podrán acceder a datos e interaccionar con los mismos conforme a sus respectivos permisos. Este tipo de información es útil para las administraciones en materias de medición de la calidad del ambiente así como en control del ruido de cualquier tipo de actividades existentes en un área urbana, ya sea de tipo lúdico, social, urbanístico o de construcción.

En resumen, un conjunto de tecnologías telemáticas y electrónicas, correctamente integradas conseguirán solucionar un problema concreto de adquisición y representación de datos de la naturaleza mencionada. Bajo esta premisa, y tal y como se comentará en la sección 1.1, el sistema nacerá de la filosofía *Internet Of Things*.

1

A pesar de que este concepto esté orientado a objectos cotidianos conectados a Internet de forma masiva y el producto de este proyecto sea una tecnología para un nicho no tan amplio, las tecnologías, filosofías y metodologías a utilizar durante el proceso de desarrollo tendrán muchas similitudes.

En cuanto a la metodología de trabajo, se pretende que con este proyecto se lleve a cabo una intesa labor de integración de tecnologías innovadoras para solucionar un problema de la forma más moderna y eficiente posible. Sin embargo no se dejarán de lado los procesos de análisis y diseño de la metodología tradicional, sobre todo en las partes relacionadas con la electrónica y el prototipado del producto.

1.1 Estado del arte: *Internet Of Things*

Antes de comenzar, y puesta sobre la mesa la motivación innovadora del proyecto, es necesario enmarcar todo el trabajo que se detallará en las sucesivas páginas dentro del concepto *Internet Of Things*.

Según Cisco, *Internet Of Things* hace referencia a la tendencia hacia la interconexión total de los dispositivos. Según las estimaciones de dicha empresa para 2020 existirán unos 50000 M de dispositivos conectados a Internet [2]. Como es lógico, las tecnologías que acompañan a esta tendencia generarán en los sucesivos años nuevos modelos de negocio y una renovación de los paradigmas actuales de computación e interconexión.

Tal y como se observa en la figura 1.1, de ser ciertas las estimaciones de Cisco, el crecimiento histórico en los último 10 años ha sido de carácter prácticamente exponencial.

No es de extrañar este gran crecimiento desde un plano económico dadas las grandes rondas de inversión por parte de la Unión Europea dentro de los programas de investigación y desarrollo conocidos como *Research Framework Programmes*. Como datos relevantes se puede observar que en los últimos 7 programas se invirtieron 118233 M€ [25] durante el periodo comprendido entre 1984 y 2013.

En comparativa, el conocido como Horizonte 2020 (*Research Framework Programm 8*) acumulará según estimaciones unos 80000 M€ entre 2014 a 2020 [4]. Las áreas de conocimiento de los proyectos amparados por este programa se cuentan por decenas, sin embargo, rápidamente se pueden encontrar en la lista temáticas que llevan de vuelta al título de esta sección: tecnologías celulares 5G, *Smart Cities*, *Cloud Computing*, *Big Data*, *Machine Learning*... O por resumir de alguna manera y citando un tema concreto del Horizonte 2020: *Smart Anything Everywhere* [14].

Desde un punto de vista tecnológico es relevante tener en cuenta el auge en los últimos años de las plataformas de prototipado (Arduino, BeagleBone, Raspberry Pi...), tecnologías de prototipado rápido (servicios de fabricación de PCB como Dirty PCBs, impresoras 3D...) o los lenguajes de alto nivel (Python, Ruby, JavaScript...) y sus correspondientes los *frameworks* de desarrollo rápido de aplicaciones (Django, Ruby on Rails, Express...).

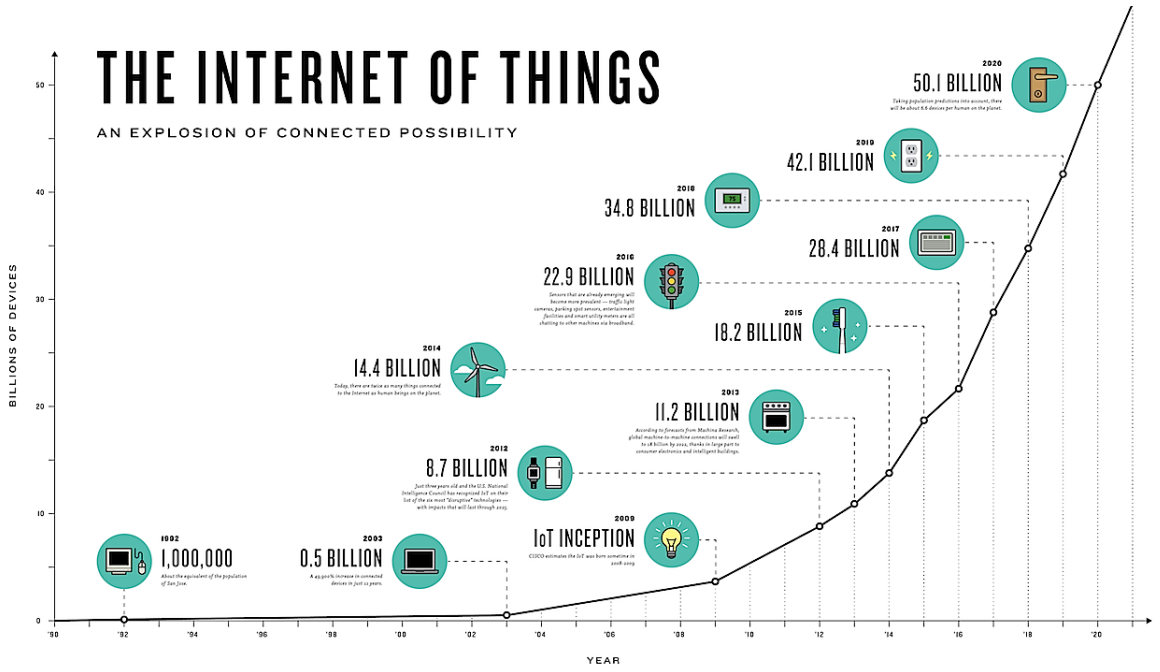


Figura 1.1 – *Evolución del número de dispositivos conectados a Internet [5].*

La relevancia de dichas tecnologías reside en que permiten realizar un proceso de diseño enfocado a prototipo en plazos relativamente cortos. De esta manera los costes de desarrollo e investigación se reducen y es posible invertir en sectores poco ágiles y rentables del mercado. Bajo esta premisa, en este trabajo se propone el desarrollo un prototipo completo y funcional del sistema de monitorización anteriormente comentado.

1.2 Motivación

En primer lugar la más importante motivación de este Trabajo Fin de Máster es poner broche final a los cerca de dos años invertidos en esta formación de postgrado.

Por otra parte, dada la vocación por el trabajo experimental del alumno, la posibilidad del desarrollo de un producto dentro del contexto de una empresa local, viendo su evolución desde el concepto hasta la fabricación, es una experiencia de gran valía personal para el mencionado.

Del mismo modo, ofrecer soluciones que respondan a problemáticas reales pone de manifiesto la utilidad de los conocimientos adquiridos a lo largo de los años de estudio y refuerza la motivación y la confianza del propio alumno en sus capacidades y en dichos conocimientos, ya sean estos últimos en las especialidad de telemática y tecnologías de la información (mención de Grado del alumno), así como en la especialidad de electrónica y sus procesos tecnológicos (esta última responde a los intereses personales del alumno).

1

Poniendo el punto de mira en el propio proyecto, el diseño y la implementación de los sistemas que se tratarán en el presente documento responde a necesidades específicas de una empresa local del sector de la acústica arquitectónica.

El producto o conjunto de ellos resultantes del trabajo, permitirá a dicha empresa una integración vertical hacia arriba mediante la cual dispondrán de la tecnología y el *know-how* necesario para comercializar y generar nuevos modelos de negocio, así como servicios que hasta ahora era necesario comprar o subcontratar.

1.3 Objetivos del proyecto

Conforme a lo expuesto hasta ahora, los principales objetivos de este proyecto son:

- Conocer y analizar las necesidades reales del sistema de monitorización de ruido ambiental requerido.
- Extraer los requisitos principales y secundarios así como los diferentes subsistemas que conformarán el producto, valorar la viabilidad de cada uno de ellos y proponer nuevos puntos de vista y opciones.
- Realizar un análisis de tecnologías disponibles mediante las cuales se puedan solucionar las necesidades de cada uno de los requisitos.
- Diseñar y fabricar un prototipo del producto electrónico de monitorización de variables ambientales.
- Realizar los test de validación y las pruebas necesarias para determinar que la fase de prototipado ha sido sido exitosa.
- Acercar al alumno al trabajo real con empresas tecnológicas.
- Formar de al alumno en el diseño de productos electrónicos.
- Poner de manifiesto los conocimientos adquiridos por el alumno principalmente durante el Máster en Ingeniería de Telecomunicación y reforzar los adquiridos durante en la especialidad de telemática y tecnologías de la información en el Grado en Ingeniería de Tecnologías de Telecomunicación.
- Superar la asignatura de Proyecto Fin de Máster con éxito.

1.4 Estructura del proyecto

El proyecto se divide en 6 capítulos y un anexo que describen cada una de las partes del proceso de desarrollo del producto propuesto. Estos capítulos pretenden describir de un modo lógico y cronológico el trabajo llevado a cabo durante la duración del proyecto.

Los capítulos que conforman el presente documentos son:

- El presente capítulo, numerado como 1, pretende ser una introducción al proyecto tanto en su faceta de Trabajo Fin de Máster como en su faceta de proyecto de colaboración tecnológica con una empresa.
- El capítulo 2 es un breve resumen de la información extraida de las conversaciones iniciales con el cliente. De ahí se extrae una idea global de las necesidades principales y secundarias del producto a diseñar en forma de requisitos.
- El capítulo 3 introduce el procedimiento de desarrollo que se utilizará a lo largo del proyecto (figura 3.1) para a continuación presentar un primer análisis en profundidad de los requisitos del cliente del capitulo 2. De este análisis en profundidad se extraerán los requerimientos técnicos del producto (sección 3.1). Una vez concretados estos requerimientos y siguiendo con la metodología de trabajo, se analizarán los pros y contras de diferentes tecnologías que solucionan las problematicas de cada uno de los subsistemas que se involucran en el diseño del producto. De este análisis se extraen un conjunto de soluciones tecnológicas viables para el producto, así como una temporización de desarrollo que se puede observar en el diagrama de Gantt de la figura 1.2.
- A continuación, el capítulo 4 detalla como con estas tecnologías seleccionadas se diseñan soluciones óptimas a los problemas que presenta el desarrollo del producto además de los detalles de implementación que resultan relevantes para describir en profundidad el proceso.
- Tras realizar el análisis y siguiendo la metodología propuesta en la figura 3.1, se presentan en el capítulo 5 una serie de pruebas de validación y test en las que se pondrá de manifiesto que el producto diseñado durante el capítulo 4 cumple con los requerimientos técnicos del capítulo 3 y por ende con los requisitos impuestos por el cliente en el capítulo 2.
- Por último, el capítulo 6 incluye un conjunto de conclusiones a modo de resumen del documento y algunas lineas de trabajo futuro que se han planteado durante el proceso de desarrollo del producto a implementar.
- Como complemento, en el anexo A se detalla el presupuesto y los costes asociados a este proyecto.

1

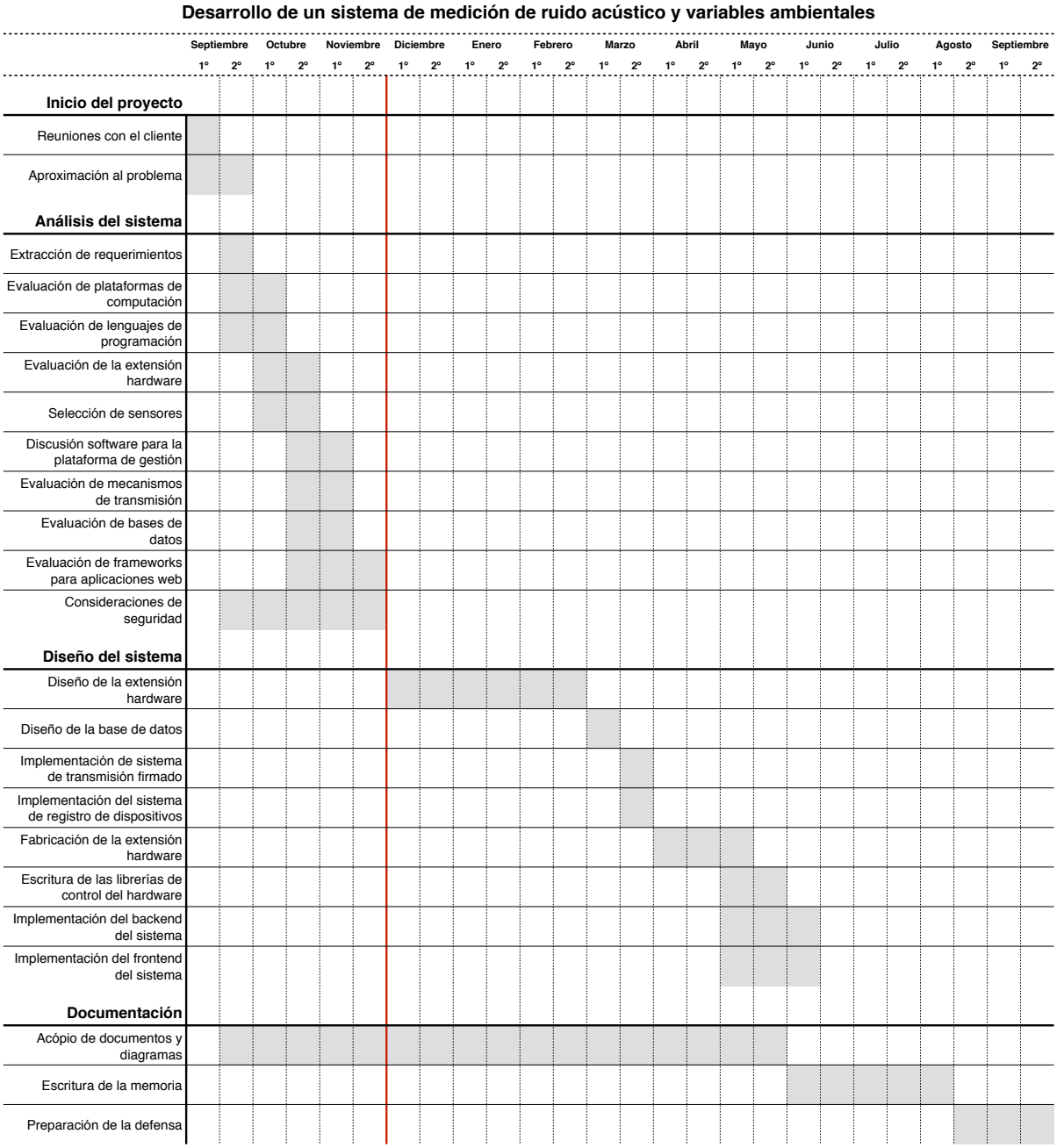


Figura 1.2 – *Diagrama de Gantt del desarrollo del proyecto.*

CAPÍTULO

2

DEFINICIÓN DE REQUISITOS

Tras introducir el tema de estudio del presente proyecto, en este segundo capítulo se definirán cuales son los requisitos principales y secundarios sobre los que el trabajo discurrirá.

2.1 Requisitos del cliente

Tal y como se mencionó en el apartado 1.2, este conjunto de requisitos responde a las necesidades de mercado analizadas por un empresa del sector de la acústica arquitectónica. A continuación se exponen los requisitos tal y como los expuso el cliente en su petición inicial, los **principales** son:

1. Un sistema de medición del ruido acústico capaz de tomar medidas de sonómetros de clase 1.
2. Que el sistema de medida tenga controles visuales y táctiles para configurarlo.
3. Que el sistema permita extraer los datos a una memoria USB.
4. Que el sistema transmita los datos hacia Internet.
5. Que los datos alojados en Internet se puedan consultar mediante una interfaz gráfica.

Por otra parte, de forma **secundaria** se requiere:

1. Un sistema de medición de variables ambientales genéricas, todas las que sea posible.
2. Que el sistema se alimente mediante una conexión a la red eléctrica.
3. Que los usuarios de la interfaz gráfica tengan diferentes permisos.
4. Que la interfaz gráfica pueda consultarse desde ordenadores, móviles o tabletas.
5. Que el sistema sea compatible con aparatos de medición de ruido similares del mercado.

CAPÍTULO

3

ANÁLISIS DEL SISTEMA

En este tercer capítulo se llevará a cabo el análisis de los requisitos del sistema para, más adelante, concluir en una solución mediante una aproximación *top-down*. El primer paso de esta aproximación al problema será describir los requerimientos técnicos del sistema a partir de los requisitos del cliente. A continuación, se analizarán diferentes tecnologías mediante las cuales dar solución a estos requerimientos y subsistemas. Todo ello, para posteriormente implementar las soluciones seleccionadas en el diseño del producto en el capítulo 4.

El proceso de desarrollo del producto está dividido en varias fases, a saber: obtención de requisitos, análisis, diseño, verificación y pruebas. El diagrama de la figura 3.1 muestra como discurre el proceso de desarrollo desde la propuesta del cliente hasta la obtención de un producto.

En primer lugar, este proceso de desarrollo plantea evaluar e interpretar los requisitos del cliente, expuestos en el capítulo 2 para obtener requerimientos técnicos del sistema. A continuación el proceso de análisis dividirá el problema en diferentes subsistemas y realizará diferentes propuestas de posibles tecnologías para cada uno de ellos. En general estos subsistemas se pueden agrupar en categorias como *hardware*, *software* y *firmware*.

El proceso de análisis determina una solución óptima para cada uno de los subsistemas y una evaluación temporal del desarrollo del proyecto (que se puede observar en la figura 1.2). Es entonces cuando la fase de diseño se pone en marcha implementando las soluciones óptimas descritas por el análisis del problema. Esta fase de diseño se resolverá con un prototipo de producto que habrá de ser validado, verificado y testeado.

Una vez realizadas las pruebas y las validaciones, si el resultado cumple los requisitos iniciales del cliente y los requerimientos técnicos extraídos, se habrá obtenido el producto y finalizado el proyecto.

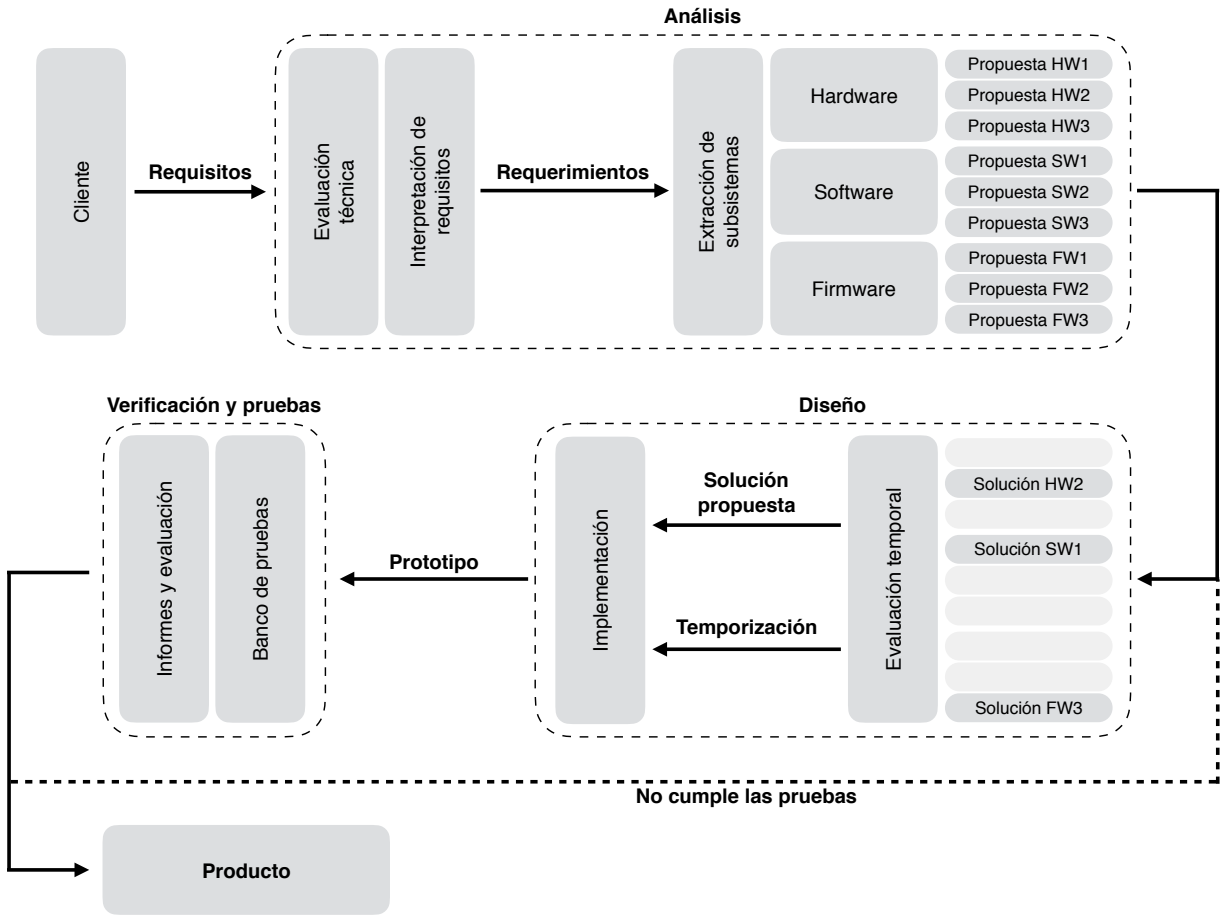


Figura 3.1 – *Proceso de desarrollo del producto.*

3.1 Requerimientos del sistema

El primer análisis de los requisitos del cliente nos muestra que el objetivo principal del proyecto es obtener un producto capaz de obtener, transmitir y gestionar variables ambientales de carácter acústico principalmente, así como, otras variables ambientales en segundo lugar.

El sistema, según las especificaciones del cliente, contará con los módulos indicados en la figura 3.2.

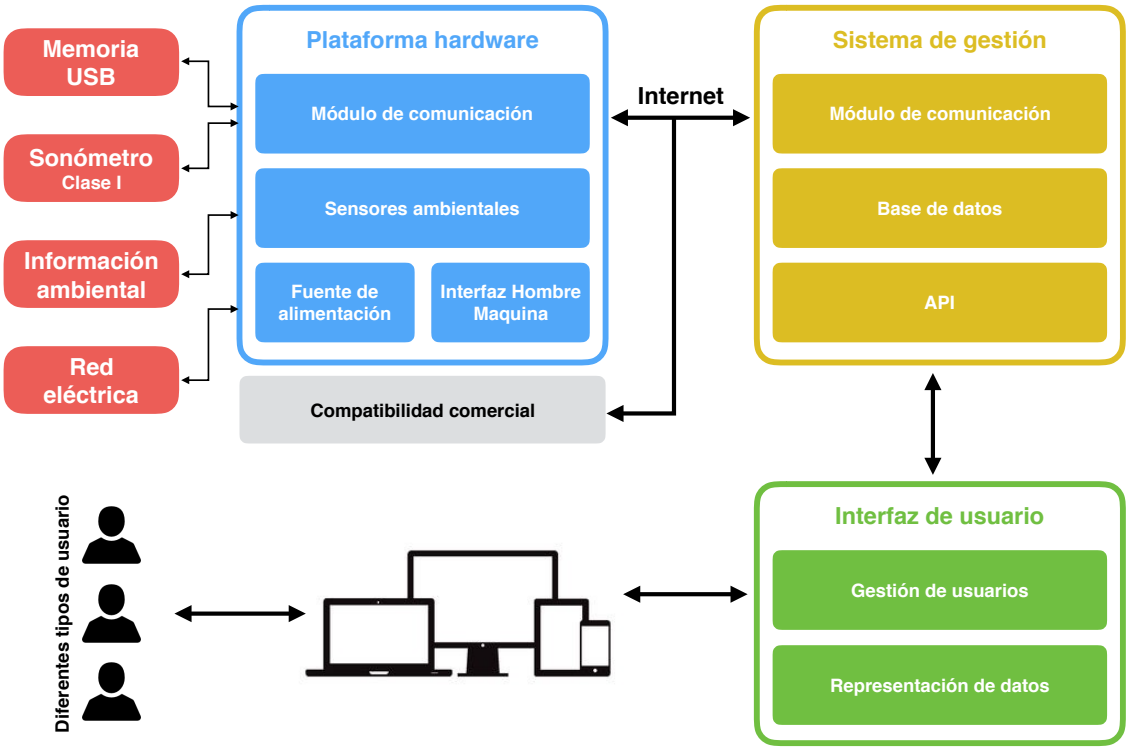


Figura 3.2 – *Esquema del sistema completo a partir de las especificaciones.*

En general, los requerimientos pasan por una plataforma *hardware* capaz de comunicarse con un sistema de gestión de datos alojado en Internet. Esta comunicación deberá producirse mediante protocolos diseñados para este propósito. Por otra parte, una interfaz de usuario o GUI permitirá la gestión y explotación de los datos alojados en dicho servidor.

A continuación se exponen los diferentes subsistemas:

3.1.1 *Hardware*

Como requerimiento principal el sistema ha de incluir un producto *hardware* que solucione los problemas de recopilación de los datos ambientales.

Tal y como muestra la figura 3.3, este sistema deberá de estar compuesto por cinco partes funcionales principales:

1. Una **plataforma de computación** capaz de procesar e interpretar de forma cómoda y programática los datos de los sensores ambientales. Dentro de esta plataforma se deberá implementar una capa *software* que incluya todas las librerías de control de periféricos.

2. Una **extensión *hardware*** que integre todos los subsistemas de medida que requieran las especificaciones. Esta extensión deberá comunicarse a través de buses digitales con la plataforma de computación.

3. Una extensión *hardware* que permita una mínima **interacción hombre-maquina** mediante botones, pantallas de pequeño tamaño y sonidos. El propósito de esta extensión será mostrar datos relevantes de la configuración o estado instantáneo del sistema.

4. Un conjunto de subsistemas que permitan la conexión del producto hacia Internet ya sea de forma cableada o inalámbrica, así como la conexión y gestión remota del dispositivo.

5. Una **fuente de alimentación** que permita al sistema obtener la energía de la red eléctrica.

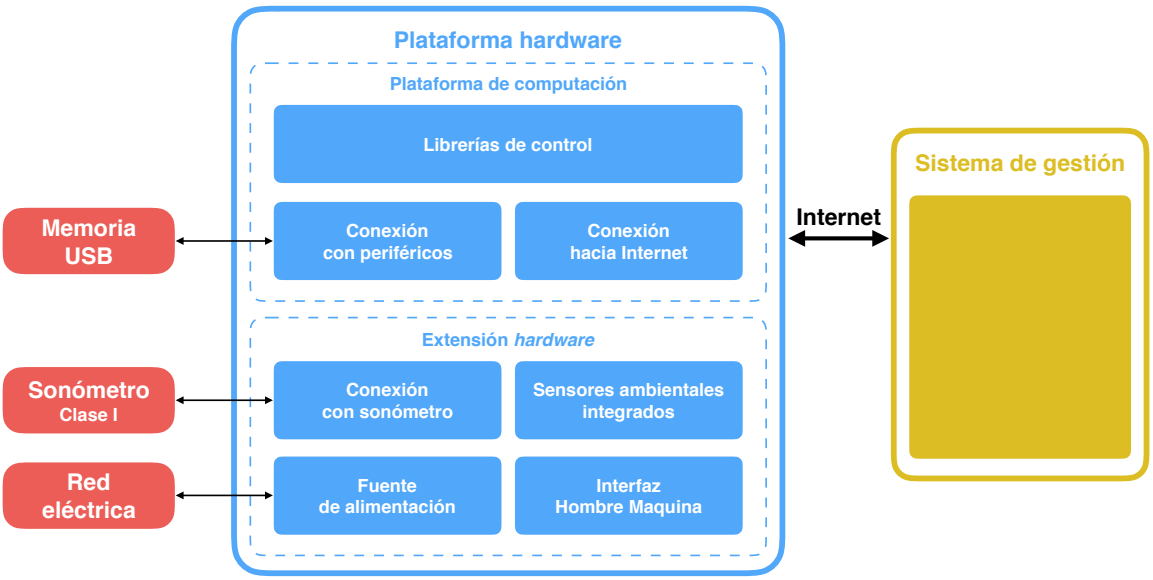


Figura 3.3 – *Detalle de subsistemas hardware.*

Como requisito transversal a todos ellos, y tal y como se presentó en la sección 1.1, el desarrollo de este hardware habrá de enfocarse al bajo costo.

3.1.2 Sistema de gestión

El conjunto de datos recopilados por los dispositivos dispositivo habrá de ser trasladado mediante una conexión a Internet hacia uno o varios servidores capaces de procesar, almacenar y gestionar esa información.

Siguiendo el flujo lógico de los datos tras ser recibidos en el sistema de gestión, los requerimientos del mismo son:

1. Permitir la conexión y recepción de datos desde un número arbitrario de dispositivos *hardware* los cuales se deben identificar mediante un **número de serie**.
2. Procesar el paquete de información recibido para extraer los datos relevantes e insertarlos en una base de datos.
3. Presentar la información de forma ordenada en un interfaz de usuario prestando especial atención a que dicha aplicación ha de estar enfocada a roles y permisos. Esta estructura se detalla en la sección 3.1.2.1.
4. Generar los informes requeridos, y permitir extraer y descargar la información necesaria. Estos informes habrán de incorporar todas las gráficas, tablas y resúmenes que se considere oportuno para el correcto análisis e interpretación de los datos. La información ha de poder descargarse en **formatos estándares** para poder ser procesada por otras herramientas.
5. De forma complementaria, ofrecer un sistema en el se pueda administrar toda la información relativa a los dispositivos *hardware* registrados en el sistema así como todos y cada uno de los usuarios y entidades junto con sus permisos.
6. De forma transversal, utilizar tecnologías modernas y flexibles, así como metodologías de desarrollo rápido acordes a las posibilidades que ofrezca actualmente el estado de la técnica.
7. Con carácter imprescindible, se habrá de diseñar un sistema robusto frente a **amenazas a la seguridad**. Dentro de lo posible se habrá de tener en cuenta durante todo el desarrollo cuestiones relativas a la disponibilidad y dimensionado del servicio, integridad y confidencialidad de los datos, así como el no repudio por parte de cada una de las entidades de lo acontecido durante las mediciones en el sistema

En la figura 3.4 se puede observar de forma sintetizada los requerimientos enumerados en esta sección.

3.1.2.1 Roles y permisos

El sistema de presentación de datos o interfaz de usuario habrá de tener en cuanta que existen cuatro roles posibles para sus usuarios: gestor, instalador, cliente y administración.

Sus relaciones, tal y como se representan en la figura 3.5, son las siguientes:

- Los usuarios con permisos de **gestor** pueden acceder a la información completa existente en el sistema así como modificarla o generar nuevos usuarios de cualquier tipo.

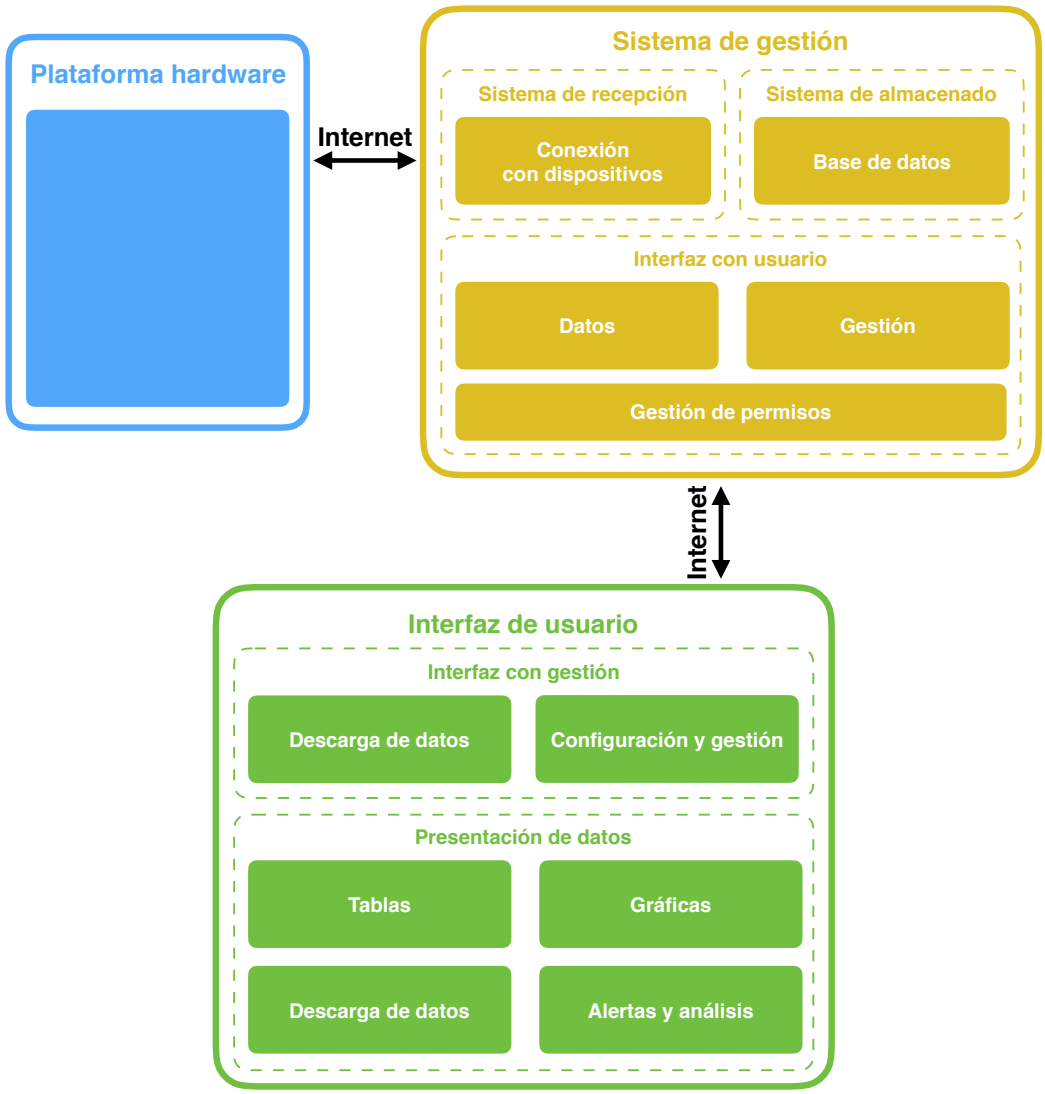


Figura 3.4 – *Detalle del subsistema de gestión.*

- Los usuarios con permisos de **instalador** pueden dar de alta nuevos **clientes** y registrar nuevos dispositivos asociados a sus propios clientes. Estos dispositivos se asociarán a una administración. También podrá acceder y modificar toda la información relativa a sus propios clientes y a los dispositivos asociados a ellos.
- Los usuarios con permisos de **cliente** tienen acceso sin permisos de modificación a la información generada por sus propios dispositivos así como a datos de contacto de su **instalador** y **administración** asociados.
- Los usuarios con permisos de **administración** tienen acceso sin permisos de modificación a los dispositivos que algún **instalador** le haya vinculado. Tendrán también acceso a información de contacto de los **clientes** asociados a los dispositivos así como a los **instaladores** del mismo.

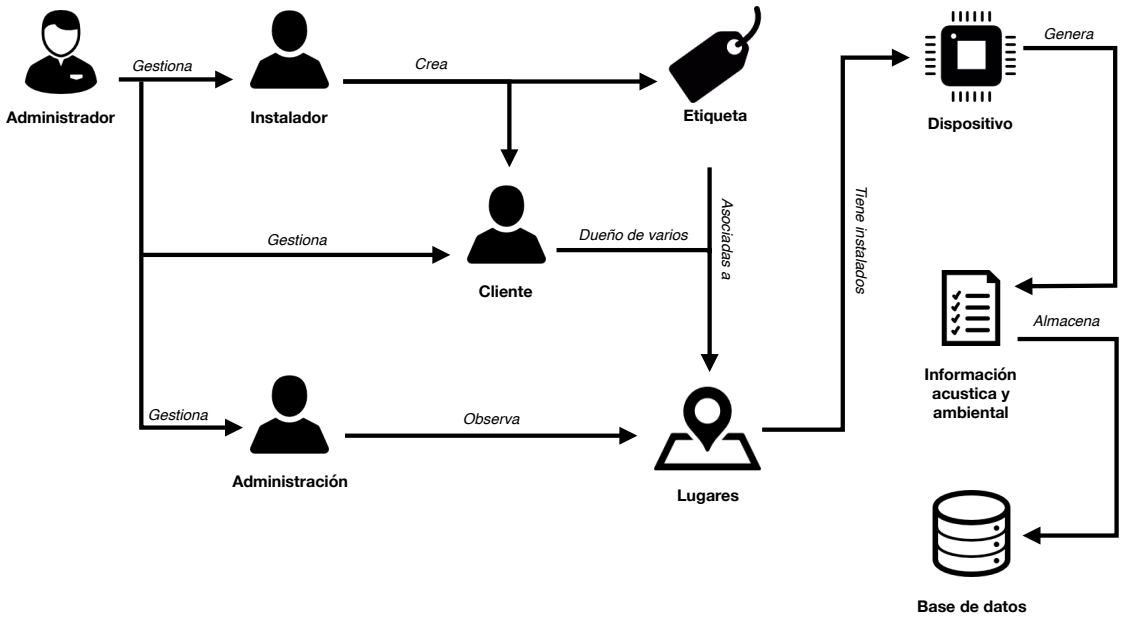


Figura 3.5 – *Modelo de datos y entidades requerido.*

3.1.3 Transmisión de datos

La transmisión de datos entre el subsistema *hardware* y el sistema de gestión de datos habrá de centrarse principalmente en la flexibilidad, tratando de buscar un compromiso entre la optimización de la transmisión (en tiempo y ancho de banda) y la facilidad en el tratamiento de los datos de dicha transmisión. Por otra parte, el orden de magnitud del periodo de transmisión de los datos desde la plataforma *hardware* hasta el sistema de gestión será de **minutos**.

Como objetivo secundario pero imprescindible, el sistema de gestión de datos habrá de ser compatible con otros posibles protocolos de comunicación existentes en el mercado sean del tipo que sean. Para ello será necesario realizar un análisis de las tecnologías existentes en el mercado, comprender los mecanismos de transmisión implementados en ellas y generar un **mecanismo de recepción actualizable e independiente del sistema de gestión principal**.

Este mecanismo de recepción deberá ser capaz de interpretar los datos recibidos y transmitirlos al sistema de gestión en el formato que este utilice. El proceso de actualización de este mecanismo para añadir nuevos protocolos de comunicación deberá ser sencillo si se conocen los pormenores del sistema de transmisión a añadir.

De forma transversal a todo este diseño, y tal y como especificó en el punto 7 de la lista de requisitos del subsistema de gestión, el protocolo habrá de ser robusto en la medida de lo

posible frente a amenazas a la disponibilidad, integridad, confidencialidad y no repudio.

3.2 Plataforma de computación

Uno de los pilares fundamentales del futuro diseño del producto *hardware* que complementa el sistema de medida de ruido y variables ambiental es la plataforma de computación.

Tal y como se definió en la sección 3.1.1, este subsistema ha de proporcionar una solución óptima a los procesos de comunicación, recolección y procesado de los datos de los sensores y posterior transmisión de los mismos a través de Internet hacia el sistema de gestión.

Comenzando por la comunicación con los sensores (punto 2 de los requerimientos *hardware* de la sección 3.1.1), para cubrir los métodos de comunicación más comunes sería de esperar que el sistema de computación facilitase el acceso a los siguientes mecanismos:

- Buses digitales de comunicación: I2C, UART (TTL y RS232), SPI...
- Buses digitales de alto nivel: USB
- GPIO suficientemente rápido para realizar Bit-banging de protocolos no implementados en *hardware*.

De forma secundaria, en base a el item 3 de la lista de requisitos *hardware* del producto (3.1.1) el subsistema habrá de proporcionar las interfaces necesarias para, además de los sensores, poder gestionar dispositivos de entrada y salida como botones, pantallas LCD o *buzzers*.

Teniendo en cuenta el item 4, sería conveniente que la plataforma de computación facilitase interfaces de comunicación hacia Internet tales como *Ethernet* o *WiFi*.

Teniendo en cuenta las anteriores consideraciones junto con el punto 1 de las especificaciones, que indica que es necesaria una capa de librerías *software* para controlar y gestionar la propia plataforma de forma programática, comienza a perfilarse que es necesario una plataforma de computación que se asemeje más a un computador tradicional que un sistema basado en microcontroladores.

El auge de los SBC dan a esta solución unos argumentos de peso en cuanto a precio, potencia de calculo y capacidades de comunicación frente a los sistemas basados en microcontroladores como pudiera ser, por ejemplo, Arduino.

Existen en el mercado multitud de SBC que están constantemente actualizándose y evolucionando en cuanto a prestaciones y capacidades, es por ello que en la tabla 3.1 se realiza una comparación de prestaciones de tres de los modelos más comunes dentro de este tipo de plataformas: Raspberry Pi, BeagleBone e Intel Galileo.

Plataforma	Raspberry Pi 2 Model B	BeagleBone Black	Intel Galileo
Procesador	ARM Cortex A7	ARM Cortex A8	Intel Pentium
Reloj	900 MHz	1 GHz	400 MHz
RAM	1 GB	512 MB	256 MB
Flash	SD Card	4 GB + SD card	8 MB + SD card
Consumo	800 mA (4.0 W)	500 mA (2.5 W)	3 A (15 W)
GPIO (PWM)	26 (17)	65	14 (6)
Entradas analógicas	0	6	6
Buses Digitales	I^2C, UART, USB	I^2C, UART, USB	I^2C, SPI, USB, PCIe, UART
Ethernet	10/100 Ethernet	10/100 Ethernet	10/100 Ethernet
Entradas audio	Analógica, HDMI	HDMI	-
Precio	33.00 €	79.00 €	75.00 €
Selección	✓	×	×

Tabla 3.1 – *Comparación de Single Board Computers.*

Además de la comparativa de prestaciones expuesta en la tabla 3.1 existen otros factores a tener en cuenta, por ejemplo, el soporte ofrecido y la comunidad de desarrolladores. Respecto a esto último la plataforma Raspberry Pi supera con creces a las otras dos soluciones propuestas.

De esta manera, un análisis cuantitativo y cualitativo de las prestaciones y facilidades de cada uno de los sistemas se puede resolver conforme a lo siguiente:

1. Únicamente las plataformas Raspberry Pi y BeagleBone permiten instalar de forma trivial sistemas operativos basados en GNU/Linux sobre los que implementar las librerías necesarias.

2. Todas las propuestas disponen de suficientes buses digitales y GPIO para suplir las necesidades expuestas en la lista del inicio de la sección 3.2. Sin embargo la plataforma Raspberry Pi dispone de una gran comunidad que pone a disposición pública diseños de extensiones *hardware* conocidas como Raspberry Pi HAT y librerías de control de las mismas.

3. Todas las propuestas disponen suficientes GPIO para controlar y gestionar una interfaz hombre-maquina simple.

4. Únicamente las plataformas Raspberry Pi y BeagleBone disponen, gracias a los sistemas operativos, mecanismos de comunicación complejos a través de Internet. Dada la naturaleza de este punto y del resto de requerimientos la comunicación del dispositivo hacia Internet, es importante que esta no se limite a la capa de TCP/IP, si no que se deberán implementar soluciones que hagan uso de *sockets* SSL/TLS, protocolos de capa de aplicación como SSH, VNC o HTTP Websockets.

Expuestas todas las conclusiones, para este proyecto se utilizará la plataforma de computación Raspberry Pi. Concretamente el modelo **Raspberry Pi 2 Modelo B+** como el que se muestra en la figura 3.6, con opción a utilizar otro superior siempre que sea compatible a nivel de *pinout* y *software*.

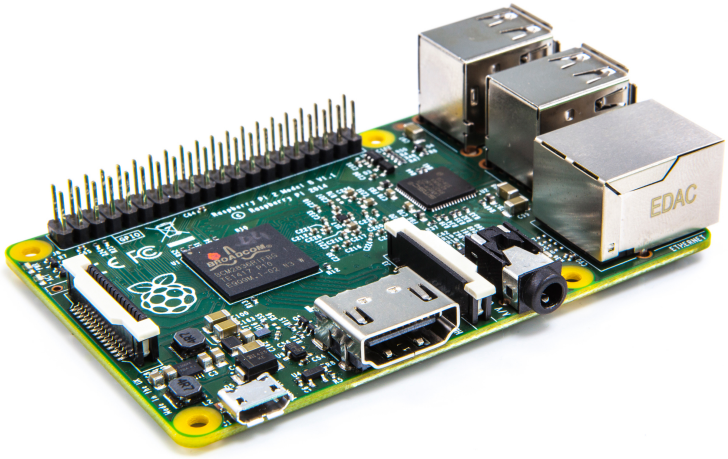


Figura 3.6 – *Plataforma de computación Raspberry Pi 2 Modelo B+.*

3

3.3 Extensión *hardware*

En segundo lugar, tal y como se explicita en los requisitos del cliente (2.1), el sistema habrá de medir la mayor cantidad de variables ambientales posibles. La lista seleccionada es es la siguiente:

1. Medición de los niveles acústicos en el ambiente.
2. Medición de los niveles de humedad en el ambiente.
3. Medición de la temperatura del ambiente.
4. Medición de la luminosidad en los espectros visibles y ultravioleta.
5. Obtención de la posición geográfica mediante GPS.

Durante el proceso de análisis se valorará la inclusión de cada uno de estos sistemas de medida dentro del producto utilizando a criterios de utilidad en relación a coste.

En cuanto a los niveles acústicos cabe realizar una descripción más en detalle, ya que se mencionarán frecuentemente a lo largo del documento.

3.3.1 Variables ambientales acústicas

Tradicionalmente en acústica, el nivel sonoro se define directamente a partir de la presión que produce en el aire la vibración acústica o el sonido. Se toma como referencía el menor valor de presión audible tal y como se muestra en la ecuación (3.3.2). Este nivel tambien es conocido como el nivel de presión sonora o L_{SPL}:

$$dB = L_{SPL} = 20log(\frac{P}{P_0}) \tag{3.3.1}$$

$$P_0 = 20\,\frac{\mu N}{m^2} = 20\,\mu Pa \tag{3.3.2}$$

En general, al ser el sonido una onda mecánica, podemos definir la potencia acústica como la energía que transporta la onda por unidad de tiempo a través de una superficie dada (3.3.3). A partir de dicha potencia expresada en watios, se puede extraer otra expresión (3.3.4) para el decibelio descrito en (3.3.1).

$$P_S = \int_S I_S\, dS \tag{3.3.3}$$

$$dB = 10log(\frac{W}{W_0}) \tag{3.3.4}$$

$$W_0 = 10^{-12}\ W \tag{3.3.5}$$

Sin embargo estas definiciones puramente físicas no corresponden con los datos que se esperan extraer de una medición del ruido ambiental, ya que este depende de un factor que no se ha tenido en cuenta hasta ahora: el oído humano.

Para tener en cuenta la respuesta en frecuencia del oído humano se utilizan curvas de poderación isofónicas, es decir curvas obtenidas de forma empírica que muestran los niveles sonoros de cada frecuencia del espectro con los que el oído humano percibe la misma intensidad subjetiva. Representan, por lo tanto, aproximaciones empíricas a la respuesta en frecuencia del oído humano.

Una de las curvas de ponderación isofónica más utilizadas se conoce como *A Weigthing* [17]. Existen además las conocidas como *B Weigthing*, *C Weigthing* o *D Weigthing*, todas ellas se pueden observar en la figura 3.7.

Ponderando las medidas de nivel de presión sonora con esta curva isofónica A obtendremos las medidas en unidades dBA. Esta unidad si es válida para realizar mediciones de ruido ambiental ya que tiene en cuenta las características del oído humano.

A continuación, sobre estas unidades, se define el nivel de presión sonora equivalente ponderado o $L_{eq,T}$ como una media temporal ponderada tal y como describe la ecuación (3.3.6).

$$L_{eq,T} = 10\,log\left[\frac{1}{T}\int_{t=0}^{T} 10^{0.1L(t)}\,dt\right] \tag{3.3.6}$$

Por último, los niveles porcentuales $L_{N,T}$ son percentiles que muestran el nivel de ruido en dBA superado durante un porcentaje N del tiempo de muestra T.

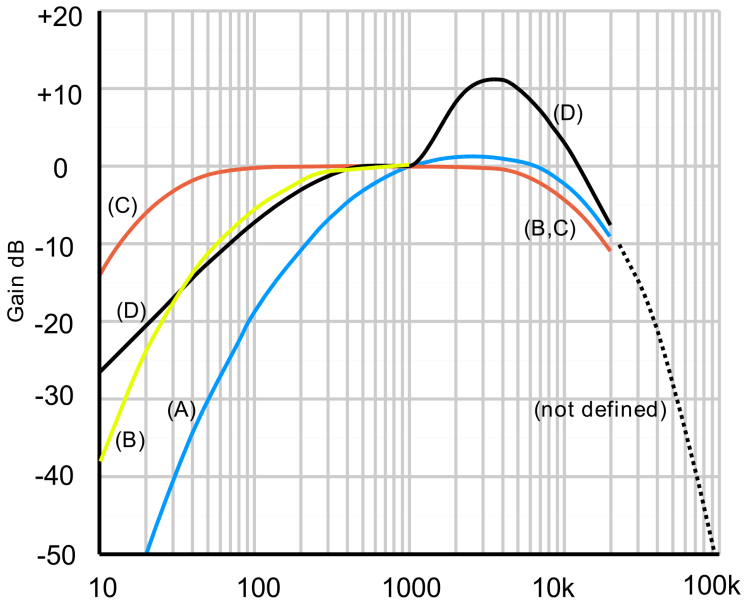


Figura 3.7 – *Curvas de ponderación isofónica A, B, C y D. [19]*

Los dos niveles porcentuales más comunes son:

- $L_{90,T}$ que representa el nivel que ha sido superado un 90% del tiempo de muestra y se puede entender como una medida del ruido de fondo existente.

- $L_{10,T}$ que representa el nivel que ha sido superado un 10% del tiempo de muestra y se puede entender como una medida de los picos de ruido que se han producido durante el periodo de medición.

3.3.2 *Hardware* y fabricación

Para cumplir los requisitos enumerados en la lista de la sección 3.3 es necesaria una ampliación al *hardware* seleccionado en la sección 3.2. Esto se debe a que la plataforma Raspberry Pi unicamente es una plataforma de computación de propósito general sin ninguna de las funciones adicionales que son necesarias para el producto que se está desarrollando.

En primero lugar, cabe mencionar que tal y como indica el punto 2 de las conclusiones del subapartado 3.2, la plataforma de computación Raspberry Pi tiene en cuenta las posibles extensiones que la comunidad de desarrolladores genere para el sistema y facilita dicho trabajo.

Estas extensiones se conocen como Raspberry Pi HAT y para su desarrollo se facilitan las magnitudes mecánicas de la placa, así como la completa documentación de todo el patillaje que existe en la misma (puerto GPIO de 40 pines).

En la extensión *hardware* o Raspberry Pi HAT, se incluirán los siguientes elementos:

- Sensores necesarios para cumplir el requisito expuesto en el punto 2 de la lista de requerimientos de la sección 3.1.1.
- Pantalla LCD y actuadores necesarios para cumplir el requisito expuesto en el punto 3 de la lista de requerimientos de la sección 3.1.1.
- Acceso sencillo al patillaje no utilizado de la placa Raspberry Pi para posibles pruebas y/o extensiones.
- Una fuente de alimentación conmutada que permita alimentar el sistema sin necesidad de usar el puerto USB de alimentación de la propia plataforma de computación.
- Un interfaz RS232 para la comunicación con dispositivos externos al sistema en caso de ser necesaria.

Y su proceso de fabricación se adaptará a las posibilidades del Laboratorio de Electrónica de la Universidad de Granada, es decir:

- PCB realizada mediante fresado CNC, utilizando la maquina de prototipado ProtoMat S62.
- Soldadura SMD y THT manual mediante lupas ópticas, pinzas de pequeño calibre y soldadores de estaño de punta fina.
- Sustrato FR4 de doble cara.
- Posibilidad de metalización de taladros y vías.

3.3.3 Librerías *software*

El control del Raspberry Pi HAT diseñado se realizará mediante programas que se ejecutan en la propia Raspberry Pi, para la realización de estas librerías se proponen dos lenguajes ampliamente aceptados: C y Python.

En la tabla 3.2 se realiza una comparación cualitativa de ambos lenguajes y se concluye que dada la naturaleza de prototipo del producto a desarrollar, los requisitos temporales del orden de minutos en transmisión descritos en la sección 3.1.3 y la facilidad de uso de las librarías de puerto serie, UART o I2C y de comunicación se selecciona el lenguaje **Python 2.7** como mejor candidato para comenzar la programación del sistema una vez fabricado.

3.3.4 Sensores y actuadores

En base a los requerimientos expuestos en la lista de la sección 3.3, se detallan a continuación la lista de sensores seleccionados para incorporarse en la extensión *hardware*:

Lenguaje	C	Python
Propósito	General	General
Nivel	Medio nivel	Alto nivel
Paradigma	Imperativo y estructurado	Orientado a objetos
Complejidad	Media-alta	Media-baja
Soporte y librerías	Muy alto	Muy alto
Tiempos de desarrollo	Alto	Bajo
Tiempos de ejecución	Muy bajo	Medio
Utilidad principal	Optimización de recursos	Facilidad de prototipado
Selección	×	✓

Tabla 3.2 – *Comparación de lenguajes propuestos.*

3.3.4.1 Sensor de humedad y temperatura

En primer lugar una solución a dos items de la lista es el sensor de **humedad** y **temperatura** DHT11. Este sensor se muestra en la figura 3.8. En la tabla 3.3 se pueden observar sus especificaciones técnicas, que dados los requisitos de cliente, cumplen las expectativas. La comunicación con el dispositivo se realiza de mediante una interfaz serie con un protocolo propio del dispositivo, para la comunicación se utilizarán técnicas de Bit-banging.

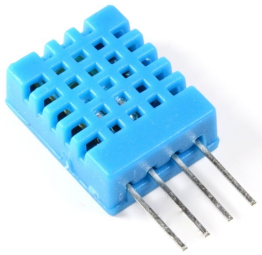


Figura 3.8 – *Sensor de temperatura y humedad DHT11.*

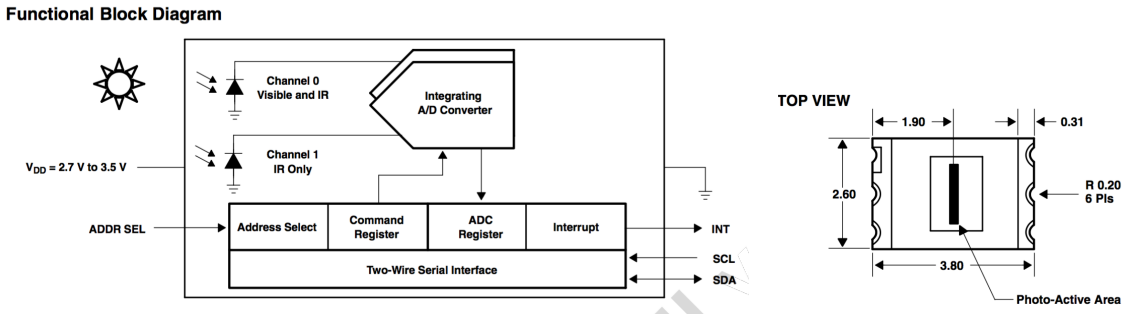
Rango humedad	20-90 % RH
Rango temperatura	0-50 ºC
Precisión humedad	± 5 % RH
Precisión temperatura	± 2 ºC
Resolución	1 % - 1 ºC

Tabla 3.3 – *Especificaciones del sensor DHT11.*

3.3.4.2 Sensor de luminosidad

Por otra parte, para solucionar la medición de la **luminosidad** en el espectro visible e IR se utilizará el sensor TSL2561T. Tal y como muestra la figura 3.9 esta solución permitirá obtener los valores de luminosidad en el espectro visible y en el espectro IR a través de una

interfaz I2C integrada en un encapsulado SMD de 6 pines.



(a) *Esquema funcional* **(b)** *Detalle del encapsulado*

Figura 3.9 – *Diagramas de sensor lumínico TSL2561T [23].*

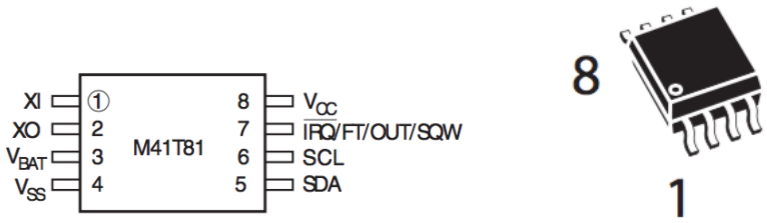
3

3.3.4.3 Gestión del tiempo

La solución a la gestión del tiempo en estados de ausencia de alimentación pasa por la incorporación de un RTC al sistema. Este integrado será el modelo M41T81 y permitirá, gracias a una batería auxiliar, mantener la fecha y hora en caso de pérdida de alimentación y de no disponer de un servidor NTP accesible en el arranque.

Tal y como se observa en la figura 3.10 en un pequeño encapsulado SOIC-8 se puede gestionar y almacenar la fecha y hora de forma *offline* a través de una interfaz I2C y con la ayuda de un cristal de 32.768 kHz. Se podrá, además, generar alarmar e interrupciones configurables a través de un pin dedicado a ello.

Este subsistema requerirá una fuente de alimentación externa e independiente. Previsiblemente será una pila de botón del tipo CR2032. Teniendo en cuenta una capacidad estimada de 200 mAh y dado el consumo del dispositivo RTC (1 µA en modo batería) la duración estimada del dispositivo de gestión de tiempo será de unos 20 años.



(a) *Esquema funcional* **(b)** *Detalle del encapsulado SOIC*

Figura 3.10 – *Diagramas de RTC M41T81 [8].*

3.3.4.4 Sistema de posicionamiento GPS

El módulo Venus 638FLPx con conector SMA distribuido por Sparkfunk [22], solucionará la geolocalización GPS del producto. Dispone de un puerto serie mediante el cual se obtendrán las coordenadas de la antena de recepción.

En la figura 3.11 se puede observar el módulo GPS seleccionado.

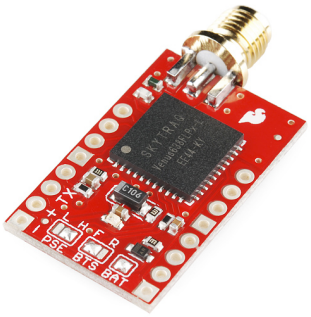


Figura 3.11 – *Módulo Venus 638FLPx de Sparkfun [22].*

3.3.4.5 Sensor de ruido ambiental

Para solucionar el punto 1 de la lista de requisitos del cliente se plantea la utilización de un sonómetro comercial de la marca RION concretamente el modelo NA-27, mostrado en la figura 3.12a.

El hecho de que este dispositivo sea clase 1, hace referencia a la norma CEI 60651. Dicha norma define un sonómetro de clase 1 aquel que genera un error comprendido entre +/- 0.7 dB respecto a un equipo de referencia. Los dispositivos sonómetros de clase 1 permiten realizar mediciones acústicas sobre el terreno con precisión.

Este sistema, tal y como muestra la tabla 3.4, dispone de todas las mediciones acústicas requeridas en la sección 3.3.1. En cuanto a la comunicación con el dispositivo, la extensión *hardware* contará con un puerto RS232 ya que, tal y como se puede observar en la figura 3.12b, el sonómetro dispone en su parte inferior de un puerto de este tipo.

Canal	*Main*	*Subchannel*
Ponderación temporal	Fast, Slow, 35ms y 10 ms	Fast, Slow, 35ms y 10 ms
Pico	No	Sí
Ponderación frecuencial	A, C, Flat	A, C, Flat
Medidas porcentuales	1,5,10,50,90,95,99	1,5,10,50,90,95,99
Medidas	$L_{eq}, L_p, L_{max}, L_E$	$L_{eq}, L_p, L_{max}, L_E, L_{tm3}, L_{tm5}$

Tabla 3.4 – *Especificaciones del sonómetro RION NA-27.*

(a) *Dispositivo sonómetro* **(b)** *Detalle del puerto RS232*

Figura 3.12 – *Sonómetro RION NA-27.*

Teniendo en cuenta que la conexión de sonómetro RION NA-27 es RS232 y la plataforma de computación Raspberry Pi 2 Model B descrita en el apartados 3.2 cuenta con una interfaz serie UART TTL es necesario implementar una solución para adaptar los niveles.

La comunicación UART de la plataforma Raspberry Pi consiste en un puerto de comunicación serie asíncrona con un puerto de transmisión y de recepción. El concepto UART especifica las características temporales de la transmisión, así como el formato de la trama utilizado para transmitir los paquetes binarios. El hecho de que sus niveles sean TTL implica que maneja niveles de voltaje iguales a los utilizados por el propio dispositivo en sus transistores internos, es decir, entre 0V y 3.3V.

Por su parte la especificación RS232 hace uso de las definiciones de paquetes y de temporización de una comunicación UART y define nuevos niveles eléctricos para la capa física. Concretamente la interfaz utilizada por el sonómetro en cuestión utiliza niveles de -12V y 12V.

Es necesario, por tanto la utilización de un adaptador de niveles que será el integrado MAX3232. Este integrado implementa dos adaptadores de nivel bidireccionales y trabaja a una tensión de alimentacion acorde a la de la plataforma de computación. En la figura 3.13 se observa la configuración típica de este dispositivo así como su doble puerto de entrada y de salida.

3.3.4.6 Interfaz humano-maquina

Por último, para solucionar el punto 2 de los requerimientos del cliente, se han de implementar soluciones mediante las cuales un usuario del producto *hardware* pueda interactuar con el sin necesidad de conectarse mediante un ordenador externo.

Dados los requerimientos extraídos en el punto 3 de la lista de requerimientos, será interesante contar con:

1. Un control *rotary-push* capaz de interpretar el giro y las pulsaciones de un *joystick* para implementar controles de selección en menús.

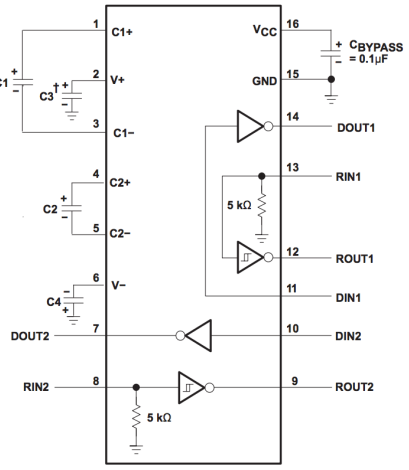


Figura 3.13 – *Esquema de configuración del MAX3232 [9].*

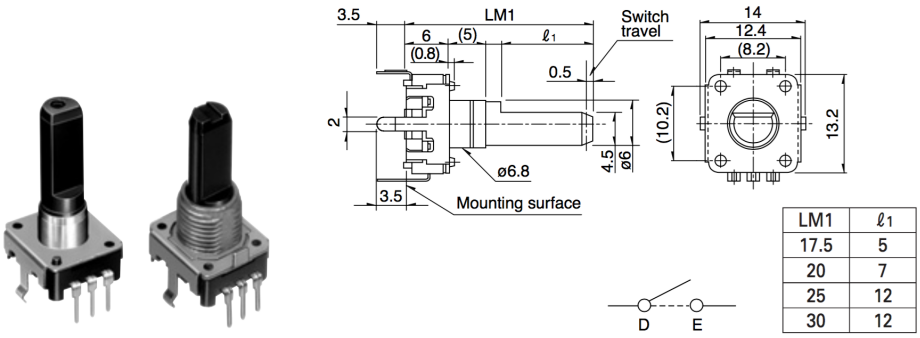
2. Un botón permitirá ejecutar rutinas de encendido y apagado.

3. Un *buzzer* o zumbador permitirar emitir señales acústicas ante eventos del sistema.

4. Un conjunto de LEDs permitirán mostrar los estados del sistema.

5. Una pantalla LCD permitirá mostrar información detallada acerca de parámetros, resultados y configuraciones.

El control *rotary-push* se muestra en la figura 3.14, permitirá detectar el giro y las pulsaciones de un control manual. Consta de cinco pines, tres de ellos dedicados al control de giro y los dos restantes hacen las veces de un pulsador convencional. De forma adiccional se incorporatá un *push button* para implementar funciones de encendido y apagado del sistema, este dispositivo se puede observar en la figura 3.15.



(a) *Rotary Push* **(b)** *Dimensions*

Figura 3.14 – *Rotary Push EC12E [3].*

Figura 3.15 – *Push Button.*

En cuanto al zumbador o *buzzer* será un sistema de dos pines alimentado por PWM. De esta manera será posible generar tonos de determinada frecuencia e interactuar con el usuario mediante sonidos simples. Se puede observar un dispositivo de este tipo en ela figura 3.16.

Figura 3.16 – *Buzzer.*

Por último, un elemento imprescindible en una interfaz humano-maquina es una pantalla capaz de mostrar texto y gráficos sencillos. Para este propósito se selecciona una pantalla LCD de 20 caracteres de largo y 4 lineas de texto similar a la de la figura 3.17. Esta pantalla dispone de retroiluminación y ajuste de contraste, los cuales previsiblemente serán configurables por software.

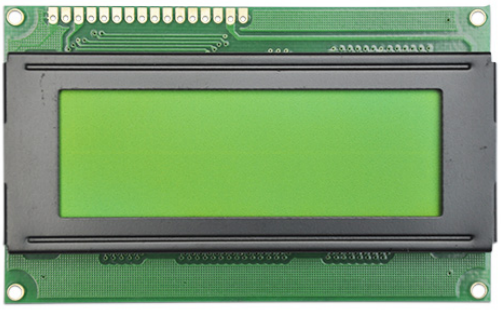


Figura 3.17 – *LCD de 20 caracteres y 4 filas.*

Uno de los problemas relacionados con la utilización de este tipo de *displays* LCD es la gran cantidad de pines digitales GPIO que se requieren para su funcionamiento: control del contraste, control de la retroiluminzación, puerto de comunicación paralelo, etc.

Este hecho hace necesaria la incorporación de un extensor de GPIO al sistema *hardware*. Teniendo en cuenta que los apartados 3.3.4.2 y 3.3.4.3 ya contemplan la necesidad de un bus de comunicación digital I2C, es una buena opción seleccionar un dispositivo que mediante una interfaz I2C permita gestionar un número de pines digitales de entrada y salida, y así interactuar con la pantalla LCD.

Una solución a esta propuesta es el módulo PCF8574, que tal y como muestra su diagrama funcional en la figura 3.18, permite implementar estos propósitos.

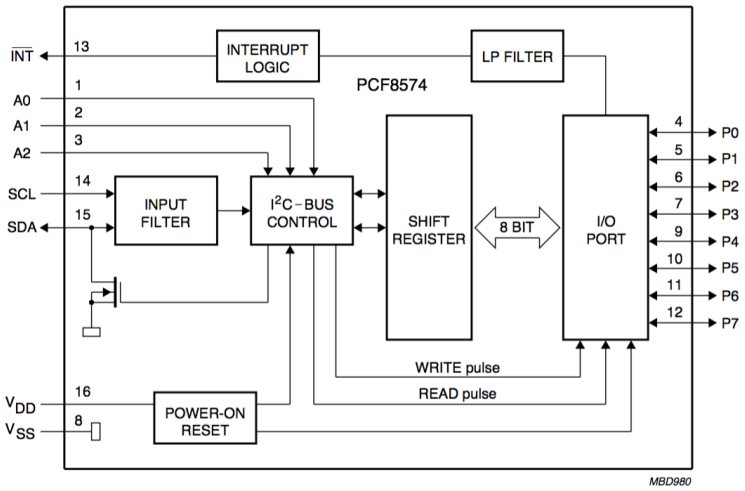


Figura 3.18 – *Diagrama de bloques del módulo expansor de GPIO por I2C PCF8574 [11].*

3.3.4.7 Componentes discretos

En cuanto al resto de dispositivos que conformarán el sistema y complementarán el diseño de los circuitos electrónicos que implementarán las funcionalidad requeridas se decide que estos habrán de ser implementados en tecnología SMT dado que es la tecnología más moderna dentro de las posibilidades.

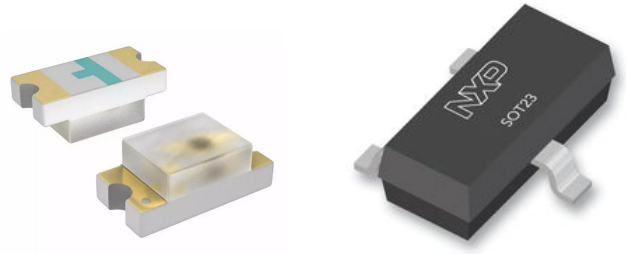
Las ventajas de utilizar tecnología SMT en comparación a la tecnología THT es el ahorro de espacio. Concretamente para este proyecto se utilizaran componentes de tamaños 0805 y 0603. Estos tamaños hacen referencia a dispositivos de 8 mils de largo por 5 mils de ancho y de 6 mils de largo por 3 mils de ancho respectivamente. La figura 3.19 muestra los principales componentes 0805 y 0603 seleccionados para el diseño así como otros modelos de componentes discretos SMD como son los condensadores electrolíticos.

Por otra parte, la figura 3.20 muestra algunos otros dispositivos secundarios a utilizar,

como por ejemplo transistores MOSFET de tecnología SMT y LEDs en formato 0805 para completar el punto 4 de la lista de componentes de la interfaz humano maquina descrita en la sección 3.3.4.6.

(a) *Resistencias 0805* **(b)** *Condensadores 0603* **(c)** *Condensador electrolítico SMD*

Figura 3.19 – *Componentes SMD propuestos en el diseño de extensión hardware.*



(a) *LED 0805* **(b)** *MOSFET con empaquetado SOT23 SMD*

Figura 3.20 – *Dispositivos SMD propuestos en el diseño de extensión hardware.*

3.4 Transmisión de datos

La cuestión de la transmisión de los datos hacia el sistema de gestión responde a las necesidades de la sección 3.1.3. De esta manera se espera llegar a un compromiso entre el tamaño de los paquetes a transmitir y la comodidad de su uso. Para ello se tendrán en cuenta dos aspectos: el formato de los paquetes y el mecanismo de comunicación.

En primer lugar, para el formato de los paquetes se proponen dos opciones.

- La primera de ella habrá de ser un lenguaje de marcado de datos estructurado que por ser estándar *de facto* será **JSON**. Se descarta utilizar XML ya que se ha quedado obsoleto frente a JSON en los nuevos paradigmas web basados totalmente en

JavaScript, así como en cuestiones de interoperabilidad entre sistemas tambien basados en JavaScript. Esta opción producirá un fichero de texto organizado como el que se muestra en la figura 3.21.

- La segunda opción de transmisión, pensando en la optimización, será un paquete binario en aritmética de punto fijo. La estructura del paquete binario se detalla en la figura 3.22 ajustada en múltiplos de 8 bits. En la tabla 3.5 indica el tamaño de cada campo del paquete binario teniendo en cuenta los rangos y la precisión de los principales datos que aparecerán.


```
var JSONpacket = {
    "Temp": 50.0,
    "Hum": 100,
    "Lum": 50000.0,
    "Vair": 20,
    "Dair": 360,
    "PosGPS": "90N/180E",
    "L90": 100,
    "L10": 100,
    "Leq": 100,
    "SerialNo": "ABCD1234",
    "Date": "16/06/16 12:58"
}
```

Figura 3.21 – *Ejemplo de paquete en formato JSON.*

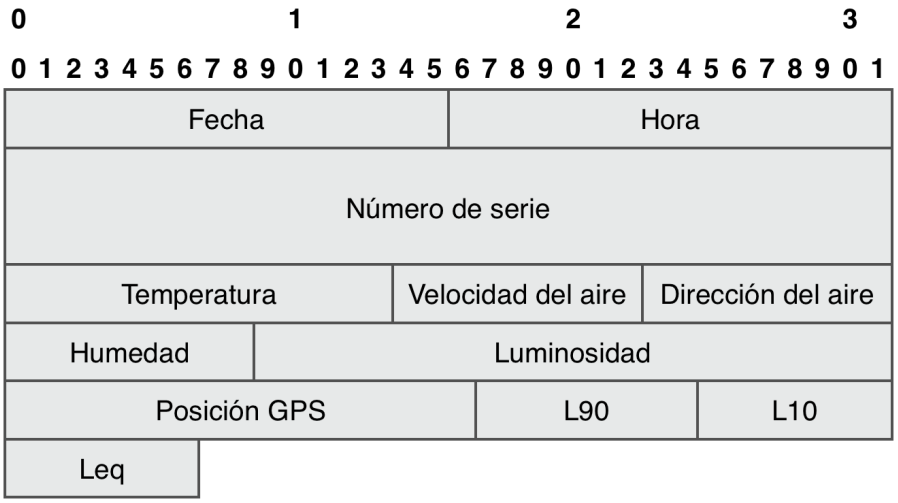


Figura 3.22 – *Ejemplo de paquete en formato binario.*

Teniendo en cuenta los tamaños aproximados de la tabla 3.5, podemos aproximar el tiempo de subida del paquete hacia el sistema de gestión de datos desde diferentes tipos de conexión a Internet. La tabla 3.6 presenta estos resultados.

Medida	Rango	Precisión	Tamaño (bits)	Tamaño JSON (bits)
Temperatura	0-50 ºC	0.1 ºC	6 + 7 = 13	32
Humedad	0-100 %	1 %	7	24
Luminosidad	0-50000 lux	0.1 lux	16 + 7 = 23	56
Velocidad del aire	0-20 m/s	1 m/s	5	16
Posición GPS	0-90 º N/S	1 "	7	16
	0-180 º E/O	1 "	8	24
Ruido (L90)	0-100 dB	1 dB	7	24
Ruido (L10)	0-100 dB	1 dB	7	24
Ruido (Leq)	0-100 dB	1 dB	7	24
Fecha	DD/MM/AA		5 + 4 + 4 = 13	64
Hora	HH:MM:SS		5 + 6 + 6 = 17	64
Numero de serie	alfanumérico	8 caracteres	64	64
Cabecera TCP			128	128
Cabecera HTTP				≈ 400
Overhead JSON				≈ 1250
TOTAL			**315**	**≈ 2500**

Tabla 3.5 – *Comparación de los protocolos propuestos.*

Tecnología	Velocidad de subida	TCP	JSON
Cellular GPRS	20 kbps	15.75 ms	125.00 ms
Cellular 3G	250 kbps	1.26 ms	10.00 ms
ADSL	1 Mbps	0.30 ms	2.50 ms
HFC + WiFi	15 Mbps	0.02 ms	0.17 ms

Tabla 3.6 – *Comparación de tiempo de subida sobre tamaños calculados.*

Como conclusión se puede decir que para sistemas nuevos las facilidades de implementación que proporciona JSON son un valor a considerar. Sin embargo, el requisito de cliente 5 hace necesario que el sistema sea compatible con otros productos similares del mercado que probablemente utilicen mensajería binaria. De esta manera se plantean **ambas opciones** para la fase de diseño y desarrollo.

Estos protocolos de mensajería deberán venir acompañados de una tecnología de transmisión. De forma natural y debido a su carácter de retro-compatibilidad, el protocolo binario utilizará la capa de transmisión TCP para enviar la información. Sencillamente un puerto de escucha TCP quedará a la espera de los mensajes que los dispositivos le transmitan.

Por su parte la transmisión de objetos JSON se puede realizar de diferentes formas, a saber:

- Sobre la capa de transporte de la pila OSI: Puertos TCP
- Sobre la capa de aplicación de la pila OSI: API HTTP REST
- Sobre el protocolo HTTP: HTTP Websockets

Por razones de seguridad y integración con el resto del sistema la implementación se realizará sobre una **API REST sobre HTTPS**.

Se descarta la utilización de HTTP Websockets debido a los ciclos de periodicidad de los mensajes, que tal y como se indicó en la sección 3.1.3 son del orden de minutos, no es necesario implementar una tecnología *full-duplex* enfocada al 'tiempo real'.

3.4.1 Compatibilidad con equipos comerciales

En cuanto a la compatibilidad con otros equipos comerciales, tal y como se ha comentado en la sección 3.4, se realizará mediante una escucha en un puerto TCP.

El protocolo *de facto* para transmisión de datos de ruido ambiental es España se conoce como ULDTP y define un conjunto de paquetes binarios. El estándar define multitud de mensaje; entre ellos están los mensajes de final de sesión o de instalación, enfocados a sistemas de limitación acústicas en locales con equipos musicales.

Sin embargo para los intereses del clientes descritos en el punto 5 unicamente es necesario tener en cuenta los mensajes relacionados con la sonometría. En la figura 3.23 se pueden observar el formato de la cabecera del mensaje (3.23a) y el cuerpo con los datos de sonometría (3.23b).

Como se ha mencionado al final sección 3.4 para estos mensajes se propone implementar en fase de diseño un puerto de escucha TCP junto con un sistema capaz de intepretar el mensaje binario.

3.5 Sistema de gestión de datos e interfaz de usuario

El sistema de gestión de datos, tal y como se detalló en la sección 3.1.2, será una plataforma que integre los servicios relacionados con la recepción y almacenaje de la información, y habrá de comunicarse con la interfaz de usuario.

Una primera consideración acerca de este subsistema pasa por delimitar el grado de integración entre la gestión de datos y la interfaz de usuario. Esta consideración es una consecuencia lógica si los datos a tratar se encuentran alojados en un servidor en Internet. Las distintas filosofías que se pueden plantear son dos:

- En una primera aproximación tradicional la interfaz de usuario o GUI será un sistema autónomo que descargue la información una sola vez al inicio de su ejecución y realice de forma local los procesos necesarios descritos en el punto 4 de la lista de requerimientos. Cuando sea necesario actualizar o modificar valores, se comunicará puntualmente con el servidor.

CABECERA DE DATOS			
Posición	**Campo**	**Longitud**	**Descripción**
0	ID	6 Bytes	Identificador del sistema: SYNKRO Tipo del campo: carácter.
6	Modelo	1 Byte	Identificador del modelo de limitador. Se asigna por el ayuntamiento. Todos los mensajes enviados por un mismo modelo de limitador deben enviar el mismo identificador: • ECUDAP: E • CAP21: C • ... Tipo del campo: carácter.
7	Número de serie	8 Bytes	Código de identificación del limitador. El número de serie de cada limitador es único y no se puede repetir. Tipo del campo Carácter
15	Tipo de Mensaje	1 Byte	Identifica el tipo de mensaje: - Instalación: i - Calibración: c - Sesión: s - Sonometría: x Tipo del campo: carácter.

(a) *Detalle de la cabecera.*

ESTRUCTURA DE LOS DATOS DE SONOMETRÍA			
Posición	**Campo**	**Longitud**	**Descripción**
16	Fecha	5 Bytes	Fecha a la cual corresponden los datos - Posición 1: día - Posición 2: mes - Posición 3: año - Posición 4: minuto - Posición 5: hora Tipo del campo: número.
21	Sesión	2 Bytes	Número de sesión en dos bytes (LSB, MSB) Tipo del campo: número.
23	L90	1 Byte	Percentil L90 en el periodo analizado. Tipo del campo: número.
24	Leq	1 Byte	Nivel Leq en el periodo analizado. Tipo del campo: número.
25	L10	1 Byte	Percentil L10 en el periodo analizado. Tipo del campo: número.
26	Lmax	1 Byte	Nivel Lmax (o percentil L1) en el periodo analizado. Tipo del campo: número.
27	Corrección del 25% de las bandas	1 Byte	Nos indica el porcentaje de tiempo en el que se corrige más del 25% de las bandas. Tipo del campo: número.
28	Corrección del 50% de las bandas	1 Byte	Nos indica el porcentaje de tiempo en el que se corrige más del 50% de las bandas. Tipo del campo: número.
29	Corrección del 75% de las bandas	1 Byte	Nos indica el porcentaje de tiempo en el que se corrige más del 75% de las bandas. Tipo del campo: número.
30	Número de Mensaje	2 Bytes	Número de mensaje de equipo enviado por el limitador en dos Bytes (valor de 0 a 65025 representado en LSB y MSB respectivamente) Tipo del campo: número.

(b) *Detalle del mensaje de sonometría.*

Figura 3.23 – *Formato de los paquetes del protocolo ULDTP.*

- En una aproximación más acorde a las tecnologías web actuales, la interfaz de usuario no será más que una aplicación que se integra completamente con el servidor y se comunica con el mediante una API HTTP REST. Esta aplicación utilizará el paradigma Modelo-Vista-Controlador y estará en constante comunicación con el servidor.

Dada la naturaleza del requisito 4 de las especificaciones del cliente, el segundo paradigma se considera más adecuado. Las razones son que mediante una implementación web de la interfaz de usuario, hacer el sistema multiplataforma se convierte en una mera metodología de trabajo (Responsive Web Design) en lugar de en un problema principal como sería implementar aplicaciones de escritorio y móviles para diferentes sistemas operativos.

En base a la decisión de realizar una aplicación web para solucionar la interfaz de usuario se proponen dos tecnologías:

- El *framework* de desarrollo rápido de aplicaciones web de Python conocido como Django.
- El *framework* de desarrollo de servidores web del entorno Node.js conocido como Express y el *framework* de aplicaciones web Angular.js

Antes de realizar una selección es importante comprender las diferentes filosofías de trabajo que proponen cada una de las opciones.

En primer lugar, Django propone una solución completa al sistema de gestión web. Para ello pone a disposición del programador de Python librerías para la utilización y gestión de bases de datos, mecanismos para gestionar los diferentes tipos de peticiones HTTP hacia una URL y un sistema de gestión de plantillas que se generan en el servidor. De esta manera el usuario, realiza una petición HTTP y se le devuelve un código HTML completo, el cual se complementa con hojas de estilo CSS y ciertos códigos JavaScript.

Por otra parte, el segundo paradigma propone tener una aplicación independiente en cuanto a código y estilo escrita utilizando el *framework* Angular.js que realizará peticiones al servidor cada vez que necesite fragmentos de la información. El servidor se implementará bajo en entorno de ejecución Node.js y la librería Express será la encargada de facilitar el servicio HTTP.

Como conclusión se pueden extraer que un sistema de gestión web con una filosofía Full Stack JavaScript permite aumentar el nivel de flexibilidad y de productividad del desarrollo. Esto se debe a que la comunicación de objetos JSON entre cliente y servidor se simplificará mucho y la ubicuidad del lenguaje JavaScript facilitará la labor de programación en ambas partes.

Tecnología	Django	Node.js + Express + Angular.js
Filosofía	Templates generadas en servidor	Aplicación MVC
Lenguaje	Python	JavaScript
Concurrencia	Un solo hilo	Multihebrado
Flexibilidad	Media	Muy alta
Autenticación	Por defecto	Passport.js
Selección	×	✓

Tabla 3.7 – *Comparación de frameworks de aplicaciones web.*

Por otro lado, Node.js y Express permiten generar un servidor multihebrado y sin bloqueos entre clientes lo cual eliminaría la necesidad de utilizar las interfaces WSGI que propone Django para integrarse con servidores web multihilo como Apache o Nginx...

Aun así, teniendo en cuenta las consideraciones de seguridad transversales requeridas por el sistema, si que se habrá de emplear un *reverse back-end web proxy*. La utilidad del mismo recaerá en poder utilizar los puertos del sistema (inferiores al puerto TCP/1024) para servir las aplicaciones e interfaces web sin necesidad de tener un proceso ejecutándose como administrador en el sistema. Este esquema de proxy se muestra en la figura 3.24

La opción del servidor web utilizado para implementar el *reverse back-end web proxy* se limita a las dos grandes opciones tradicionales: Apache y Nginx. Dado que es una decisión menor y que **Nginx** incorpora la funcionalidad necesaria sin necesidad de instalar ningún *plug-in* [21], se determina utilizar este *software*.

Una última consideración, tras haber tomado la decisión de diseño es que la autenticación de usuario se delegará, tal y como indica la tabla 3.7, a la librería **Passport.js**.

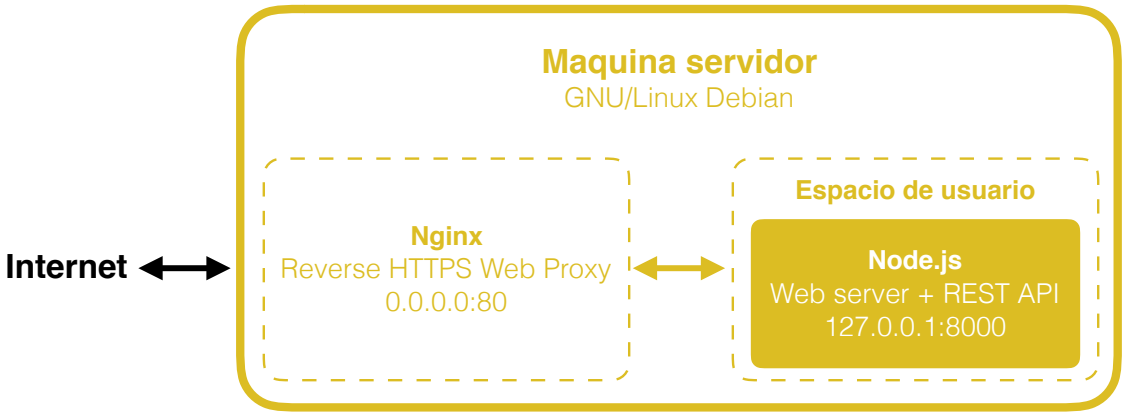


Figura 3.24 – *Arquitectura del reverse back-end web proxy.*

3.5.1 Base de datos

Una consideración más acerca del sistema de gestión de datos es el subsistema de almacenado de datos o base de datos referido en la figura 3.4. Son varias las propuestas, a saber: MongoDB, MySQL y SQLite.

En primer lugar, la justificación de la primera de las opciones MongoDB se debe a la filosofía Full Stack JavaScript. MongoDB es una base datos documental NoSQL, no estructurada, no relacional y no necesariamente ACID *compliant* [20]. El hecho de estar completamente integrada con el sistema de notación de objectos JSON permite que su integración con el resto de *stack* JavaScript sea sencillo. Sin embargo data la naturaleza relacional de los datos que esta aplicación va a tratar, esta opción no se va a tener en cuenta.

En cuanto a las otras dos opciones, la bibliografía [24] proporciona una comparativa mediante la cual podremos llegar a ciertas conclusiones. El mencionado artículo explica:

- SQLite es una base datos basada en ficheros, parcialmente compatible con SQL, ideal para testeo y desarrollo. No dispone de gestión de usuarios. No es recomendable en entornos de alto volumen de transacciones.
- MySQL es una base datos basada en servidor, parcialmente compatible con SQL, con implementaciones de sistemas de seguridad, escalable y altamente optimizada, ideal para entornos de producción. Dispone de gestión de usuarios a nivel de base de datos.

Estas consideraciones unidas al proceso de instalación y gestión de una base de datos MySQL llevan a tomar la decisión de utilizar una base de datos sencilla como es SQLite en el entorno de desarrollo para, dado el momento cuando la implementación esté en una fase avanzada y estable, realizar una migración a un entorno basado en MySQL.

3.6 Seguridad transversal

Como valor transversal del producto, tal y como se indicó en el punto 7 de las lista de requerimientos (3.1.2), subsistema de gestión, así como el de transmisión, han de ser robustos ante amenazas a la seguridad.

Idealmente todos los subsistemas que conforman el producto habrían de ser robustos frente a amenazas a la disponibilidad, confidencialidad, integridad y no repudio. Sin embargo teniendo en cuenta las posibilidades reales del diseño de este producto se tendrán en cuenta los siguientes puntos:

- La **disponibilidad** del sistema de recepción y gestión de datos se tratará de mantener el mayor tiempo sin fallas utilizando gestores de procesos que mantengan los servicios de red activos tras eventos que generen errores o fallos, así como

dimensionando correctamente las prestaciones del servidor donde se instalen los servicios. Al tratarse de una transmisión vía Internet, se agrupan demasiados agentes externos como para garantizar la transmisión de datos.

- Las amenazas a la **confidencialidad** y el **no repudio** en el *sistema de gestión* se solventarán utilizando el protocolo HTTPS y un certificado de la organización Let's Encrypt, además del sistema de autenticado propuesto al final de la sección 3.5, basado en la librería Passport.js.

- Las amenazas a la **confidencialidad** en el *sistema de transmisión de datos*, de forma similar al punto anterior, se solucionarán implementando la interfaz de transmisión de datos API HTTP REST tras un servidor que ofrezca una implementación de HTTPS. De esta manera se garantiza que nadie salvo el servidor y el equipo transmisor podrá acceder a la información contenida en el mensaje.

- Las amenazas a la **integridad** y el **no repudio** en el *sistema de transmisión de datos* se solventará mediante la utilización de una solución basada en algoritmos de firma digital. Esta solución generará un función *hash* o función resumen del cuerpo del mensaje utilizando el algoritmo SHA-2 que posteriormente será firmado con la clave privada del cliente. De esta manera, si unicamente el cliente conoce su clave privada y el servidor almacena la clave pública del mismo en el momento del registro del equipo, el sistema de recepción de información del servidor no solo podrá garantizar que el mensaje proviene del legítimo equipo emisor, si no que además podrá garantizar que el mensaje no ha sido modificado ya que dispone de una función *hash* firmada por el emisor.

La figura 3.25 representa las funcionalidades de creación de las claves pública y privada, así como el proceso publicación de llave pública durante el registro del dispositivo.

En la parte inferior se muestra el proceso de firmado y verificación del mensaje a transmitir. Este proceso garantiza la **integridad**, la **autenticidad** y el **no repudio** del mensaje. De forma complementaria y como se concluyo en la sección 3.4, el hecho de producirse la comunicación bajo un contexto HTTPS garantiza la **confidencialidad** del intercambio de mensajes.

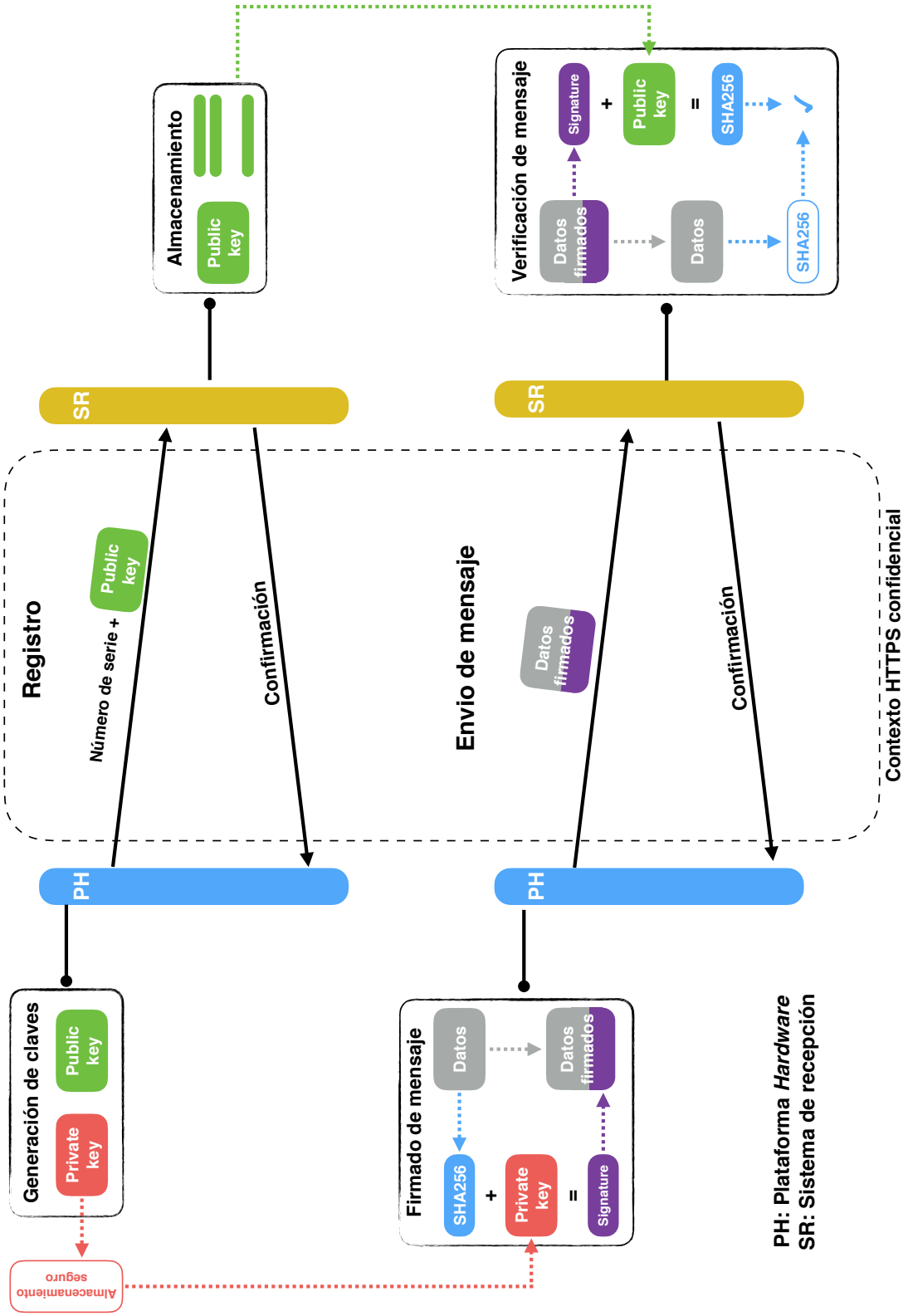


Figura 3.25 – *Infraestructura de mensajes seguros: confidencialidad, autenticación, integridad y no repudio.*

CAPÍTULO

4

DISEÑO DEL SISTEMA

En el presente capítulo se especificarán los detalles de diseño e implementación de cada subsistema descrito en el capítulo 3. En cada uno de los subapartados de este capítulo se tendrán en cuenta tanto los requerimientos extraídos en la sección 3.1, así como las consideraciones de diseño analizadas en las secciones 3.2, 3.3, 3.4, 3.5 y 3.6 acerca de la plataforma de computación, la extensión *hardware*, el sistema de transmisión, el sistema de gestión de datos, la interfaz gráfica y la securización del sistema respectivamente.

Se comenzará a detallar el diseño del sistema empezando por la extensión *hardware*, su integración con la plataforma seleccionada mediante librerías, el sistema de transmisión y su implementación de seguridad, para acabar tratando el diseño del sistema de gestión de datos y la interfaz de usuario.

4.1 Diseño de la extensión *hardware*

En primer lugar, se diseñará el producto *hardware* que solucionará el problema de la integración de sensores requeridos en la lista de la sección3.3 y seleccionados en la sección 3.3.4.

La PCB que proveerá una solución se diseñará el *software* Altium Designer 14, esta herramienta EDA permitirá realizar un diseño sobre esquemáticos electrónicos para luego proceder al diseño de la PCB y del modelo 3D del producto.

En primer lugar, el diseño esquematico del sistema presenta una solución a la interconexión

de los diferentes componentes descritos en la sección 3.3.4 con la plataforma de computación así como una solución al requisito de cliente número 2 mediante una fuente de alimentación conmutada.

En las siguientes páginas se puede observar el resultado del diseño de estos esquemáticos tal y como quedaron en la versión final del proceso de diseño. Esta iteración final es la última fase de un proceso de desarrollo y mejora iterativo que constó de siete versiones.

Una vez completado el diseño de los esquemáticos se procede a al diseño de la PCB. El diseño de la PCB vuelve a ser un proceso iterativo que generalmente consta de dos partes: el *placing* y el *routing*.

El proceso de ***placing*** es aquel en el cual se colocan los componentes sobre una superficie delimitada de forma que se optimice dicho espacio y se plantee una solución lógica a la interconexión de elementos. A continuación en el proceso de ***routing*** se interconectan los componentes mediante lineas eléctricas siguiendo unas reglas de diseño.

Esta reglas de diseño responden a las limitaciones que existen en el proceso de fabricación conocido como *milling* o fresado. Dichas limitaciones surgen de las características de fabricación expuesta en la lista 3.3.2. Entre otras cosas, es imprescindible tener en cuenta las limitaciones de la herramienta de fresado automático o CNC LPFK ProtoMat S62 mostrada en la figura 4.1.

Figura 4.1 – *Herramienta de fresado automático LPFK ProtoMat S62.*

En general, esta consideraciones sobre las limitaciones del proceso de fabricación se han de ver reflejadas en ambos procesos: el ***routing*** y el ***placing***. Las principales reglas a tener en cuenta son las que aparecen en la lista de a continuación. Algunos de los conceptos tratados

se muestran de forma gráfica en la figura 4.2:

- **Pistas**: El ancho de las pistas idealmente será del mayor tamaño posible para evitar comportamientos resistivos, poniendo especial atención en las dedicadas a la alimentación del sistema.

- ***Clearance***: Por *clearance* se entiende el mínimo espacio entre dos pistas adyacentes. Dada la naturaleza del proceso de *milling* o fresado, este espacio mínimo viene determinado por la diámetro mínimo del conjunto de útiles de fabricación. Para el caso que nos atañe, se considera la fresa de menor diámetro a utilizar de 10 mils, de esta manera el *clearance* entre pistas y elementos eléctricos diferentes habrá de ser de 10 mils o mayor.

- ***Drills***: Se conoce como *drill* a todo aquel agujero que atraviese el sustrato completo sobre el que se fabrica la PCB. Los diametros posibles para dichos agujeros vienen determinados por los útiles de perforación disponibles, a saber: 0.6, 0.7, 0.8, 0.9, 1, 1.1, 1.3 y 1.5 mm. En este diseño se utilizarán perforaciones de 0.6 mm para las vías o elementos de unión entre las dos caras de la PCB debido a que, como se decidió en la lista 3.3.2, existe la posibilidad de metalización de *drills* en este proyecto. Para el resto de elementos THT se utilizará la medida necesaria para satisfacer las caracteristicas del elemento. En cuanto a los *drills* de diametro mayor a 1.5 mm se utilizará un útil de corte para realizar la perforación.

- ***Annular ring***: El concepto *annular ring* hace referencia a la corona que es necesario producir entorno a un *drill* THT o una vía no metalizada que necesite ser soldada. Estas coronas serán de 70 mils para facilitar la soldadura cuando sea necesario.

- **Capa de texto**: Al no tener disponibles tecnologías de serigrafiado, se utilizará la propia tecnología de fresado para generar textos relevantes sobre la PCB.

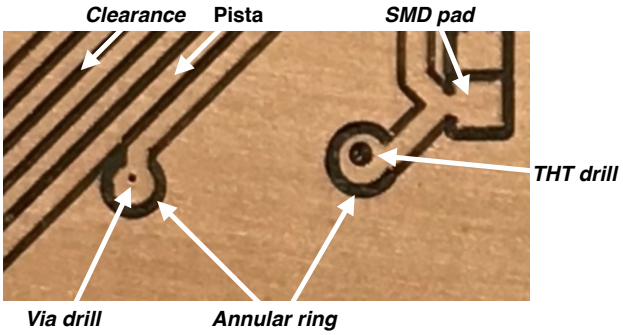
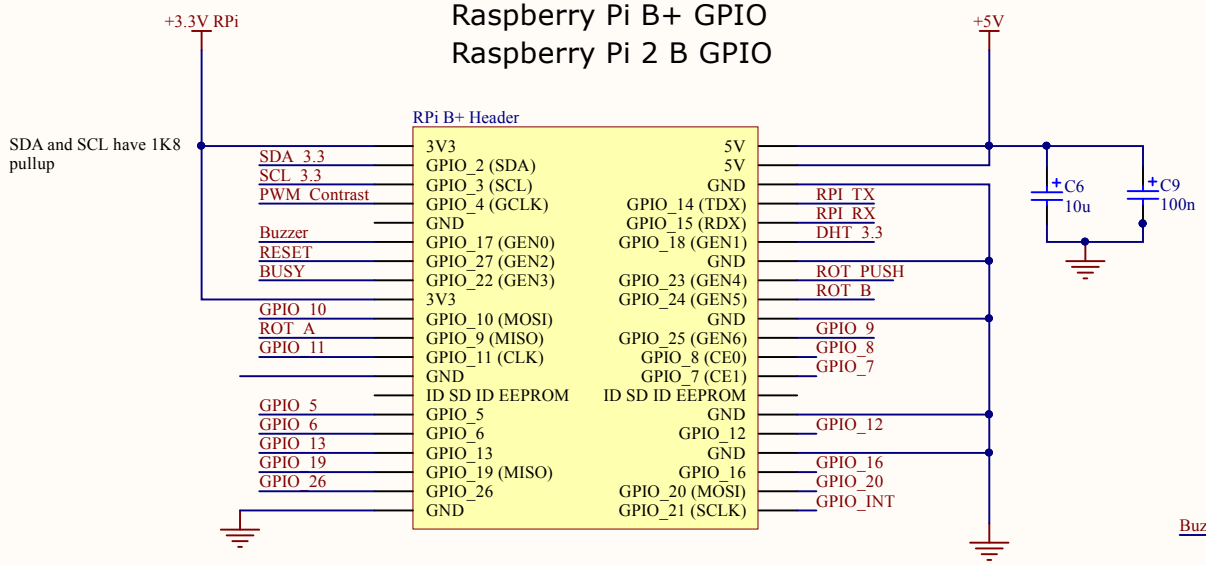
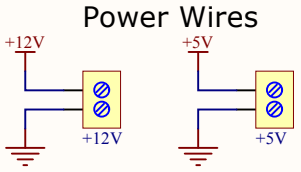
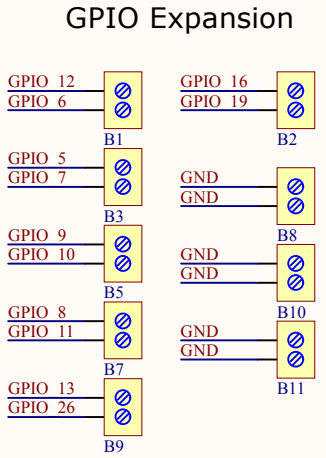


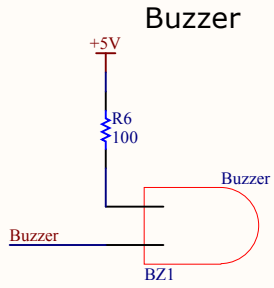
Figura 4.2 – *Conceptos relacionados con las limitaciones de fabricación de la PCB.*

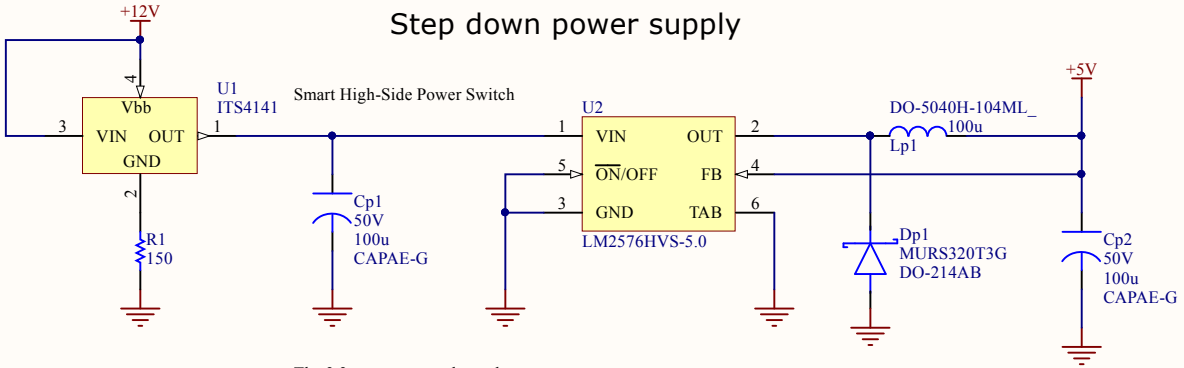
En las siguientes páginas se puede observar los diseños de las PCB tanto en su capa *bottom* como en su capa *top* tras haber solventado iterativamente todos los errores relacionados con las reglas anteriormente mencionadas.

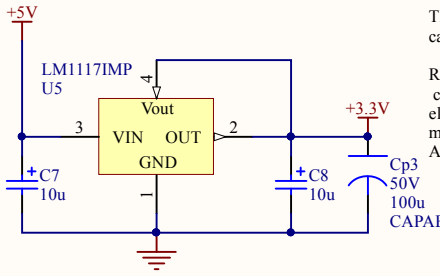






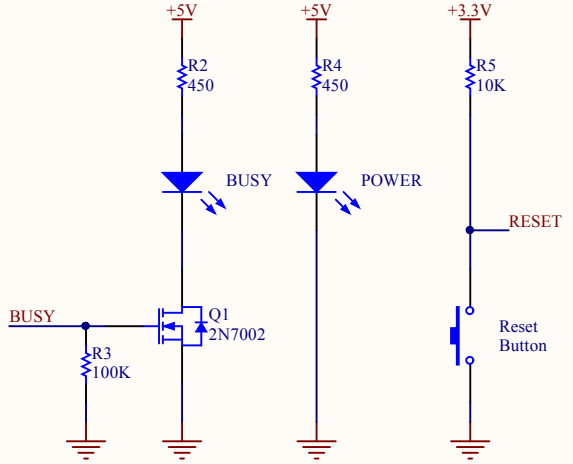






The 3.3 converter need tantalum capacitors

Reparte el condensador por la piste: ceramico (100n) en paralelo con electrolitico (cuanto mas grande mejor 5V o 16V)
AL lado de cada integrado

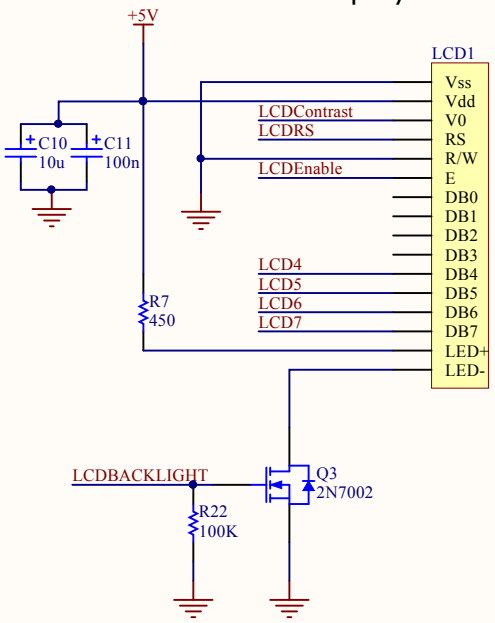


Designer's signature	Sheet title: **dBAHAt Schematic 2**		
	Project title: **dBAHat_V2.PrjPcb**		
Supervisor's signature	Desginer: **Pablo Garrido Sánchez**		
	Date: **20/09/2015**	Revision: **Version HW 1.0**	Sheet 1 of 3

Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain
Sr. Andrés Roldán Aranda

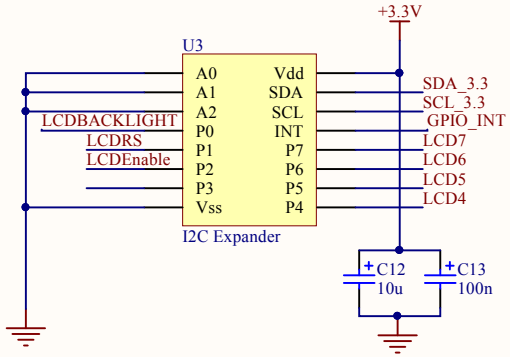
heimdal

LCD Display

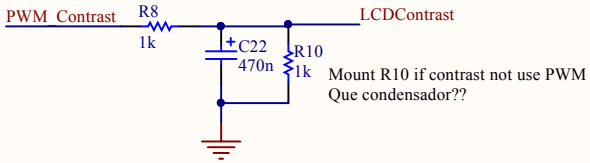


Check if R7 is in LCD board
To I2C
http://www.instructables.com/id/LCD-display-I2C-adapter-for-Arduino-with-PCF8574A/
Poner electrolitico entre 5V y gnd cerca del power del lcd por si pistas muy largas en paralelo otro de 100 nF ceramico

LCD I2C Expander

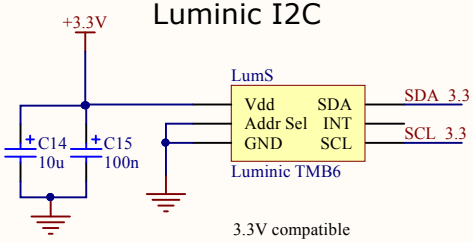


PWM LCD Contrast



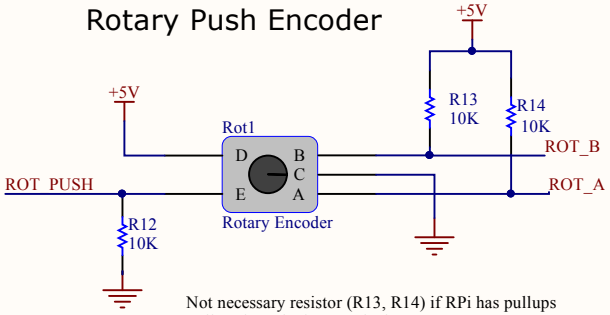
Mount R10 if contrast not use PWM
Que condensador??

Luminic I2C



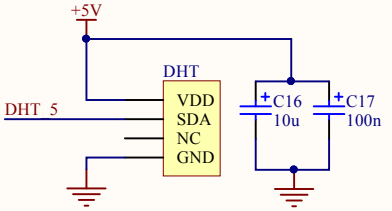
3.3V compatible

Rotary Push Encoder



Not necessary resistor (R13, R14) if RPi has pullups
Pull-up is 50 kOhm - 65 kOhm
Pull-down is 50 kOhm - 60 kOhm

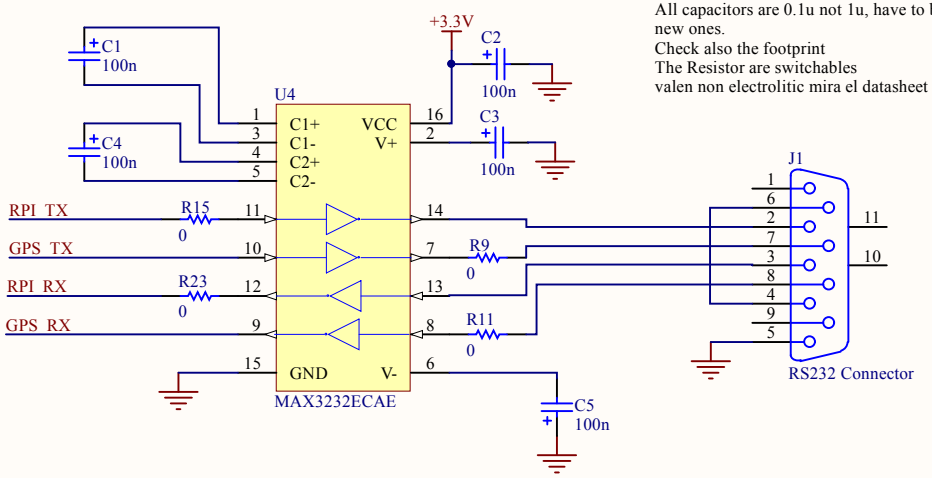
Temperature and Humidity Sensor



Designer's signature	Sheet title: **dBAHAt Schematic 2**		*Dpto. Electrónica y Tecnología de Computadores* *University of Granada* *C/ Fuente Nueva, s/n, 18001* *Granada, Granada, Spain* *Sr. Andrés Roldán Aranda*
	Project title: **dBAHat_V2.PrjPcb**		
Supervisor's signature	Desginer: **Pablo Garrido Sánchez**		
	Date: **20/09/2015**	Revision: **Version HW 1.0**	Sheet 2 of 3

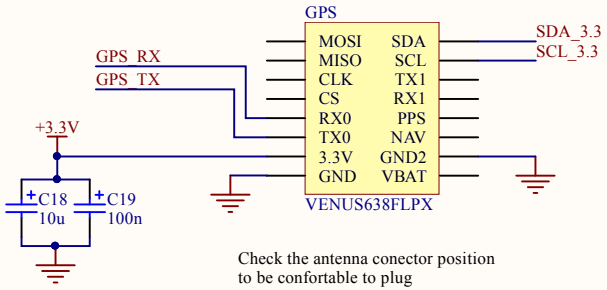
heimdal

UART to RS232 Interface



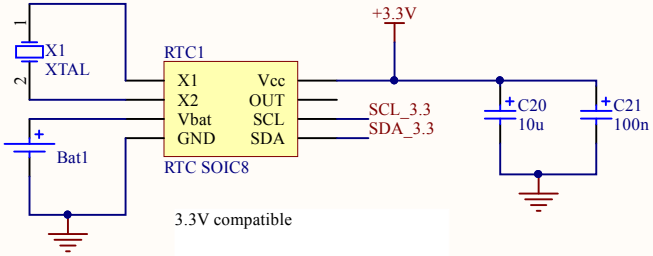
Check the RS232 connections.
All capacitors are 0.1u not 1u, have to buy new ones.
Check also the footprint
The Resistor are switchables
valen non electrolitic mira el datasheet

GPS Module I2C



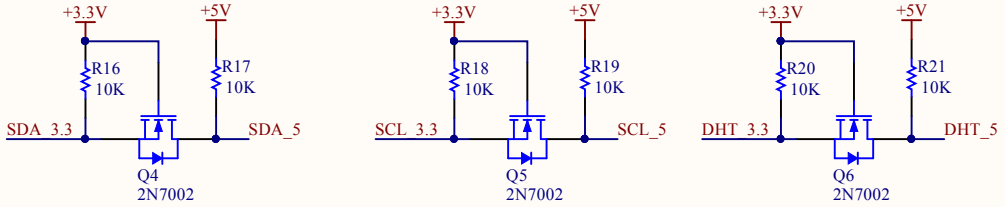
Check the antenna conector position to be confortable to plug

RTC I2C



3.3V compatible

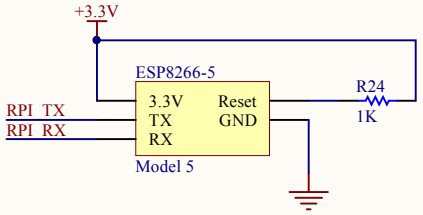
Level Converters



I2C Levels are always 3.3, this converters are not used

The original trans is BSS138, 2N7002 is used here.
Check that.
Ref:
https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide
https://cdn.sparkfun.com/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf

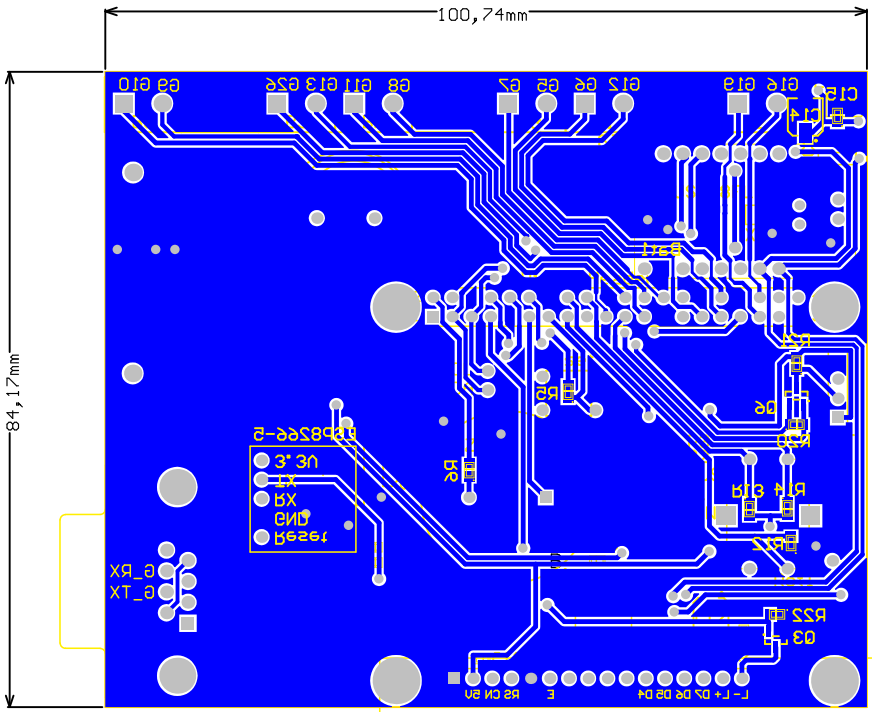
ESP8266 UARt to Wifi



Designer's signature	Sheet title: **dBAHAt Schematic 2**		
	Project title: **dBAHat_V2.PrjPcb**		
Supervisor's signature	Desginer: **Pablo Garrido Sánchez**		
	Date: **20/09/2015**	Revision: **Version HW 1.0**	Sheet 3 of 3

Dpto. Electrónica y Tecnología de Computadores
University of Granada
C/ Fuente Nueva, s/n, 18001
Granada, Granada, Spain
Sr. Andrés Roldán Aranda

heimdal

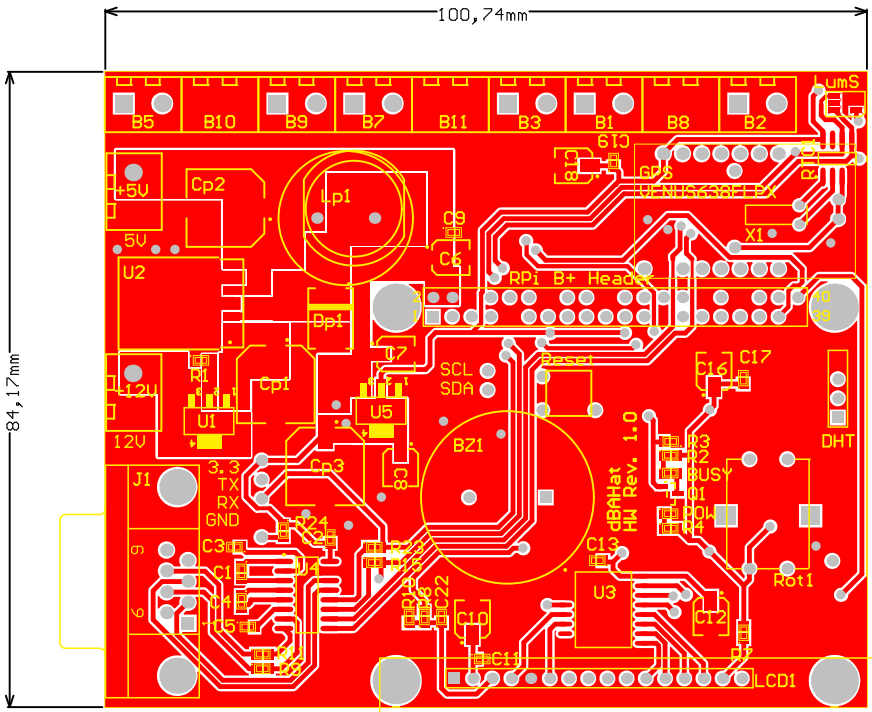


dBAHat

Description:
This board will be the main support for the sensors in the Raspberry Pi's HAT

Raspberry Pi 2 Model B screws included.

Designer's signature:	Sheet title: dBAHAT DV 5.0	Dpto. Electronica y Tecnologia de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andres Roldan Aranda
	Project title: dBAHAT	
Supervisor's signature:	Designer: Pablo Garrido Sanchez	
	Supervisor: Andres Roldan Aranda	Date: 02.09.2015 Revision: 6



dBAHat

Description:
This board will be the main support for the sensors in the Raspberry Pi's HAT

Raspberry Pi 2 Model B screws included.

Designer's signature:	Sheet title: dBAHAT DV 5.0	Dpto. Electronica y Tecnologia de Computadores University of Granada C/ Fuente Nueva, s/n, 18001 Granada, Granada, Spain Sr. Andres Roldan Aranda
	Project title: dBAHAT	
Supervisor's signature:	Designer: Pablo Garrido Sanchez	
	Supervisor: Andres Roldan Aranda	Date: 02.09.2015 Revision: 6

Algunos otros aspectos a tener en cuenta en el proceso de diseño y fabricación de la PCB son:

1. Se aprovecha que el integrado de conversión de niveles MAX3232 descrito en la sección 3.3.4.5 dispone de dos puertos para conectar uno de ellos al puerto UART de la plataforma Raspberry Pi 2 Model B y otro al puerto de comunicación serie del módulo Venus GPS de Sparkfun descrito en la sección 3.3.4.4.

2. Se considera diseñar la fuente de alimentación de entre 12 y 30 V de entrada hacia 5 V y 3.3V de salida.

3. La fuente de alimentación se basa en un diseño de referencia proporcionado por Texas Instruments [10] sobre el regulador DC-DC *step-down* conmutado LM2576. Este integrado convertirá una tensión de alimentación de hasta 40 V en 5 V regulados, ambos en corriente continua.

4. Una segunda fase de la fuente de alimentación basada en el integrado LM1117 de Texas Instruments convertirá los 5 V de corriente continua de la fase anterior en un nivel estable de 3.3 V. Del mismo modo, se trata de un regulador DC-DC *step-down* conmutado.

5. Las pistas que conforman la fuente de alimentación en el diseño de la PCB se dimensionan, dentro de lo posible como planos con el fin de evitar comportamientos resistivos en los mismos. Las distribución de dichas tensiones de alimentación se implementan en una disposición de árbol de forma que se generen las mínimas superficies sobre las que se puedan inducir corrientes espurias o generar interferencias electromagnéticas con otros dispositivos. Estas dos consideraciones se enfocan hacia generar un dispositivos que cumpla, en el futuro, las normas y estándares EMC.

6. El bus I2C se diseña de forma que recorra todos los dispositivos a interconectar utilizando el camino más corto. Dado el *placing* definitivo de los módulos, este orden será: plataforma de computación Raspbery Pi, módulo RTC (3.3.4.3), sensor de luminosidad (3.3.4.2) y expansor GPIO (3.3.4.6).

7. Para implementar el control de la retroiluminación de la pantalla LCD descrito en la sección 3.3.4.6, es necesario implementar un interruptor digital utilizando un transistor MOSFET controlador mediante un pin GPIO de la plataforma de computación. Se puede observar esta configuración en el módulo nombrado como *LCD Display* de la segunda página de los esquemáticos anteriormente expuestos.

8. De forma similar al punto anterior, para controlar el contraste de la pantalla LCD es necesario conectar el pin de contraste del dispositivo a un nivel de tensión que será proporcional al contraste producido. Este nivel de tensión se generará mediante un filtro paso baja alimentado por una señal PWM procedente de un pin GPIO de la plataforma de computación. Este montaje se puede observar en la sección titulada *PWM LCD Contrast* de la segunda página de los esquematicos anteriormente expuestos.

Un detalle importante a mencionar dentro del punto 5 de la lista inmediatamente anterior es que para la realización de las lineas de alimentación de la placa es necesario tener en cuenta los consumos de cada uno de los dispositivos que conforman el subsistema.

Para ello en la tabla 4.1 se realiza un resumen de los consumos habituales de cada uno de los subsistemas que conforman el *hardware*. A partir de dichos consumos, y dada su tensión de alimentación y el número de apariciones de dicho componente en la PCB, se obtendrá la potencia consumida expresada en mW. Como se puede observar en la mecionada tabla, el consumo total del sistema, a partir de los datos obtenidos de los diferentes *datasheets*, será como máximo de, aproximadamente, 3.85 W.

En este punto hay que considerar que la plataforma Raspberry Pi es el dispositivo con mayor consumo (1.75 W), sin embargo, el valor indicado responde a una situación de carga computacional intensiva. En casos de procesos ligeros corriendo en el sistema operativo, el consumo se reducirá drásticamente.

Por otra parte, la retroiluminanción LED de la pantalla LCD consume 1.2 W, se ha de tener en cuenta que no siempre permanecerá encendida. De esta manera se reduce el consumo medio de dispositivo sin dejar de tener en cuenta los consumos máximos posibles durante el dimensionado de la fuente de alimentación.

Elemento	Consumo (mA)	Cantidad	Voltaje (V)	Total (mA)	Potencia (mW)
RPi	350.00	1	5.00	350.00	1750
Buzzer	0.00	1	3.30	0.00	0
LEDs	25.00	2	5.00	50.00	250
Button	0.00	1	3.30	0.00	0
LCD	3.00	1	5.00	3.00	15
LCD LED	240	1	5.00	240.00	1200
DHT11	1.00	1	5.00	1.00	5
GPIO Expander	121.21	1	3.30	121.21	400
Luminic	0.24	1	3.30	0.24	0.792
Rotary	1.00	1	5.00	1.00	5
MAX3232	0.30	1	3.30	0.30	0.99
Venus GPS	68.00	1	3.30	68.00	224.4
RTC	1.00	1	3.30	1.00	3.3
TOTAL	**810.75**			**835.75**	3854.482

Tabla 4.1 – *Potencia consumida por cada unos de los componentes del subsistema hardware.*

Este detalle junto con los condensadores de alimentación existentes físicamente junto a cada uno de los dispositivos conforman el diseño completo de la alimentación del subsistema *hardware*. Estos condensadores se seleccionan con los siguientes propósitos:

- Los condensadores electrolíticos 10 µF permitirán mantener la tensión frente a fluctuaciones lentas en la alimentación debido a picos de consumo del dispositivo.
- Los condensadores cerámicos de 100 nF permitirán inmunizar la alimentación del dispositivo frente al ruido en la señal de alimentación.

Continuando con la descripción, durante el proceso de diseño de la PCB se incluyen los modelos 3D de cada uno de los componentes, así como de la plataforma de computación Raspberry Pi 2 Model B. De esta manera se obtiene una primera aproximación de como será el resultado de la fabricación. La figura 4.3 muestra el resultado de este renderizado 3D y en la figura 4.4 se pueden observar detalles y diferentes vistas.

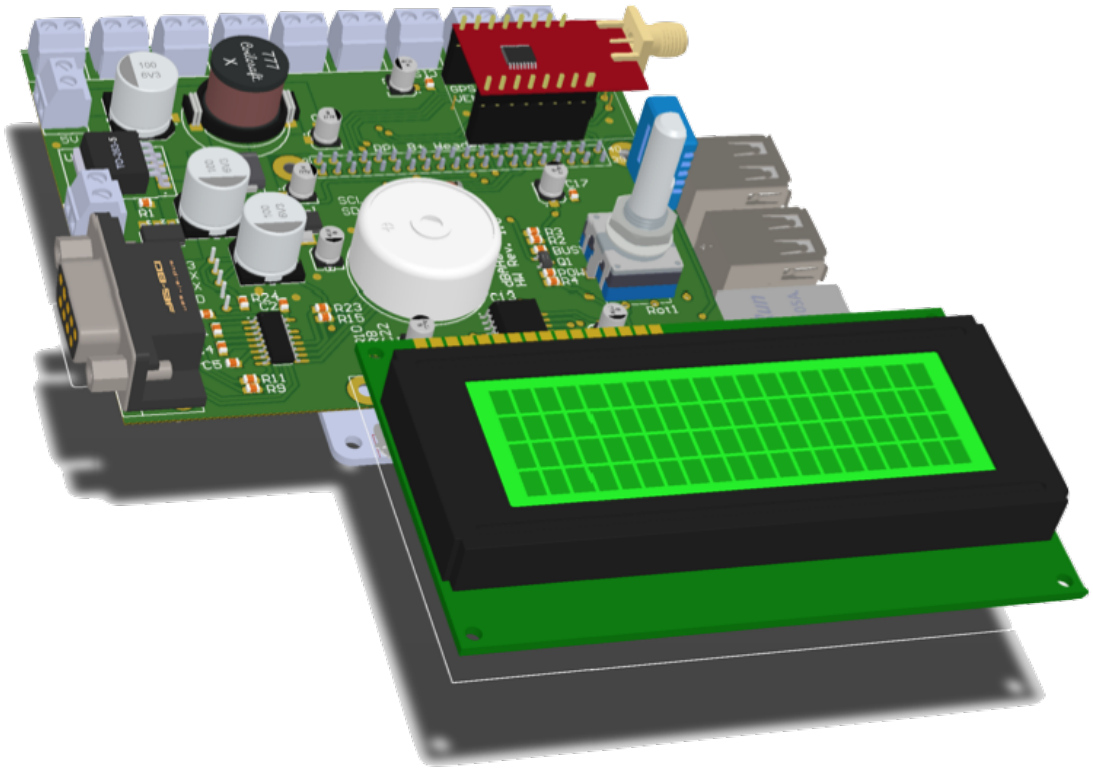
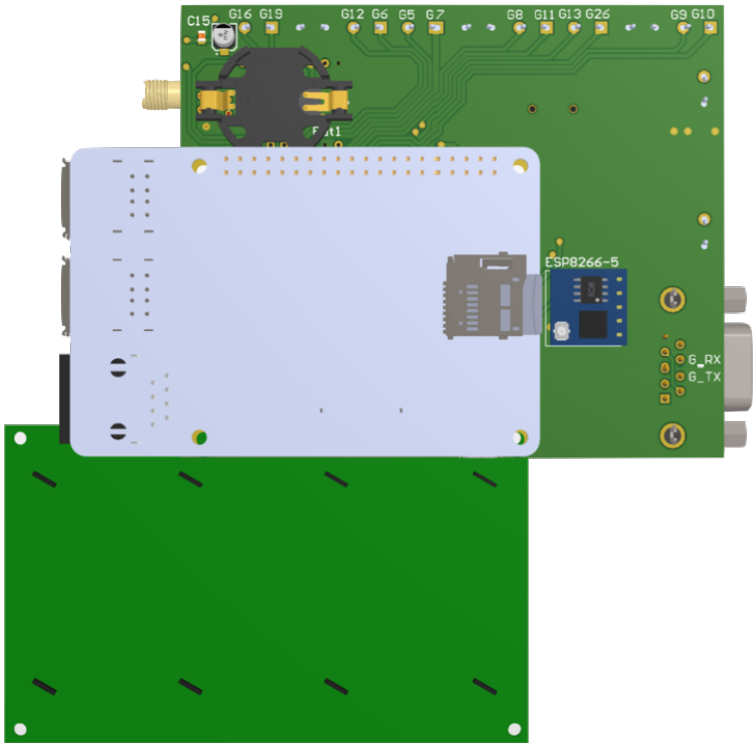
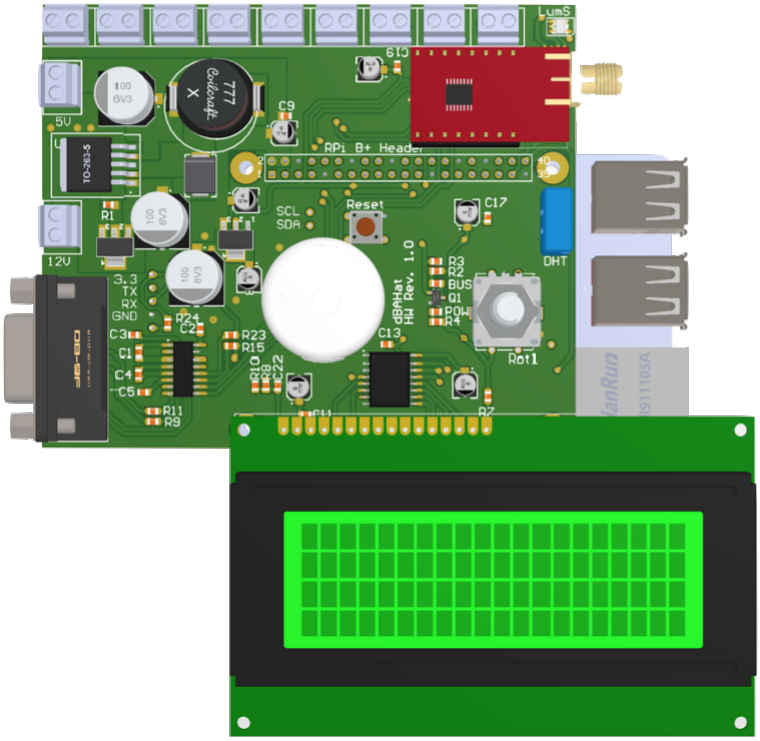


Figura 4.3 – *Modelo 3D de la PCB.*

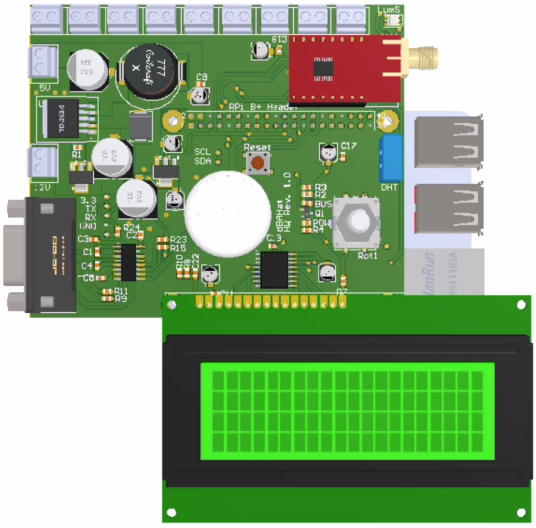


(a) *Capa bottom*



(b) *Capa top*

Figura 4.4 – *Modelo 3D de la PCB. Vistas alternativas.*



Video 4.1 – *Video de la PCB.*

4

Una vez validado el diseño esquemático, el diseño de la PCB y el modelo mecánico 3D se procede a la fabricación del prototipo.

La figura 4.5 se observa un detalle del proceso de fabricación por extracción mediante fresado o proceso de *milling* utilizando la CNC ProtoMat S62 de la marca LPFK. El resultado de dicho proceso, tanto en la capa *top* como en la *bottom*, se observa en la figura 4.6.

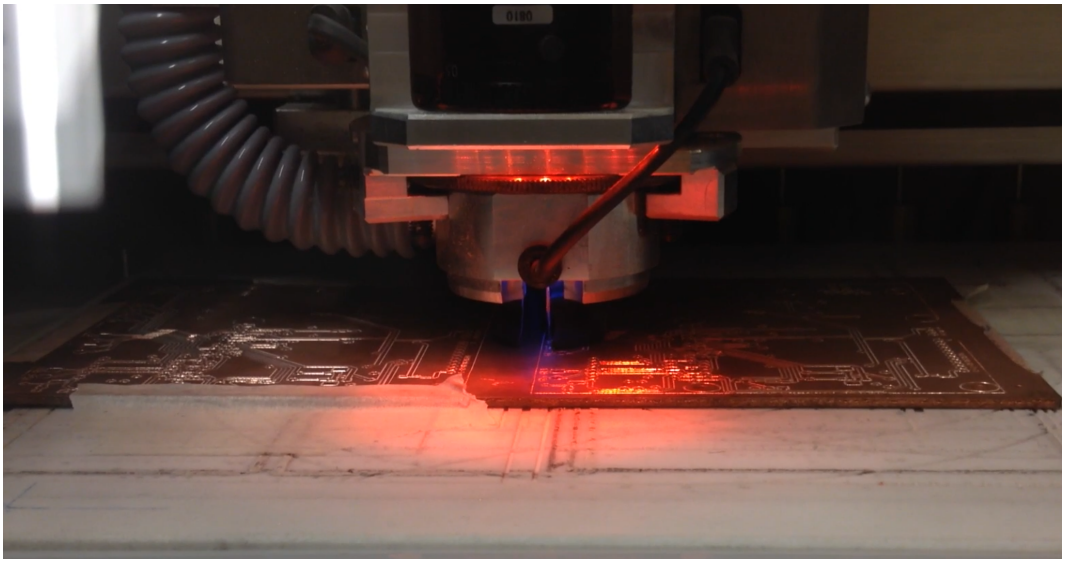
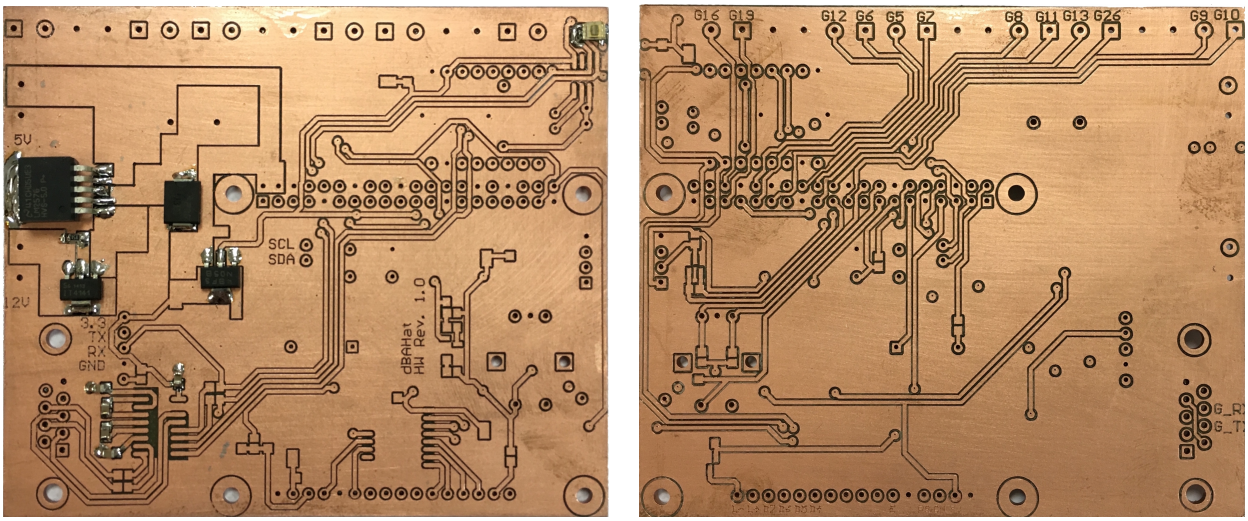


Figura 4.5 – *Detalle de la fabricación.*



(a) *Capa Top* **(b)** *Capa Bottom*

Figura 4.6 – *Resultado de fabricación de la PCB.*

Cabe mencionar que tras el proceso de fresado es necesario barnizar la PCB completa, antes de comenzar la soldadura con barniz a base de colofonía y acetona. Este barniz, ademas de proteger del oxido al prototipo, facilita la soldadura.

Tras el secado del barniz, se procede a soldar todos los componentes THT así como los SMD. Es conveniente comprobar la conectividad de las pistas utilizando un multímetro digital antes y después de la soldadura de cada elemento para tratar de detectar posibles cortocircuitos o ausencia de conectividad en vías.

Una vez completada la fabricación del dispositivo, tal y como se muestra en la figura 4.7, se procede al montaje del conjunto de componentes hardware que conforman el producto completo, a saber:

- Plataforma de computación Raspberry Pi 2 Model B.
- Módulo Venus GPS de SparkFun con conector SMA.
- Tarjeta SD con 8 GB de almacenamiento.
- Módulo LCD desmontado.
- Subsistema *hardware* con todos los componentes SMD y THT soldados.
- Tornillos de anclaje.

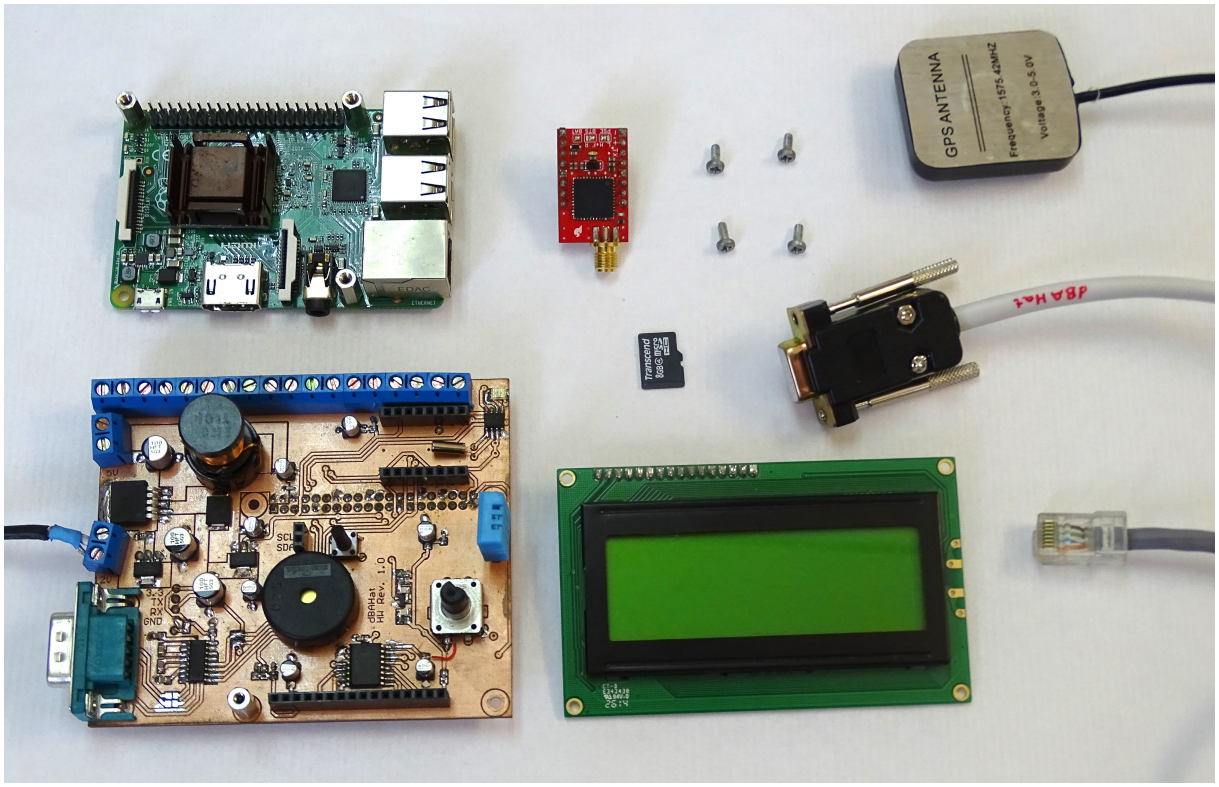


Figura 4.7 – *Componentes del sistema hardware.*

4.2 Librerias *software*

Tal y como se indicó en la la sección 3.3.3, una vez conformado el sistema *hardware* es necesario implementar un conjunto de librerías software en Python 2.7 para el control de periféricos y la transmisión de datos hacia Internet.

El sistema operativo instalado en la plataforma de computación es Raspbian Jessie Lite [12], una versión reducida (sin GUI ni paquetes extra), adaptada al *hardware* de la plataforma Raspberry Pi y compilado para ARM v7 de Debian 8 *Jessie*. Además de dicho sistema operativo es necesario tener disponibles ciertos paquetes y realizar ciertas configuraciones quer permiten el correcto funcionamiento del *software* que controlará la PCB de expansión.

Los pasos a seguir para configurar el sistema antes de ejecutar los procesos de lectura de sensores son:

1. Descargar e instalar Raspbian Jessie Lite en la tarjeta SD mediante un *software* de copiado de imagenes como ApplePi-Baker [6].
2. Conectar por SSH con la plataforma Raspberry Pi mediante el comando (en Linux): *ssh pi@<ip>* utilizando la contraseña *raspberry*.
3. Cambiar la contraseña utilizando la utilidad *raspi-config*.
4. Ampliar el espacio de usuario mediante la utilidad *raspi-config*.

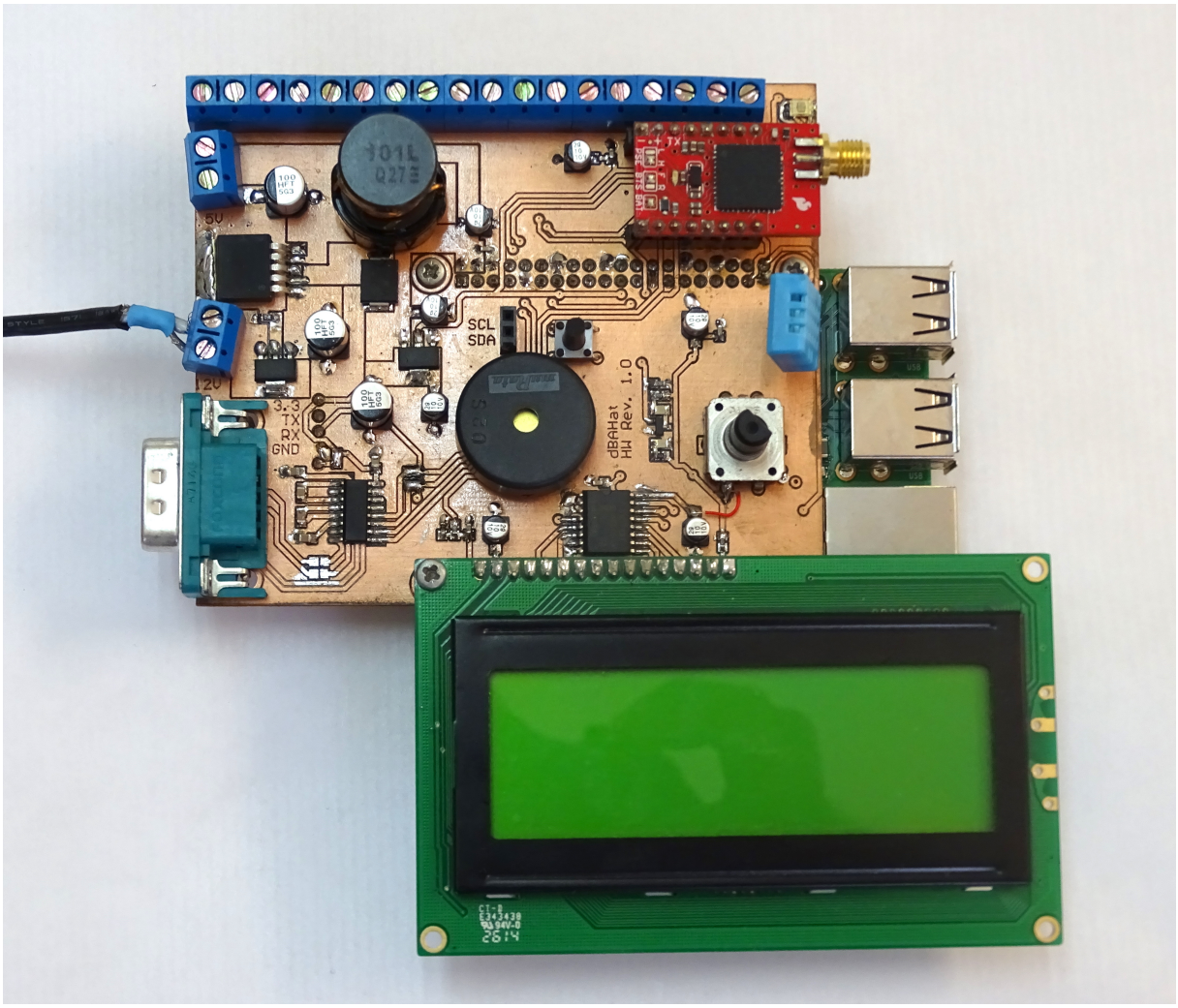


Figura 4.8 – *Sistema hardware ensamblado.*

5. Instalar las herramientas de gestión del bus I2C mediante el siguiente tutorial: https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c.

6. Compruebar el correcto funcionamiento del bus I2C mediante el siguiente comando: *sudo i2cdetect -y 1*. En general deben aparecer tres dispositivos (referidos mediante sus direcciones del bus): 29 (sensor de luminosidad TSL2561T), 38 (extensor GPIO utilizado para la comunicación con el LCD) y 68 (RTC M41T81).

7. Cambiar el puerto serie en *sudo raspi-config*. Mediante esta opción se permite tener el puerto UART disponible para la interacción con periféricos en lugar de utilizarse como terminal del sistema Linux subyacente.

8. Instalar el paquete de gestión del puerto serie de Python: *sudo apt-get install python-serial*.

9. Conectar en el puerto serie un adatador que interconecte el bus serie del sistema con el bus serie del GPS y comprobar que ambos se comunican con el comando: *miniterm.py /dev/ttyAMA0* .

10. Instalar las utilidades de gestión de repositorios: *sudo apt-get install git*.

11. Instalar los paquetes de desarrollo de Python: *sudo apt-get install python-dev python-pip*.

12. Instalar paquetes adicionales para gestión de módulo de Python: *sudo apt-get install python-setuptools*.

13. Instalar las librerías WiringPi de control de GPIO para Pyhton y Raspberry Pi: *sudo pip install wiringpi2*.

14. Instalar libreria de control de del sensor de humedad y termperatura DHT11 mediante el siguiente enlace: `https://learn.adafruit.com/dht-humidity-sensing-on-raspberry-pi-with-gdocs-logging/software-install-updated`.

15. Instalar las librerias de control de GPIO PIGPIO con soporte para el control de la LCD por I2C desde el siguiente enlace `http://abyz.co.uk/rpi/pigpio/download.html`.

16. Copiar la carpeta *HWTest*. Dentro de ella se podrá encontrar un test para comprobar que la extensión *hardware* esta funcionando correctamente. En el siguiente enlace se puede observar este test en funcionamiento: `https://www.youtube.com/watch?v=Sw-Hk6kAX1w`

17. Copiar la carpeta *dBAStation*. Dentro de ella se encuentran los ficheros y librerías que implementan el funcionamiento de la estación de monitorización ambiental. La estructura de esta carpeta se puede observar en la figura 4.10.

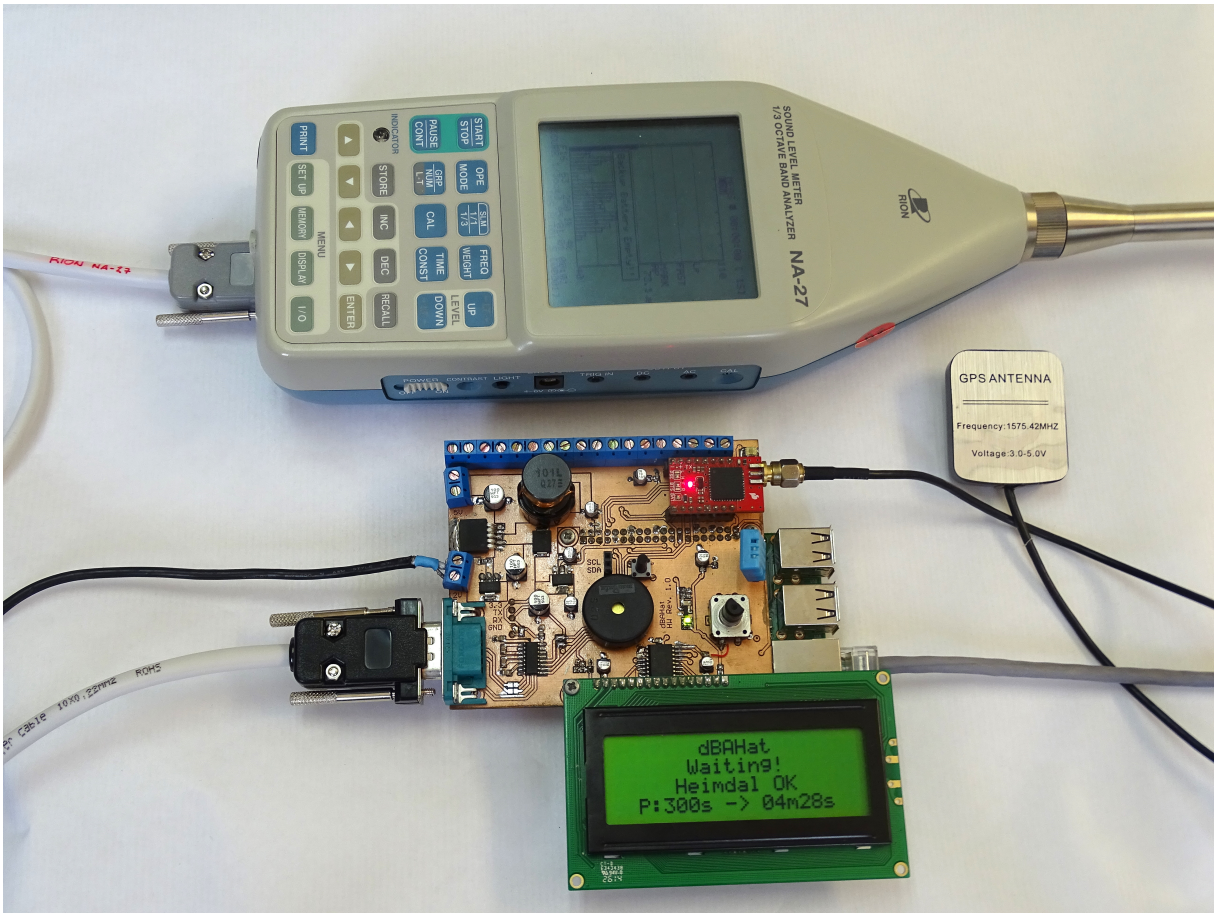


Figura 4.9 – *Sistema hardware completo en funcionamiento.*

Una vez instaladas todas las dependencias, el *software* contenido en las carpetas de los últimos puntos conforman el conjunto de librerías necesarias para el control de periféricos alojados en la extensión *hardware*.

```
/
└── dBAStation
    ├── AdafruitI2C.py
    ├── Buzzer.py
    ├── LCD.py
    ├── keys
    │   ├── public.pem
    │   └── private.pem
    ├── RTC.py
    ├── TSL2561.py
    ├── SPL.py
    └── client.py
```

Figura 4.10 – *Arbol de directorios del software*

El programa **client.py** es el proceso principal que ejecutará un ciclo de lectura, transmisión, guardado y espera, tal y como indica la maquina de estados representada en la figura 4.11. Este proceso principal consiste en un ciclo de lectura y transmisión, así como en gestión de eventos. Al inicio del sistema operativo la utilidad *crontab* [26] se encargará de ejecutar este programa.

El proceso de guardado consistirá en un fichero con formato CSV que almacenará todas las mediciones realizadas por el producto junto con una marca temporal.

En primer lugar varias funciones será invocadas como hilos dentro del proceso principal. Estas funciones o procesos Daemon realizarán las tareas de comunicación por puerto serie con el módulo GPS, comunicarse por el puerto serie con el sonómetro digital, gestionar los eventos usuario relacionados con el *rotary-push* y gestionar los eventos de usuario relacionados con el botón de apagado.

Entrando en cierto detalle, el hilo que se comunica con el GPS actualizará una variables global con la geoposición que se esté leyendo. El hilo relacionado con el sonómetro digital leerá datos sonométricos y variables de ruido ambiental cada vez que exista una en dicho bus y de la misma manera se las comunicará al proceso principal.

Por su parte el control del *rotary-push* permitirá modificar el periodo de transmisión de datos hacia el sistema de gestión y su función como botón permitirá forzar la transmisión de un paquete.

El botón de apagado se encargará de cerrar los procesos de forma correcta y enviar una señal de apagado al resto del sistema. De esta manera el cierre del sistema operativo será adecuado y no habrá problemas de corrupción del sistema de archivos de la tarjeta SD.

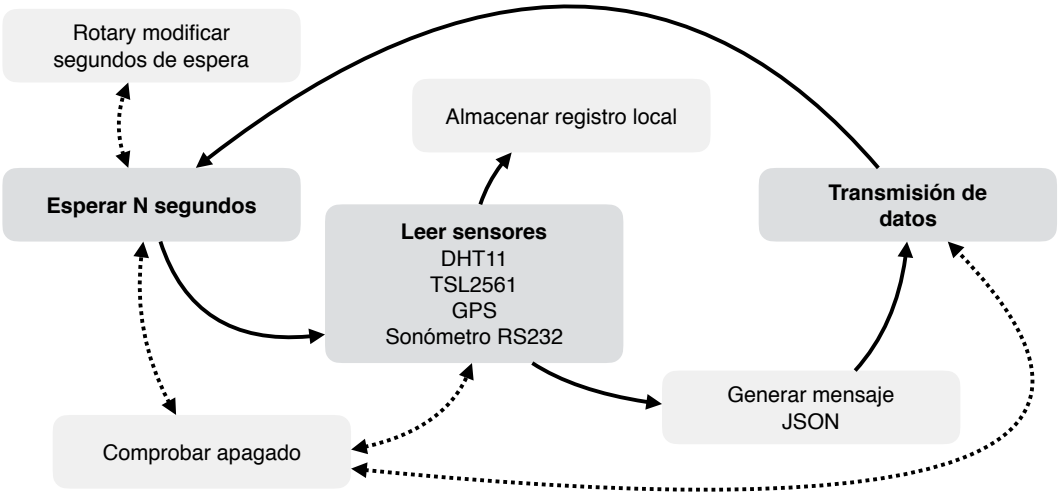


Figura 4.11 – *Maquina de estados del software.*

4.2.1 Implementación de librerías de lectura del sonómetro

Para finalizar la sección de diseño de librerías software es importante poner atención en la librería que controla el proceso de adquisición de datos principal de este proyecto: la de los datos de ruido acústico.

Como se explicó en la sección 3.3.4.5, este dispositivo implementará la lectura de datos de un sonómetro de Clase 1 de la compañía RION, concretamente el modelo RION NA-27.

Siguiendo con el lenguaje utilizado hasta el momento, esta utilidad será programada en Python y constará de dos fases: la configuración de dispositivo y la adquisición de datos periódicamente.

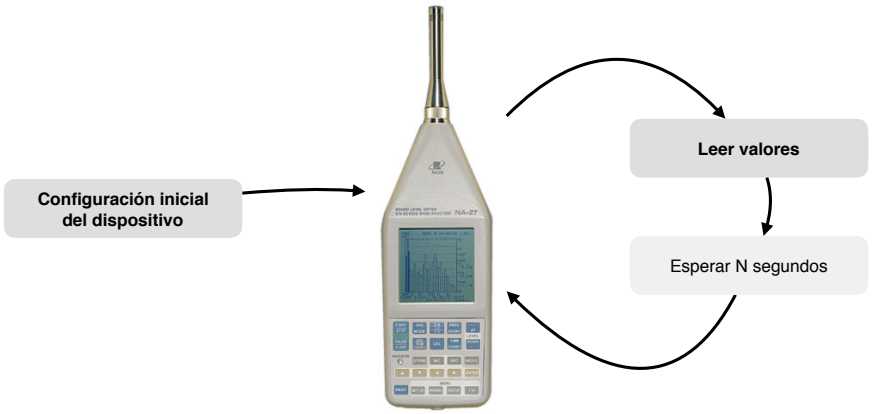


Figura 4.12 – *Ciclo de configuración y lectura del sonómetro RION NA-27.*

En el código 4.1 se puede observar la estructura y la metodología utilizada para comunicar el producto con el sonómetro RION NA-27 mediante el lenguaje Python.

```
1  import time
2  import serial
3  from datetime import datetime
4
5  class RIONNA27:
6
7      #Init object and serial port
8      def __init__(self, p, debug):
9          self.ser = serial.Serial(
10             port=p,
11             baudrate=9600,
12         )
13         self.debug = debug
14         self.ser.isOpen()
15
16     #Message encoding based on RION documentation
```

```
17      def encodeMessage(self, command):
18          packet = bytearray()
19          packet.extend([0x02, 0x01, 0xFE])            #Header
20          packet.extend(command)                       #Payload
21          packet.extend([0x1A] * (32 - len(command))) #Padding
22          packet.extend([sum(packet[3:]) % 256])       #Checksum
23          return packet
24
25      def setParam(self, cmd):
26          self.ser.write(self.encodeMessage(cmd))
27          while self.ser.inWaiting() < 1:
28              pass
29          out = self.ser.read(1)
30          if self.debug:
31              print ''.join('{:02x}'.format(x) for x in bytearray(out))
32
33          #Check if ACK have been received
34          if ''.join('{:02x}'.format(x) for x in bytearray(out)) == "06":
35              return True
36          else:
37              return False
38
39      def askParam(self, cmd):
40          cmd = cmd + " ?"
41
42          self.ser.write(self.encodeMessage(cmd))
43          if self.debug:
44              print "Sending " + cmd
45
46          while self.ser.inWaiting() < 1:
47              pass
48
49          out = self.ser.read(1)
50
51          s = ''.join('{:02x}'.format(x) for x in bytearray(out))
52
53          #Check if ACK have been received
54          if s == '06':
55              if self.debug:
56                  print "ACK Received"
57                  print "Sending NACK (21)"
58
59              #Send NACK
60              self.ser.write(bytearray([21]))
61
62              while self.ser.inWaiting() < 1:
63                  pass
64
65              #Read packet length
66              out = self.ser.read(1)
67              tam = ''.join('{:02x}'.format(x) for x in bytearray(out))
68              if tam == "01":
```

```
69                  while self.ser.inWaiting() < 131:
70                      pass
71                  out += self.ser.read(self.ser.inWaiting())
72              else:
73                  while self.ser.inWaiting() < 35:
74                      pass
75                  out += self.ser.read(self.ser.inWaiting())
76
77              time.sleep(0.1)
78
79
80              while True:
81                  if self.debug:
82                      print "Sending ACK (6)"
83                  #Send ACK
84                  self.ser.write(bytearray([6]))
85
86                  while self.ser.inWaiting() < 1:
87                      pass
88
89                  outend = ''
90                  while self.ser.inWaiting() > 0:
91                      r = self.ser.read(1)
92                      out += r
93                      outend += r
94
95                  if self.debug:
96                      print "Readed: " + str(len(outend))
97                      print "String: " + str(outend)
98                      print "Hex: " + s
99
100                 c = ''.join('{:02x}'.format(x) for x in bytearray(
                     outend))
101                 #If EOT found, finish communication
102                 if c == "04":
103                     if self.debug:
104                         print "EOT Found"
105                     break
106
107         #Return payload without padding
108         return out[3:].replace("\x1A", "")
109
110
111 #Open device
112 r = RIONNA27("/dev/tty.usbserial", 0)
113
114 # Configure date and time
115 r.setParam("CLK " + datetime.now().strftime("%y %m %d %H %M %S"))
116
117 # Configure SPL mode
118 r.setParam("IMD 0")
119
```

```
120 # Configure statistical calculation:
121 # L10, L90, Lmax and Leq
122 r.setParam("MSR 1")
123 r.setParam("STT 0 0 1 0 1 0 0 1 0 1")
124
125 # Configure calculation time to 5 min
126 r.setParam("SMT 5 1")
127
128 # Configure group mode
129 r.setParam("AUT 1 # #")
130
131 while true:
132     # Start 5 minutes measurement
133     r.setParam("SRT 1")
134
135     # Wait until SRT command says: measurement finished
136     print "Measuring..."
137     while r.askParam("SRT").split(",")[1][0] == "1":
138         time.sleep(1)
139
140     # Retrieve the results of the measurement
141     ans = r.askParam("DOD 0")
142
143     # Display the results
144     print "Waveform peak: " + ans.split(",")[9] + " dB"
145     print "LAmax: " + ans.split(",")[10] + " dB"
146     print "LAeq: " + ans.split(",")[12] + " dB"
147     print "LA10 " + ans.split(",")[14] + " dB"
148     print "LA90: " + ans.split(",")[16] + " dB"
```

Código 4.1 – *Implementación del sistema de extracción de datos del sonómetro RION NA-27*

4.3 Base de datos

Antes de comenzar a escribir cualquier código relacionado con el sistema de gestión y recepción de datos es imprescindible definir el sistema del almacenamiento del que el sistema hará uso. La estuctura diseñada es la que muestra la figura 4.13.

Este sistema de base de datos responde a las necesidades expuestas durante la fase extracción de requerimientos en la sección 3.1.2 y 3.1.2.1. La tecnología utilizada para la implementación de este subsistema es SQLite por las razones que se expusieron en la sección 3.5.1.

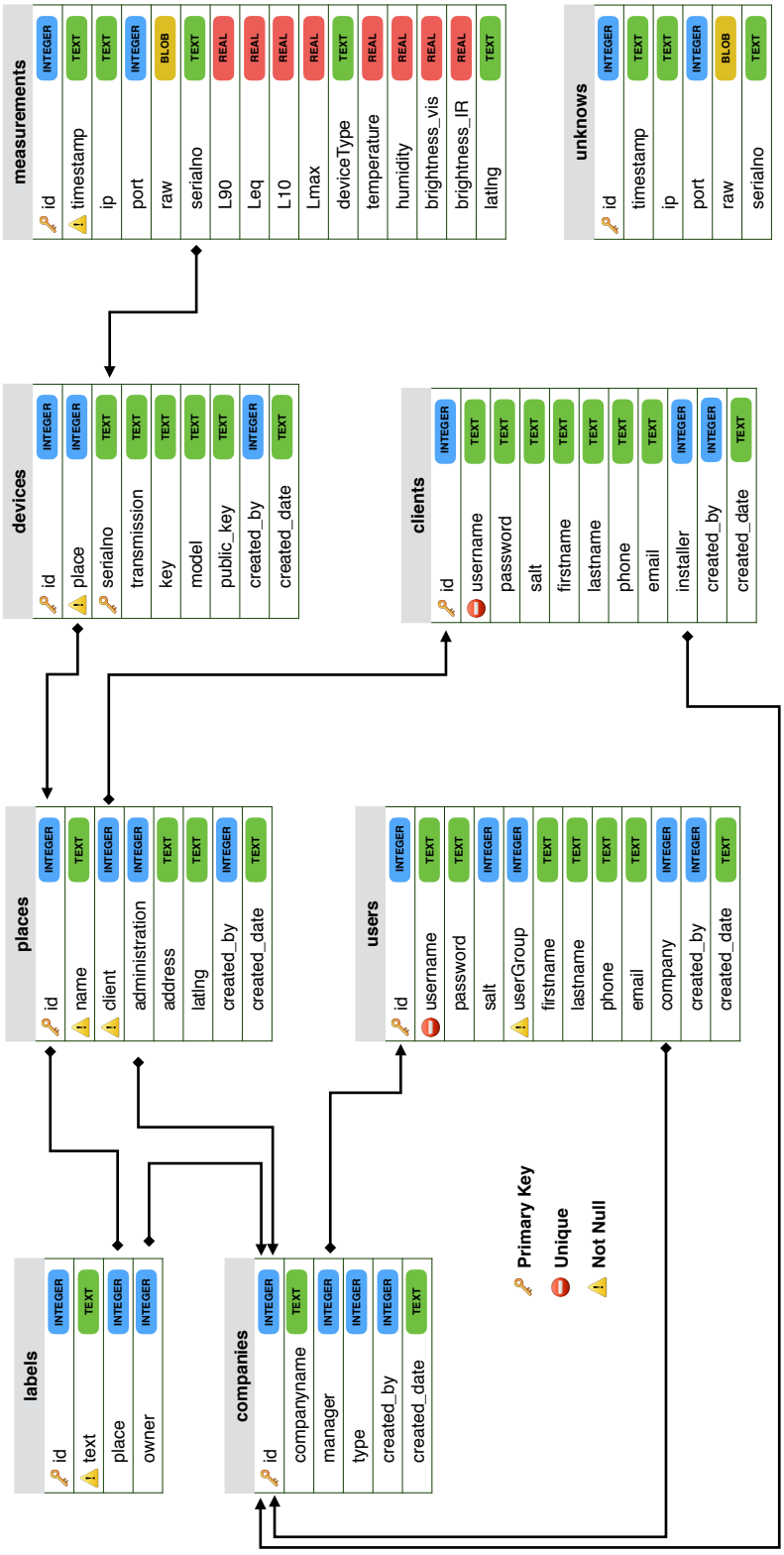


Figura 4.13 – *Esquema de implementación de la base de datos.*

4.4 Sistema de transmisión

En la parte de la derecha de la figura 4.11 se puede observar el proceso de generación y transmisión del mensaje en formato JSON.

Sin embargo el proceso de diseño tratará de integrar los procedimientos descritos en las secciones 3.4 y 3.6. Para ello se ha de definir un formato de paquete y una implementación de transmisión con las siguientes características:

- La información a transmitir habrá de estar incrustada en un paquete con formato JSON.
- El mensaje habrá de transmitirse de forma confidencial mediante el método POST sobre HTTPS.
- El mensaje habrá de estar firmado digitalmente mediante criptografía asimétrica para garantizar integridad y no repudio.
- El emisor del mensaje habrá de poder registrarse una única vez en el sistema antes de comenzar su funcionamiento habitual. En este registro se habrá de transmitir el número de serie y una clave pública hacia el sistema de gestión de datos, ambas serán almacenadas indefinidamente y garantizará que se cumplan lo objetivos del punto anterior.

En la figura 4.14 se puede observar el procedimiento de registro y transmisión de paquetes seguro propuesto para el sistema. En esta figura se unen la maquina de estados propuesta en la figura 4.11 con el modelo de transmisión segura de la figura 3.25.

En primer lugar al arrancar el sistema tratará de localizar el par de claves público-privada RSA en la carpeta */keys* (observar figura 4.10), se pueden dar dos situaciones:

1. El sistema **no** encuentra un par de claves en dicha carpeta. Se procederá entonces a la generación de un par de claves RSA de 2048 bits de longitud, a almacenar dichas claves en formato PEM en la carpeta */keys*, y por último transmitirlas a la dirección del servidor de gestión de datos. Dicha dirección será https://<*url*>/deviceRegister. El mensaje transmitido incluirá el numero de serie y la clave pública generada, tal y como se puede observar en la figura 4.16, y en caso de ser un registro válido, la clave se asociará en la base de datos SQLite al dispositivo que la ha transmitido (véase figura 4.13).
2. En caso de encontrar un par de claves en la carpeta */keys* el sistema supondrá que se encuentra registrado en el sistema de gestión y procederá a ejecutar la maquina de estados de la figura 4.11.

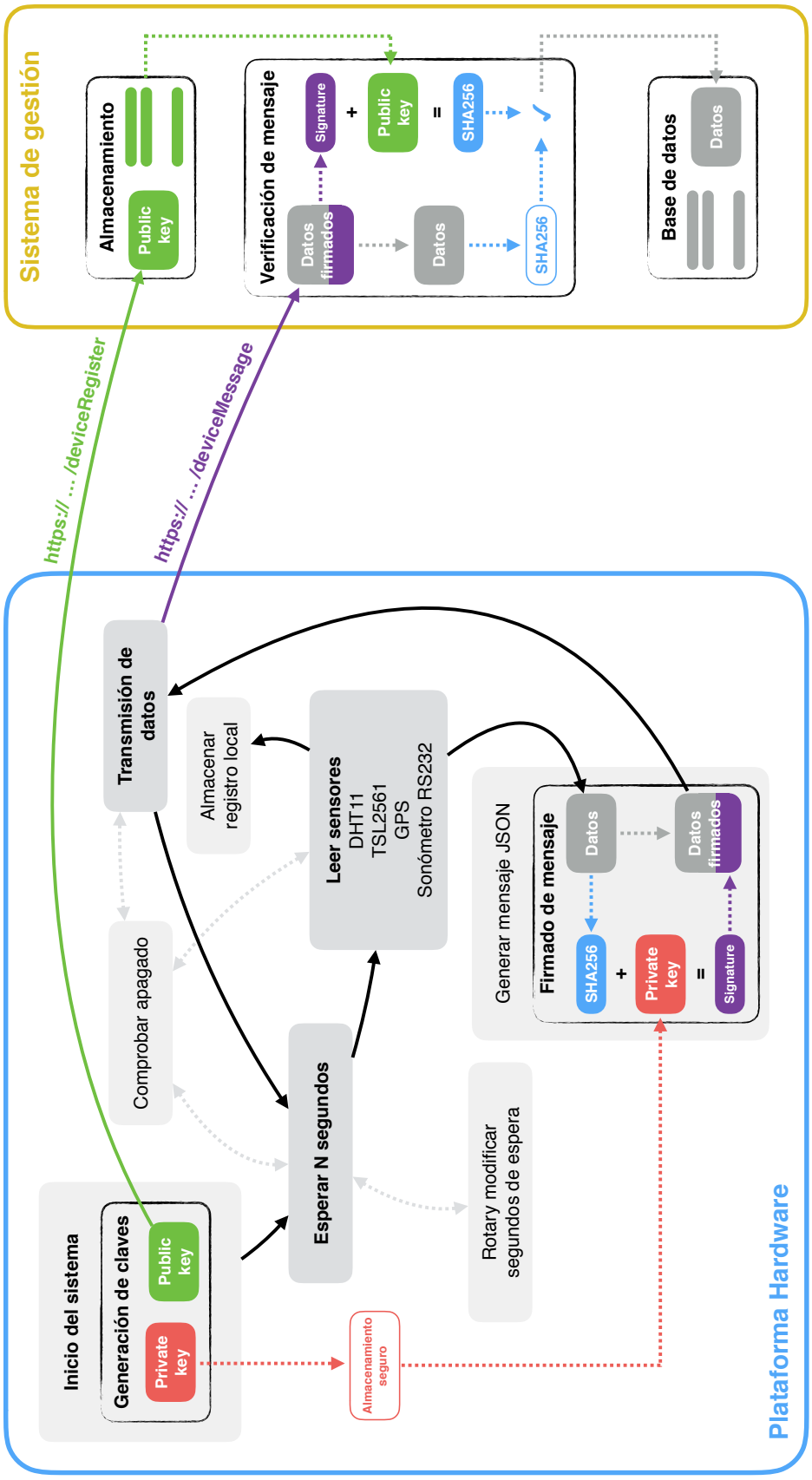


Figura 4.14 – *Infraestructura de registro y transmisión junto con el funcionamiento de la máquina de estado de la plataforma hardware*

Tras realizar uno de estos dos procesos, el *software* transmitirá cada cierto tiempo mensajes con el formato indicado en la figura 4.15 hacia la dirección https://<*url*>/deviceMessage. Este mensaje incluye un campo *data* con los datos a transmitir y almacenar, y otro campo *sign* que incluye la firma digital. Esta firma está generada con el mecanismo expuesto en la figura 3.25.

El sistema de recepción, verificará la firma digital y considerará como validos aquellos que estén relacionados con la clave pública provista en el proceso de registro para un numero de serie concreto.


```
{
    "data": {
        "hum": 24,
        "tem": 35,
        "aco": [27.5700700786, 27.7746005137, 28.1779100839, 28.5495266537],
        "serialno": "DBASTATION1",
        "lum": [11.379199999999997, 2368, 1344],
        "gps": [37.179715, -3.6088233333333335]
    },
    "sign": "ruQuqxPmaokUod903thqEj5c/TZxLdAayLUJrYFRtoVNOL8MEDxO2krwSzobntiv
    58aWd7Y6VrA+eH/GFJzNpxGGouAV8L979F3LBIVY2mclpmy+tH+N6SwbwCisQdjyy2zAidtxJ/
    gn3TtRI3FCTTKsGgdP4VztfQPtm6RY2aniFZEZZs5fM5MR8l9LveKEuHX/8AZDYWUi5GoHPLXI
    xdDHMVkjiyzso9kFLAevtDY+zq5ET6nFkRk8LmwLvfCIZoW7gCJOrg6P3akZDM9y1mfLTAC5L0
    yDCf/FBZuJT5gzAdyWPmZ62Kn/lbDQpUdRLr8tsKbftYz4/4hCwVsjVA=="
}
```


Figura 4.15 – *Mensaje JSON de datos con campos de seguridad incluidos.*


```
{
    "seriano": "DBASTATION1",
    "public_key": "-----BEGIN PUBLIC KEY-----
        MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAv / 7 iClYGGHFClEcafs6f
        QiX2UEti4Ccjg / gK2cb6JCH7F4oSKYfGqOSEBvkqyTFar / Ou05771zlz1SjjdXdT
        WDan8kJ1lOBZ9USl / Yy5C1kmJliW42d / kl5lmJhUpVMDaT7QdFS7Bl5 + O18b1t3B
        xnydYHYYGHLwjrC / UYoGRR1mFyX1sQInnUu / cQXnZZGaQwghsNkqDsW26Jp2HmS7
        tLIDLbAYp0Afdmu6oqpcFEmue7OijJarD9JtZ5dak0mj4gA16JelD0F85aGmOARo
        d1pN3u / lqeivMxAmVv10 / sprTFEWIthEG7po4UeQbAiP + I5lGZ8D0Z8SMApvig1p
        LwIDAQAB
        -----END PUBLIC KEY----- "
}
```


Figura 4.16 – *Mensaje JSON de registro de dispositivos.*

El proceso de firmado digital de los mensajes se implementa en las librerías que gestionan la plataforma de computación y por tanto dicha implementación será en el lenguaje de alto nivel Python. Para ello se utilizará la librería PyCrypto.

La verificación de la firma digital en el subsistema de recepción de mensajes se implementará en Node.js. El paquete Express.js se encargará de implementar la interfaz HTTP y de gestionar las llamadas al verbo POST contra las direcciones https://<*url*>/deviceRegister y https://<*url*>/deviceMessage. Por otra parte, el paquete *crypto* de Node.js se encargará de realizar la verificación de la firma digital mediante la firma recibida y la clave pública que se ha almacenado en la base de datos SQLite.

Otra consideración a tener en cuenta antes de continuar es el aumento del tamaño de los datos debido a la firma digital utilizada. Teniendo en cuenta la columna referida al formato JSON de la tabla 3.5, se tuvieron en consideración unos 2500 bits. A este número se han de añadir aproximadamente 2500 bits (2048 de la firma utilizando RSA de 2048 bits más cabeceras). De esta manera se puede reescribir la tabla 3.6 de tiempos de transmisión de la manera que se muestra en la tabla 4.2: teniendo en cuenta paquetes de unos 5000 bits.

Tecnología	Velocidad de subida	JSON
Cellular GPRS	20 kbps	250.00 ms
Cellular 3G	250 kbps	20.00 ms
ADSL	1 Mbps	5.00 ms
HFC + WiFi	15 Mbps	0.33 ms

Tabla 4.2 – *Comparación de tiempo de subida sobre tamaños calculados teniendo en cuenta la firma digital.*

Por último se muestra en el código 4.2 la implementación en Node.js y Express.js del sistema de recepción y registro de los dispotisivos en el sistema de gestión. En este fragmento aparecen dos funciones, que responden a las peticiones POST de almacenado de datos y de registro respectivamente.

```
var express = require('express');
var app = express();

//MESSAGES ENDPOINT
app.post('/deviceMessage', function(req, res) {

    // Extract and separate data and cryptographic signature
    var crypto = require('crypto');
    var verify = crypto.createVerify('RSA-SHA256');
    var d = JSON.parse(JSON.stringify(req.body));
    var sign = new Buffer(d.sign, 'base64');
    delete d.sign
    verify.write(d.data);
    verify.end();
    var message = JSON.parse(d.data)

    //Searching for the device that is trying to save data
    db.get("SELECT * FROM devices WHERE serialno = ?", message.serialno
     , function(err, row) {

        //If device exists and is registered
        if (row && row.public_key) {

            //Verify the cryptographic signature
            if (verify.verify(row.public_key, sign)) {
                //Insert the data in the database
                db.run("-SQL DATA INSERTION-", [data], function(err) {
                    console.log("HTTP Measure saved as " + this.lastID)
                    res.json({status: "OK"})
```

```
29                  });
30              //Not valid cryptographic signature
31              } else {
32                  res.json({error: "Signature Error"})
33              }
34          //Nonexistent or not registered device
35          } else {
36              res.json({error: "Non valid serialno"})
37          }
38      })
39  });
40
41  //REGISTRATION ENDPOINT
42  app.post('/deviceRegister', function(req, res) {
43      var data = req.body
44
45      //Search the device that is trying to save data in the system
46      db.get("SELECT * FROM devices WHERE serialno = ?", data.serialno,
         function(err, row) {
47
48          //If device exists
49          if (row) {
50              //If device is not registered
51              if (row.public_key == null) {
52                  db.run("-SQL PUBLIC_KEY INSERTION-",data.public_key,
                     function(err) {
53                      res.json({status: "OK"})
54                  })
55              //Device already registered
56              } else {
57                  res.json({error: "Registered device"})
58              }
59          //Nonexistent or not registered device
60          } else {
61              res.json({error: "Non valid serial number"})
62          }
63      })
64  });
65
66  //Start the webserver in a certain TCP port
67  app.listen(WEBPORT, HOST);
```

Código 4.2 – *Implementación en Node.js y Express.js del sistema de recepción y registro*

4.4.1 Compatibilidad con ULDTP

Por último, respecto al sistema de transmisión y recepción de los mensajes de medición de ruido acústico y variables ambientales, cabe mencionar la implementación del sistema de compatibilidad con sistemas comerciales que soluciona el punto 5 de la lista de requisitos de cliente y que fué tratada en la sección 3.4.1.

Esta implementación se realizará en JavaScript dentro del entorno de ejecución Node.js, acorde a la propuesta de implementación del *backend* realizada en la sección 3.5.

Dicha implementación, mostrada en el código 4.3, no es mas que una interpretación binaria de los datos del paquete de la figura 3.23 utilizando Node.js y la apertura de un puerto de escucha en TCP/12000 para permitir la conexión de los dispositivos que cumplan el estándar ULDTP.

Este caso concreto trata de decodificar los mensajes ULDTP transmitidos por la serie CAP21 de limitadores acústicos con funciones de medición de ruido ambiental de la marca dBElectronics [13].

```
1  var net = require('net');
2  var path = require("path");
3  var dateFormat = require('dateformat');
4  var pad = require("left-pad")
5
6  //Parsing raw data message
7  function parseMessage(messageData) {
8      var data = {}
9
10     var modelCAP = messageData.toString("utf-8", 0, 6)
11     var type = messageData.toString("utf-8", 15, 16)
12
13     //Check if CAP21 string is in header and if X is the type
14     if (modelCAP == " CAP21" && type == "X") {
15
16         //Extract the serial number
17         data.serialno = messageData.toString("utf-8", 7, 15)
18
19         //Extract the message timestamp
20         data.date =
21         pad(parseInt(messageData.toString("hex",18,19),16),2) + "-" +
22         pad(parseInt(messageData.toString("hex",17,18),16),2) + "-" +
23         pad(parseInt(messageData.toString("hex",16,17),16),2) + " " +
24         pad(parseInt(messageData.toString("hex",20,21),16),2) + ":" +
25         pad(parseInt(messageData.toString("hex",19,20),16),2) + ":00"
26
27         //Extract acoustic noise info
28         data.L90 = parseInt(messageData.toString("hex",23,24),16)
29         data.Leq = parseInt(messageData.toString("hex",24,25),16)
30         data.L10 = parseInt(messageData.toString("hex",25,26),16)
31         data.Lmax = parseInt(messageData.toString("hex",26,27),16)
32     }
33     return data
34 }
35
36 //Listening function with callback and config
37 exports.listen = function(cb, config) {
38     var HOST = config.host || '0.0.0.0';
39     var PORT = config.port || 12000;
40     net.createServer(function(sock) {
```

```
41
42          //Device sending data
43          sock.on('data', function(messageData) {
44
45              //Save connection info
46              var info = {}
47              info.address = this.remoteAddress;
48              info.port = this.remotePort;
49              info.timestamp = dateFormat(new Date(), "yyyy-mm-dd
                 HH:MM:ss")
50
51              //Call the callback with connection info and parsed data
52              cb(mergeObjects(info, parseMessage(messageData)))
53          });
54      }).listen(PORT, HOST);
55  }
```

Código 4.3 – *Implementación del sistema de recepción de mensajes del protocolo ULDTP sobre TCP*

4.5 Sistema de gestión de datos e interfaz de usuario

Tal y como se analizó en la sección 3.1.2, el producto *hardware* ha de ir acompañado de una plataforma de gestión que permita implementar los requerimientos extraidos.

Conforme al análisis de la sección 3.5 esta implementación se realizará mediante varias tecnologías web, a saber:

- La estrucutura de *back-end* se implementará en Node.js como entorno de ejecución y se utilizará Express.js como librería de gestión web.
- La estructura de *front-end* se implementará utilizando el paradigma Modelo-Vista-Controlador ofrecido por Angular.js. Como librería de diseño web e interfaz de usuario se utilizará Bootstrap.
- Las cuestiones relacionadas con la autenticación, persistencia de la conexión y *cookies* se delegarán a la librería Passport.js

En general, la figura 4.17 se puede observar como se integran estas tecnologías en el sistema. En primer lugar, una llamada a la URL raíz del servidor web devolverá un conjunto de ficheros HTML y CSS relacionados con Bootstrap que conformarán la interfaz de usuario. En esta misma llamada se descargarán en el navegador web del usuario un conjunto de códigos en JavaScript que conforman la lógica de funcionamiento del programa, esta implementación hace uso de la tecnología Angular.js.

Una vez descargada la interfaz de usuario web y la lógica de dicho programa, el navegador del usuario se autenticará contra el sistema haciendo una llamada a **/login** y descargará la

información de las entidades que se relacionan con él (véase la arquitectura de entidades y relaciones en la sección 3.1.2.1) mediante una llamada a **/appData**.

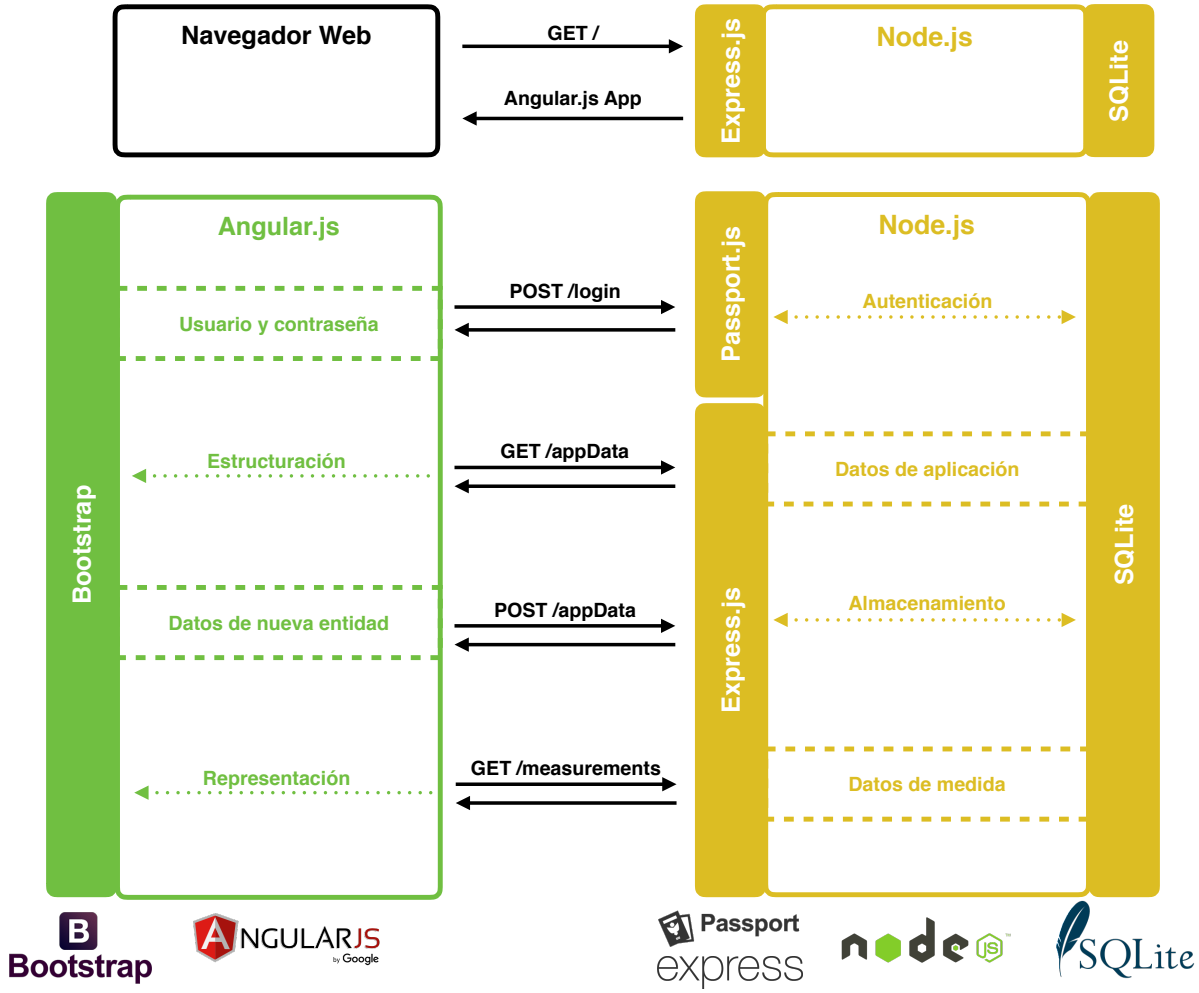


Figura 4.17 – *Estructura y tecnologías utilizadas en la sistemas back-end y front-end.*

Esta última llamada a **/appData** devolverá un fichero JSON con:

- En caso de ser un usuario de tipo cliente: información relacionada con sus lugares y sus propios dispositivos, información de su instalador e información de las administraciones a las que pertenecen sus lugares.
- En caso de ser un usuario de tipo instalador: información relacionada con los lugares y dispositivos de cada uno de los clientes, así como información de todas las administraciones registradas.

- En caso de ser un usuario de tipo administración: información relacionada con los lugares y dispositivos que estén asociados a su jurisdicción.

- En caso de ser un administrador del sistema: toda la información disponible.

Por último, mediante llamadas a la dirección **/measurements** se podrá descargar los datos de un dispositivo concreto durante un periodo de tiempo concreto. Estos datos se descargarán en formato JSON y estarán listos para que la interfaz de usuario basada en Bootstrap los represente en gráficas o tablas.

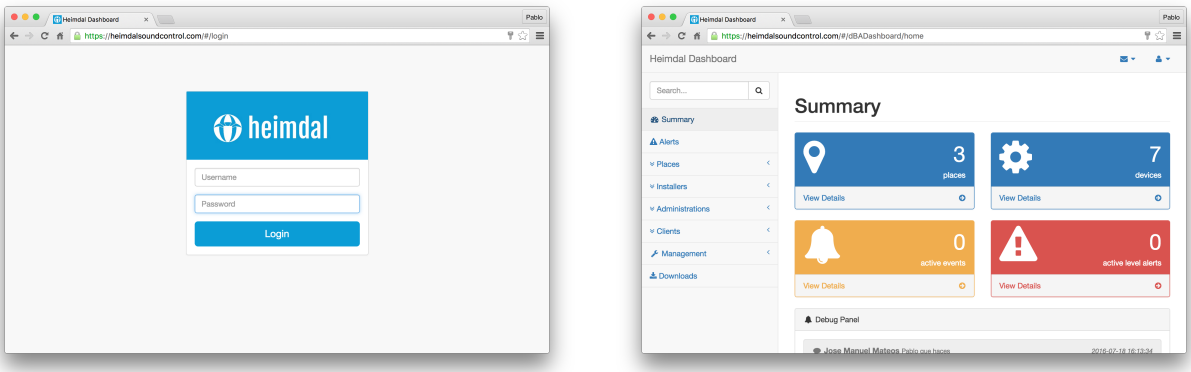
Algunas otras tecnologías que cabe mencionar debido a que se han utilizado para resolver ciertos problemas del subsistema tratado en este capítulo son:

- El tema gratuito SB Admin 2 [15] para Bootstrap que ha permitido implementar un panel de administración estético sin mucha dificultad en la parte de diseño gráfico

- La librería de gráficos C3.js [1] ha permitido generar gráficos con los datos de las mediciones acústicas.

- La librería Leaflet.js [18] para generación de mapas personalizados.

El resultado gráfico de la implementación del sistema se puede observar en la figura 4.18 y en la figura 5.6.



(a) *Login* **(b)** *Dashboard*

Figura 4.18 – *Interfaz web de usuario basada en Bootstrap.*

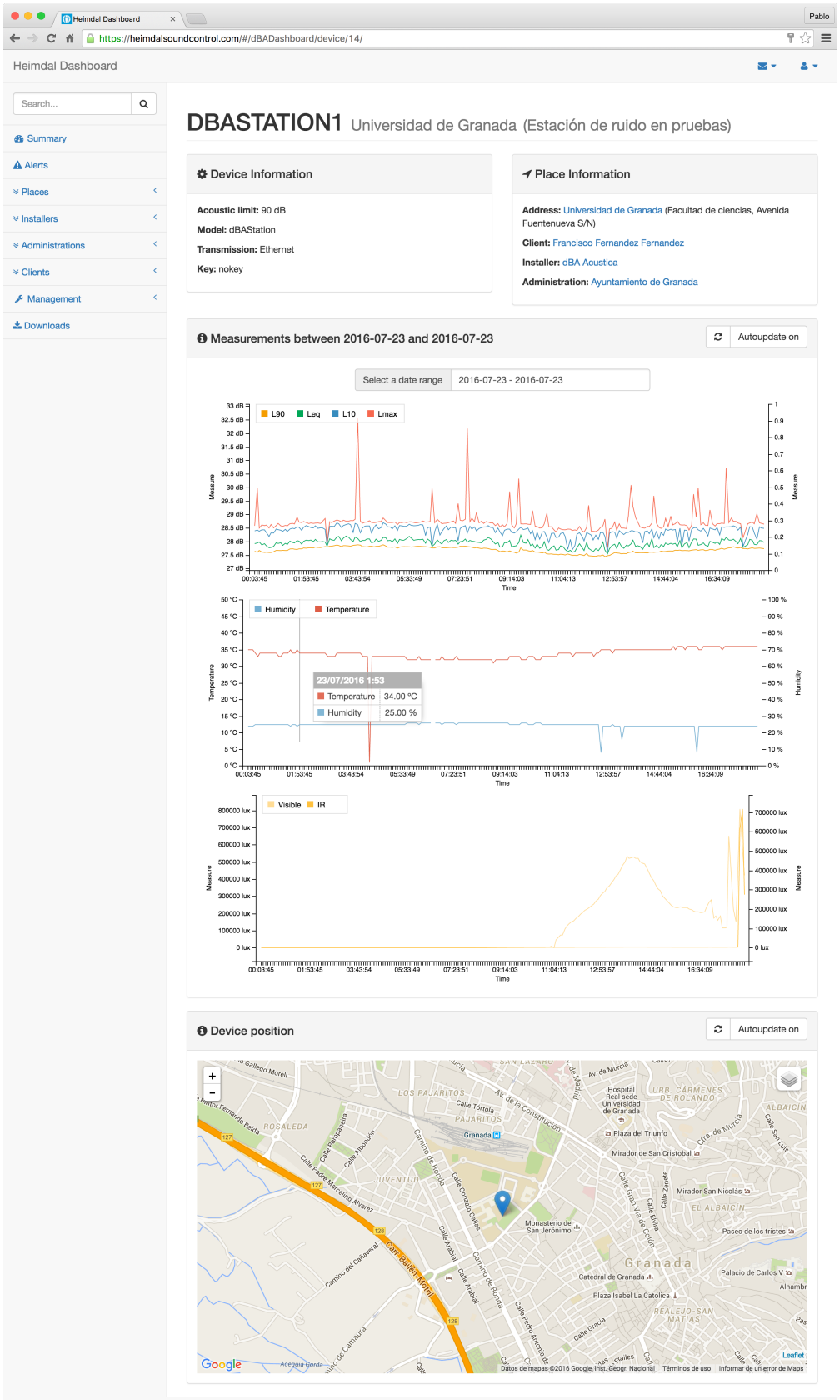


Figura 4.19 – *Interfaz de muestra de datos del dispositivo hardware.*

4.6 Implementaciones de seguridad

En esta última sección del capítulo 4 de diseño, se comentarán las implementaciones de seguridad tratadas a lo largo del mismo.

4.6.1 Seguridad en la transmisión de información

En primer lugar respecto a la transmisión de datos entre los dispositivos y la plataforma de gestión, los parámetros en base a los que se genera la firma digital tratada en la sección 4.4 son un par de claves público-privada RSA de 2048 bits, un algoritmo *hash* SHA-2 de 256 bits y el mecanismo de firma criptográfica digital PKCS1.

En cuanto al sistema que garantiza la confidencialidad, la implementación confia en los certificados digitales emitidos por Let's Encrypt [7]. Mediante dichos cetificados, emitidos a partir del certificado raiz DST Root CA X3 de la empresa Digital Signature Trust Co., se asegura la utilización del protocolo HTTPS de forma confiable.

Los certificados se implementan sobre un servidor HTTPS *reverse back-end web proxy* basado en Nginx, tal y como se mostró en la figura 3.24

En la figura 4.21 se trata de representar las implementaciones y la forma en la que las mismas actuan para robustecer el sistema frente a las amenazas a la seguridad más comunes, a saber:

- En la figura 4.21a se representa como mediante el protocolo HTTPS implementado, un posible atacante (representado en naranja) sería incapaz de leer los datos transmitidos entre la plataforma *hardware* y el sistema de gestión.

- En la figura 4.21b se representa como mediante el sistema de firma criptográfica digital implementado, aunque un posible atacante interceptara los mensajes, cualquier modificación en los datos sería detectada gracias a la firma digital mediante la clave privada del emisor.

- En la figura 4.21c se observa como un intento de suplantación de la identidad de un emisor sería detectada gracias a que unicamente el legítimo emisor tiene la clave privada necesaria para almacenar datos en el sistema de gestión con un número de serie concreto.

- Por último, en la figura 4.21d se trata de representar como es imposible que un mensaje firmado y transmitido sea repudiado por el mismo alegando intercepciones, modificaciones o errores en la transmisión. Siempre suponiendo que la clave privada no ha sido compartida con ninguna tercera entidad.

4.6.2 Seguridad en el sistema de gestión e interfaz de usuario

Además de la securización de la transmisión de los dispositivos es necesario mantener confidencial y autenticado el acceso de los diferentes usuarios al sistema de gestión de datos. Tal y como se comentó en la sección 3.6, la librería **Passport.js** se encargará de implementar estos mecanismos.

En general, tal y como muestra la figura 4.20, el proceso de registro se realiza mediante el almacenamiento en un medio persistente (base de datos SQLite) una versión *hasheada* y *salteada* de la contraseña del usuario. El hecho de utilizar un hash permite que en caso de filtración de información no sea posible obtener la contraseña de los usuario en texto plano. Por su parte el salt de 30 caracteres alfanuméricos dificulta los ataques de fuerza bruta mediante diccionario o *rainbow tables*.

En cuanto a la autenticación, unicamente se repetirá el procedimiento de *hash* con *salt* de la contraseña a comprobar y se comparará con la versión almacenada. Una vez autenticado el usuario Passport.js se encargará de mantener el acceso utilizando *cookies* de sesión que caducan una hora después de haber sido iniciada, si el usuario no cierra la sesión antes.

Por último, y como sucedía en la sección 4.6.1, el protocolo HTTPS permite que las comunicaciones entre ambas partes sean confidenciales y segura frente a escuchas.

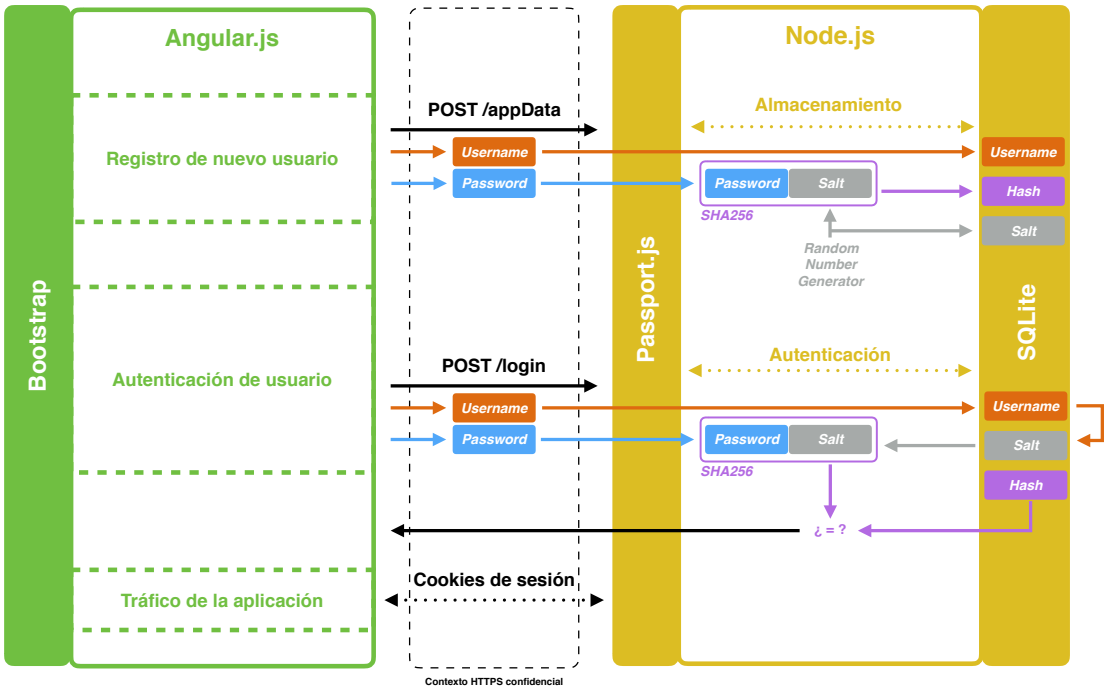
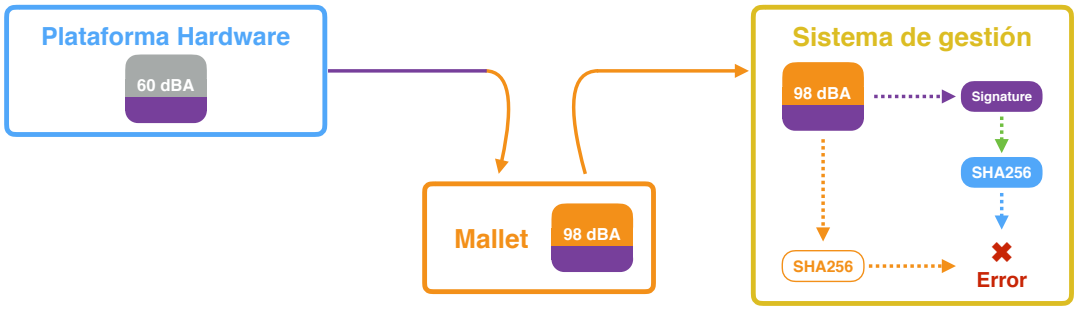
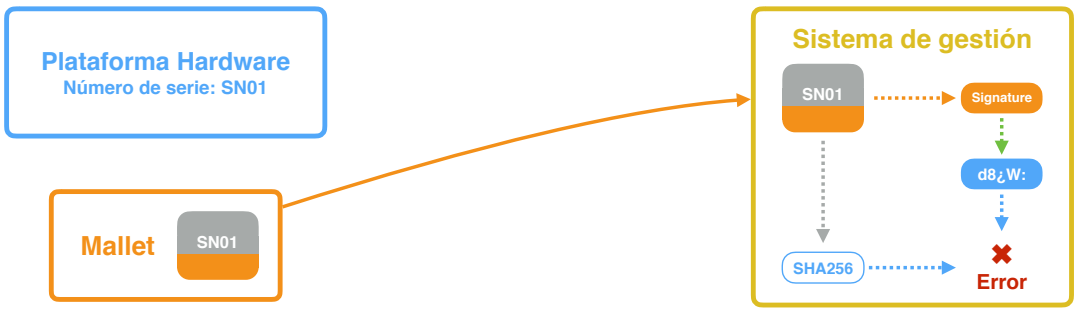


Figura 4.20 – *Registro, autenticación y securización de la comunicación entre usuario y sistema de gestión.*

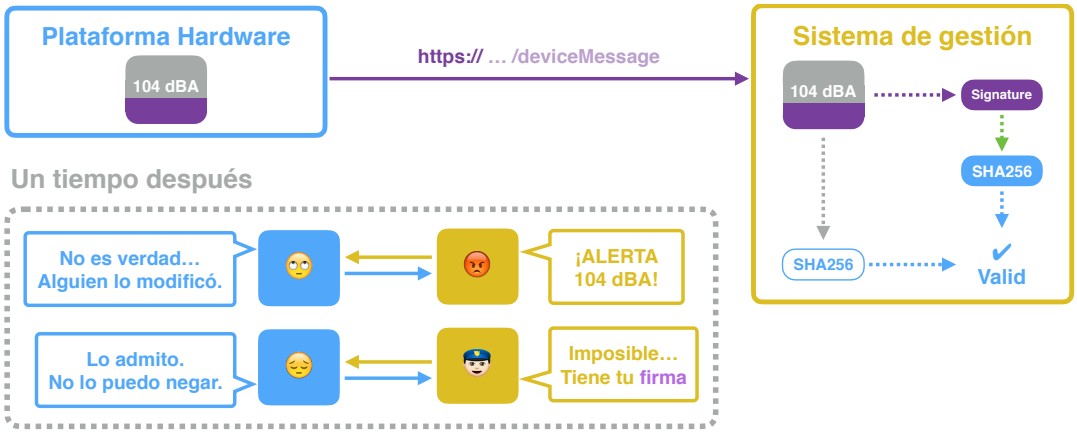
(a) *Confidencialidad*



(b) *Integridad*



(c) *Autenticación*



(d) *No repudio*

Figura 4.21 – *Implementaciones relacionadas con la seguridad.*

CAPÍTULO

5

VALIDACIÓN Y TEST

Tal y como se detalló en el procedimiento de desarrollo enfocado a producto mostrado en la figura 3.1 del capitulo de análisis (capitulo 3), tras el análisis, selección de tecnologías, diseño e implementación es necesario revisar los requisitos del cliente y validar que el producto obtenido cumple con ellos.

De esta manera, regresando a los requisitos firmados con el cliente expuestos en la sección 2, se extraen los siguientes resultados numerados acorde a los requisitos **principales**:

1. El sistema es capaz de **configurar** y **extraer** información del sonómetro de clase I RION NA-27 utilizando el puerto serie implementado en el *hardware* (sección 3.3.4.5). Existe una librería que implementa esta funcionalidad (expuesta en el código 4.12).

2. El producto *hardware* dispone de controles de configuración implementados en la propia plataforma. Mediante ellos es posible observar el estado actual del sistema y realizar pequeños cambios en la configuración, concretamente en el periodo de lectura y transmisión de los datos de los sensores. Los controles implementados son:

   - Pantalla LCD de 4 filas y 20 columnas (figura 5.1).
   - Zumbador o *buzzer* para emitir señales sonoras.
   - *Rotary-push* para navegar y seleccionar variables mediante la rotación de un control.
   - Botón de apagado.

Figura 5.1 – *Interfaz LCD en funcionamiento.*

3. El total de los datos generados por el dispositivo se pueden descargar desde la plataforma web de gestión en formatos Microsoft Excel o CSV tal y como muestra la figura 4.19. Por otra parte el dispositivo almacena un fichero de registro con todas las mediciones realizadas y sus respectivas fechas en formato CSV que puede ser recuperado al tener acceso al *hardware*.

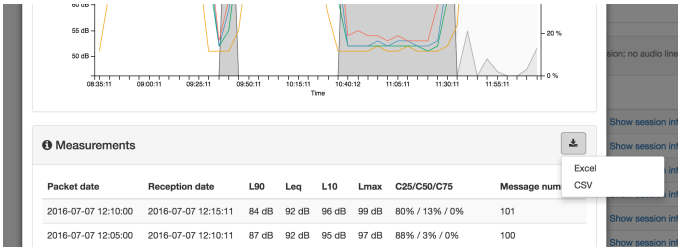


Figura 5.2 – *Botón de descarga de datos acústicos en formatos estándares.*

4. El producto dispone de un protocolo **seguro** de transmisión de datos hacia Internet implementado en Python y basado en firmas digitales. Mediante este protocolo, es capaz de registrarse en el sistema de recepción de datos y posteriormente transmitir periódicamente información hacia dicho sistema. Los detalles del procedimiento de transmisión se pueden observar en la sección 4.4. Los detalles de la securización de este procotolo se muestran en la sección 3.6 y 4.6.1.

5. El sistema dispone de una interfaz gráfica de gestión implementada en tecnología web capaz de mostrar los datos recopilados por las estaciones de monitorización tanto en formato gráfico como en formato de tabla. Esta interfaz, mostrada en las figuras 4.18 y 4.19 permiten además gestionar toda la estructura de entidades relacionadas con los productos: instaladores, administraciones, clientes, lugares, etc...

Respecto a los requisitos **secundarios** del cliente, se pueden extraer las siguientes conclusiones en base al trabajo realizado:

1. Además de la funcionalidad principal de medición de ruido acústico, el sistema es capaz de percibir y registrar las siguientes variables ambientales:
   - Humedad.
   - Temperatura.
   - Luminosidad visible e infrarroja.
   - Posición GPS.
2. El sistema se alimenta mediante una fuente conmutada alimentada con entre 12 y 40 V. El consumo de la extensión hardware alimentada se puede observar en la figura 5.4. Si se añade la plataforma de computación y el sistema operativo, el producto funcionando, como se observa en la figura 5.3 mantiene un consumo de 250 mA.

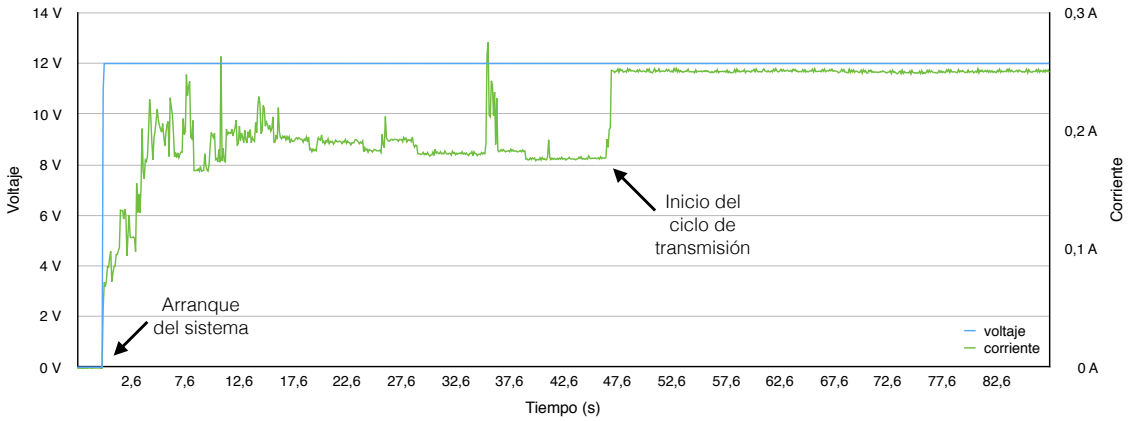


Figura 5.3 – *Consumo del sistema completo durante el arranque.*

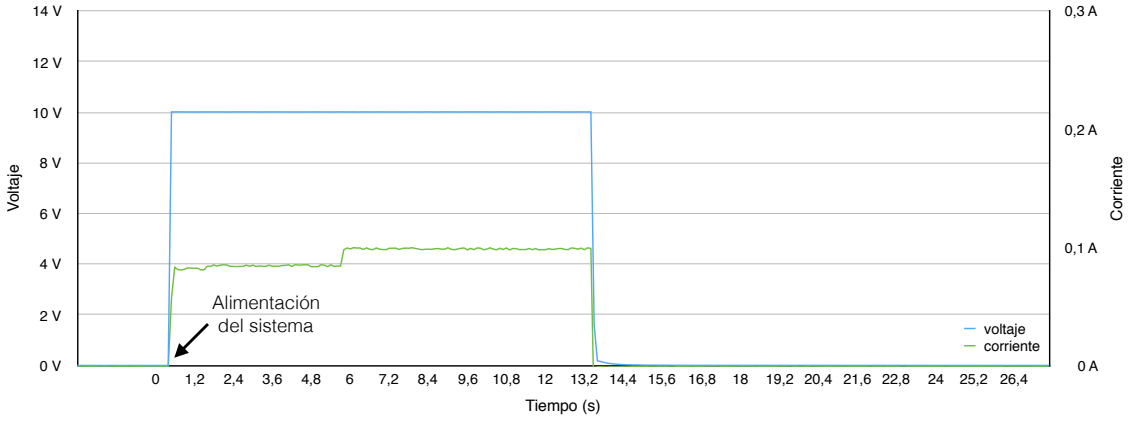


Figura 5.4 – *Comsumo de la extensión hardware alimentada.*

3. El sistema de gestión implementa una estructura de roles y permisos descrita en el apartado 4.5 capaz de proporcionar a cada usuario un tipo de información diferente, siempre relativa a su rol dentro del sistema y a las entidades asociadas al mismo.

4. La interfaz de usuario, gracias la implementación Responsive Web Design de Bootstrap, es capaz de ajustarse a multitud de dispositivos. De esta manera, sin necesidad de un rediseño la interfaz se puede adaptar a dispositivos móviles o tablets. En la figura 5.5 se puede observar el resultado.

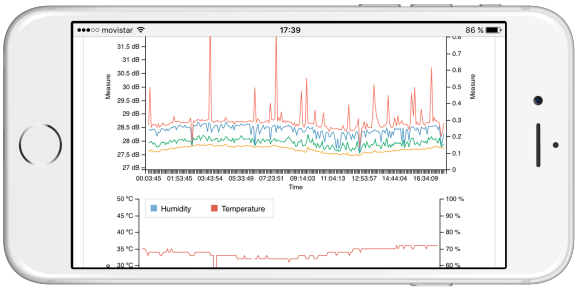


Figura 5.5 – *Interfaz gráfica mostrada en un dispositivo móvil.*

5. Respecto a la compatibilidad con aparatos similares del mercado, el sistema es capaz de recibir mensajes de dispositivos de medición acústica que implementen el protocolo ULDTP. En la sección 4.4.1 se describe la interfaz de recepción de los dispositivos que utilicen este protocolo. De forma complementaria se ha adaptado la interfaz de usuario para que se muestren sin problemas los datos recibidos desde este tipo de dispositivos comerciales (figura 5.6)

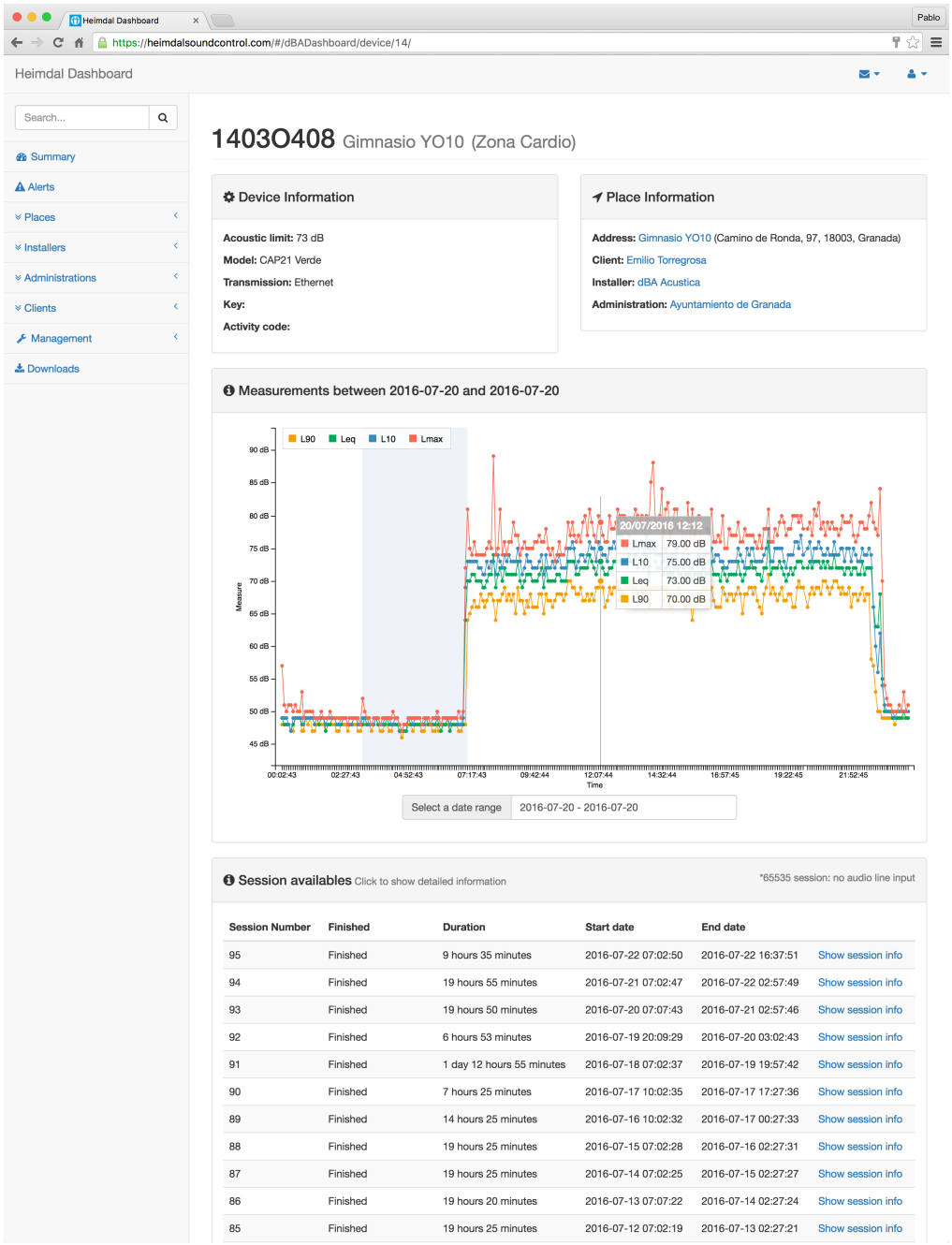


Figura 5.6 – *Interfaz gráfica mostrando datos de equipos comerciales que utilizan ULDTP.*

5

CAPÍTULO

6

CONCLUSIONES Y TRABAJO FUTURO

En este documento se ha presentado el desarrollo de un sistema de medición de ruido acústico y variables ambientales dentro del contexto de un proyecto de innovación de una empresa local. Este sistema permite una integracion vertical hacia arriba de la empresa, permitiendo desarrollar tecnología y *know-how* propio.

Desde el punto de vista del alumno ha supuesto un reto a superar en forma de Trabajo Fin de Máster de la titulación de Máster en Ingeniería de Telecomunicación. Para ello se han tratado de presentar tantos problemas como capacidades se han adquirido a lo largo del grado y el ciclo superior ofrecidos por la Escuela Técnica Superior de Ingeniería Informática y de Telecomunicación. Dentro de estas capacidades, especialmente se pueden observar los conocimientos en electrónica, telemática y seguridad en las comunicaciones a través de Internet.

De forma a transversal a estas capacidades, se ha tratado de completar un proyecto de desarrollo de un producto basado en los requisitos de un cliente para tratar las problemáticas y pormenores de un proceso de ingeniería de producto completo, por ejemplo: adquisición de material, gestión de la comunicación con el cliente, aceptación de presupuestos, etc. De forma que el producto desarrollado sea óptimo en cuanto a solución tecnológica, así como en cuanto a inversión, tiempos y satisfacción por parte del cliente.

La estación de medición de ruido acústico y variables ambientales desarrollada y fabricada se resuelve como un producto completo y funcional, listo para ser implementado en el terreno

bajo las restricciones impuestas por el cliente. Sin embargo a continuación se plantean algunas mejoras y lineas de trabajo futuras que se han obtenido a lo largo del proyecto:

- En cuanto al sistema de alimentación propuesto por el cliente (toma de corriente, punto 2 de la sección 2.1), se propone realizar una implementación mejorada que tenga en cuenta la posible necesidad de alimentación autónoma del sistema. Para ello se propone la utilización de un sistema de alimentación secundario basado en baterías. El trabajo de desarrollo se habría de centran en los procesos de carga de la bateria, control de la carga y seguridad del sistema frente a eventos de descarga.

- Relacionado con el punto anterior, y con el fin de desarrollar una versión completamente inalámbrica del producto, se propone integrar un sistema de conexión a Internet inalámbrico. Dadas las caracteristicas del sistema actual. Esta propuesta pasa por incorporar al sistema un dispositivo WiFi USB compatible con el sistema operativo utilizado (Raspbian Jessie Lite). O por otra parte actualizar la plataforma de computación al modelo Raspberry Pi 3, la cual incorpora conectividad inalámbrica de serie.

- Desde el punto de vista del despliegue del producto, sería interesante disponer de un sistema telemático de actualización del software que controla el funcionamiento. Dado el sistema de registro y autenticación de los dispositivos, la solución a esta linea pasa por implementar un sistema de control de versiones *hardware* y un mecanismo de descarga y actualización de nuevas versiones de *software* compatibles con la versión del producto.

- Con el fin de presentar un producto completo y vendible es imprescindible diseñar una caja o soporte mecánico para el *hardware* diseñado.

- Se propone implementar una aplicación de escritorio de gestión del sistema completo basada el paquete de desarrollo Electron [16], el cual permite generar aplicaciones de escritorio basadas en tecnología web.

6

A modo de conclusión final, cabe remarcar la gran cantidad de conocimiento que el alumno considera haber adquirido a lo largo del año de trabajo, y sobretodo la satisfacción personal que siempre supone enfrentarse a retos asociados a la profesión que se ha decidido ejercer.

REFERENCIAS

[1] C3.js graphic library. SB Admin Webpage. http://c3js.org/.

[2] Cisco internet of things. Cisco Website. http://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html.

[3] Ec12e rotary push datasheet. Datasheet. http://www.alps.com/prod/info/E/PDF/Switch/Encoder/EC12E/EC12E.PDF.

[4] Financiacion ue fp8. Times Higher Education Website. https://www.timeshighereducation.com/news/triple-miracle-sees-huge-rise-in-eu-funds-for-frontier-research/416952.article?storycode=416952.

[5] The growth of the internet of things. Video Infographic. https://www.ncta.com/platform/industry-news/infographic-the-growth-of-the-internet-of-things/.

[6] Installing an os in a sd card. Tweaking4All Website. http://www.tweaking4all.com/hardware/raspberry-pi/install-img-to-sd-card/#macosx.

[7] Lets encrypt public certificates. Lets Encrypt Webpage. https://letsencrypt.org/.

[8] M41t81 datasheet. Datasheet. http://pdf.datasheetcatalog.com/datasheet/stmicroelectronics/7529.pdf.

[9] Max3232 datasheet. Datasheet. http://www.ti.com/lit/ds/symlink/max3232.pdf.

[10] Montaje de referencia de una fuente conmutada. LM1117 datasheet. http://www.ti.com/lit/ds/symlink/lm1117.pdf.

[11] Pcf8574 datasheet. Datasheet. http://www.ti.com/lit/ds/symlink/pcf8574.pdf.

[12] Raspbian jessie lite. Raspberry Pi Website. https://www.raspberrypi.org/downloads/raspbian/.

[13] Serie cap21 de dbelectronics. dBElectronics Webpage. http://www.dbelectronics.es/.

[14] Smart anything everywhere. UE Research Website. https://ec.europa.eu/research/participants/portal/desktop/en/opportunities/h2020/topics/5061-ict-04-2017.html.

[15] Tema sb admin 2 para bootstrap. SB Admin Webpage. http://startbootstrap.com/template-overviews/sb-admin-2/.

[16] GitHub. Electron framework. Electron Webpage. http://leafletjs.com/.

[17] IEC. A weighting definition. Standard IEC 61672:2003.

[18] Leaflet.js. Leaflet.js maps library. Leaflet.js Webpage. http://leafletjs.com/.

[19] Lindosland. Acoustic weighting curves. Wikipedia.

[20] MongoDB. Mongodb. MongoDB Website. https://www.mongodb.com.

[21] Nginx. Nginx. Nginx Website. https://www.nginx.com/.

[22] Sparkfun. Venus638flpx sparkfun. Sparkfun Website. https://www.sparkfun.com/products/11058.

[23] TAOS. Tsl2561t datasheet. Datasheet. https://cdn-shop.adafruit.com/datasheets/TSL256x.pdf.

[24] Tezer, O. Sqlite vs mysql comparation. Digital Ocean Website. https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql/-a-comparison-of-relational-database-management-systems.

[25] UE. Financiacion ue fp1-7. UE Website. http://ec.europa.eu/research/fp7/pdf/fp-1984-2013_en.pdf.

[26] Vixie, P. Crontab manpage. Manpage. http://linux.die.net/man/1/crontab.

ANEXO

A

PRESUPUESTO

A.1 Costes electrónicos

En la presente sección se detallan los costes relacionados con la implementación electrónica de la plataforma *hardware*.

En la tabla A.1 se presentan los costes asociados a la fabricación de la PCB sin tener el cuenta la mano de obra de fabricación.

Por otra parte, en la tabla A.5 se detallan los costes asociados a cada uno de los componentes utilizados en la fabricación.

Elemento	**Coste (€)**
Sustrato de cobre	10.00 €
Recursos humanos	50.00 €
Tiempo de máquina	5.00 €
TOTAL	**65.00 €**

Tabla A.1 – *Costes de fabricación PCB.*

De esta manera, en la tabla A.2 se presenta el coste total de la producción de la PCB: teniendo en cuenta la fabricación y los componentes utilizados asciende a 162.46€.

Item	Coste (€)
Construcción de la PCB	65.00 €
Dispositivos electrónicos	97.46 €
TOTAL	**162.46 €**

Tabla A.2 – *Coste total de la implementación de la PCB.*

A.2 Software

A continuación se especifican los precios de las licencias de los paquetes de *software* utilizados.

Software	**Dueño de la licencia**	**Coste (€)**
Altium Designer 14.3	UGR	Licencia gratuita
Atom IDE	Pablo Garrido Sánchez	Licencia gratuita
TeXnicCenter	Pablo Garrido Sánchez	Licencia gratuita
Miktex	Pablo Garrido Sánchez	Licencia gratuita
SumatraPDF	Pablo Garrido Sánchez	Licencia gratuita
	TOTAL	0 €

Tabla A.3 – *Costes del software.*

A.3 Recursos humanos

Durante la realización de este proyecto se han generado unos costes de contratación de dos personas.

La primera de ellas es un ingeniero junior (10 €/h) trabajando a tiempo completo durante un nueve meses. Por otra parte, como encargado del proyecto y mentor, se contrata a un ingeniero senior (50 €/h) con una disponibilidad de 5 horas semanales. Son, por tanto, los gastos en recursos humanos los que indica la figura A.4.

Cargo	Horas	Coste(€)
Ingeniero junior	1440	14400.00 €
Ingeniero senior	180	9000.00 €
TOTAL		**23400** €

Tabla A.4 – *Costes en recursos humanos.*

Tabla A.5 – *Presupuesto de componentes electrónicos de la PCB*

Item	Descripción	Unids.(Usado)	Coste/un.(€)	Coste(€)	Coste usado(€)
Raspberry Pi 2 M. B	Plataforma de computación	3(1)	35.300 €	105.900 €	35.300 €
LM1117IMP-3.3	Regulador de voltaje 3.3 V	5(1)	0.960 €	4.800 €	0.960 €
LM2576HVS	Regulador de voltaje 5 V	5(1)	6.290 €	31.450 €	6.290 €
ITS4141	Diodo rectificador 3 A	5(1)	0.604 €	3.020 €	0.604 €
Int. de sobrecorriente	Protección alimentación.	4(1)	1.785 €	7.142 €	1.785 €
MAX3232	Conv. de niveles TTL-RS232	5(1)	1.826	9.130	1.826
Conector DB9	Con. comunicación RS232	0(1)	0.300 €	0.000 €	0.300 €
DHT11	Sensor de hum. y temp.	0(1)	5.000 €	0.000 €	5.000 €
TSL2561T	Sensor luminosidad	5(1)	2.794 €	13.970 €	2.794 €
M41T81	RTC SOIC	0(1)	2.180 €	0.000 €	2.180 €
Crystal 32.768KHz	Cristal para RTC	3(1)	0.550 €	1.650 €	0.550 €
Venus GPS	Modulo GPS	0(1)	49.950 €	0.000 €	49.950 €
Conectores hembra	Con. para módulo THT	3(3)	1.200 €	3.600 €	3.600 €
Interruptor tactil	Interruptor de apagado	20(1)	0.156 €	3.120 €	0.156 €
LG R971	LED SMD verde	250(2)	0.040 €	10.000 €	0.080 €
Rotary push	Controlador H-M	3(1)	2.380 €	7.140 €	2.380 €
Buzzer	Avisos Acústicos	5(1)	0.586 €	2.930 €	0.586 €
LCD 20x4	Pantalla Humano-Máquina	4(1)	9.540 €	38.160 €	9.540 €
PCF8574	Expansor GPIO para LCD	4(1)	1.160 €	4.640 €	1.160 €
Conec. atornillados	Conectores externos	0(11)	0.200 €	0.000 €	2.200 €
MOSFET N	115 mA 60 V SOT23	200(5)	0.027 €	5.400 €	0.135 €
Inductor	ELC18B 100uH	3(1)	1.890 €	5.670 €	1.890 €
Cond.	100 uF SMT	10(1)	0.481 €	4.810 €	0.481 €
Cond.	10 uF SMT	15(9)	0.194 €	2.910 €	1.746 €
Cond. SMD 0805	Varios valores	0(13)	0.020 €	0.000 €	0.260 €
Res. SMD 0805	Varios valores	0(19)	0.020 €	0.000 €	0.380 €
				TOTAL	**97.46 €**