Universidad Zaragoza
1542

Trabajo Fin de Grado

Sistema de monitorización climática para agricultura, mediante el uso de tecnologías IoT

Agriculture climatic monitoring system, using IoT technologies

Autor/es
Javier Perales Gran

Director/es
José Antonio Perales Gran

Titulación del autor
Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2021

Resumen: Sistema de monitorización climática para agricultura, mediante el uso de tecnologías IoT

Buscamos desarrollar un sistema para monitorizar las variables climáticas en tiempo real, con el objetivo de detectar precozmente las condiciones para el desarrollo de enfermedades fúngicas en cultivos agrícolas de frutales y viña, para minimizar el daño producido por estas enfermedades, y reducir el uso de pesticidas. Para ello, en el marco de la empresa Agropredicción, se ha diseñado y construido un dispositivo electrónico alimentado por batería recargable, capaz de obtener información de un sensor de temperatura y humedad, y otro de humectación de hoja, en un entorno remoto. Además, este dispositivo es capaz de enviar la información a la nube de forma desatendida mediante las redes de telefonía móvil de tipo NB-IoT, LTE-M y GSM, y utilizando el protocolo de mensajería MQTT en la capa de aplicación.

También se ha desplegado en la nube un broker MQTT, al que se conecta el dispositivo, y un servidor Node.js que hemos desarrollado, que se conecta al broker MQTT y se encarga de procesar los mensajes recibidos para almacenarlos en una base de datos PostgreSQL. Así mismo, este servidor permite crear y consultar en base de datos, mediante una API REST, usuarios, clientes, o fincas, entre otras cosas. Por último, se ha creado una interfaz de usuario que se comunica con el servidor Node.js, y permite iniciar sesión a un usuario, crear y consultar fincas, puntos geográficos de instalación para los dispositivos sensores y visualizar de forma gráfica la información del histórico de temperatura y humedad de los dispositivos asociados a una finca.

Tabla de contenidos

1. Introducción

En este capítulo se describe la necesidad que pretendemos solucionar con el proyecto, se detalla el contexto en el que se lleva a cabo, los objetivos del proyecto en su totalidad, y el alcance del trabajo académico. Además de una explicación de la metodología utilizada para su desarrollo.

1.1. Motivación

El presente proyecto surge de una idea de mi hermano José Antonio, Ingeniero Técnico Agrícola y director del proyecto. Se trataba de encontrar una forma de detectar precozmente las condiciones para el desarrollo de enfermedades fúngicas en los cultivos. Esto permitiría minimizar los daños provocados por estas enfermedades, al poder tratarlas a tiempo, antes de que los daños se hagan visibles, lo que ya sería demasiado tarde para preservar la calidad del fruto y la salud de los árboles. Al mismo tiempo, esto podría reducir el uso de fungicidas frente a un enfoque de uso preventivo, en el que se apliquen estos productos sin información precisa acerca de las condiciones climáticas de proliferación de los hongos que provocan la enfermedad. Así pues, los potenciales usuarios del sistema son agricultores de cultivos intensivos (principalmente frutales y viña), interesados en mejorar la salud de sus plantaciones, y ahorrar dinero y tiempo en aplicar fungicidas.

1.2. Contexto

Tras comprobar que existían modelos matemáticos predictivos para algunas enfermedades como el *Moteado* del manzano, o el *Mildiu* de la vid, basados en variables climáticas tales como la temperatura del aire, la humedad relativa del aire, y la humectación de las hojas, nos dispusimos a investigar las soluciones existentes para medir estas variables. Descubrimos diferentes opciones en el mercado como los dispositivos de Metos[1], Libelium[2] y posteriormente Cesens[3]. Sin embargo, estas empresas no ofrecían un servicio completo para el agricultor, sino más bien un “hazlo tú mismo”, y aunque en otros casos sí ofrecían un servicio orientado al agricultor, los precios de este tipo de dispositivos eran demasiado elevados para ser rentabilizados por explotaciones agrícolas frutales, y su nicho de mercado se limitaba casi en exclusiva a la viña, por ser un cultivo con mayor rentabilidad, cuyos márgenes sí admiten esos costes. Por este motivo nos planteamos desarro-

1 https://metos.at/es/
2 https://www.libelium.com/
3 https://www.cesens.es/

llar una solución integral de bajo coste[4], que fuese viable para un número amplio de explotaciones agrícolas.

En el año 2018 constituimos una sociedad para desarrollar el proyecto, Agroprediccion SL, y comencé a montar prototipos electrónicos para la recogida de los datos en el campo en tiempo real. Tras varios tropiezos y aprendizajes sobre diseño de electrónica, y tras un par de años de experiencia a tiempo parcial en una empresa de desarrollo de software, este trabajo presenta el desarrollo de un prototipo funcional, a partir del cual se pretende lanzar un producto mínimo viable al mercado.

1.3. Objetivos y alcance

El objetivo último del proyecto es desarrollar un sistema integral de recogida y procesamiento de datos climáticos en tiempo real, que alerte a los agricultores del nivel de riesgo de enfermedades. Esto incluye el desarrollo de un dispositivo electrónico al que se puedan conectar diferentes sensores climáticos, que sea resistente a las inclemencias meteorológicas, envíe los datos a Internet en tiempo real de manera autónoma, sea energéticamente independiente, y en definitiva, que pueda funcionar ininterrumpidamente con la mínima intervención humana durante, al menos, 1 año.

Entre los objetivos también se incluye el desarrollo de un sistema de información que reciba las lecturas de los sensores, las almacene en una base de datos, y procese los datos aplicando los modelos matemáticos de riesgo para enviar alertas a los agricultores. Así mismo deben poder visualizarse de forma gráfica las lecturas de los sensores de un dispositivo: temperatura, humedad, humectación de hoja y otros sensores.

Los objetivos anteriormente expuestos son los objetivos generales del proyecto. Sin embargo, considero que el desarrollo con detalle del sistema completo mencionado excede la extensión que debe tener un Trabajo de fin de grado, por lo que el alcance de esta memoria se limitará al desarrollo de un prototipo electrónico que recoja la lectura de humedad y temperatura de un sensor y envíe datos a un servidor en Internet, el desarrollo y configuración de un sistema de información que recoja y almacene los datos recibidos, y el desarrollo de una interfaz web que permita ver de forma gráfica los datos recogidos. Es decir, será un sistema similar al objetivo final, pero sin algunas características, como el procesamiento de modelos matemáticos, el envío de alertas a los clientes, y otras carac-

4 Ver el anexo “Valoración de costes”

terísticas que serían deseables en un producto comercial, pero cuyo desarrollo llevaría mucho tiempo.

1.3.1. Funcionalidad del sistema

Prototipo electrónico. Se realizará una placa de circuito electrónica (PCB), que sea capaz de alimentarse a partir de una batería recargable, que tenga un sistema de comunicaciones inalámbricas para el envío de los datos de los sensores, y a la que se pueda conectar, al menos, un sensor que mida la humedad ambiental y la temperatura del aire. Además, se desarrollará el software embebido necesario para recoger los datos de humedad y temperatura del sensor, y enviarlos periódicamente a un servidor en Internet.

Sistema de información. Se configurará y desarrollará un servidor en la nube que sea capaz de recibir los datos del sensor con el mismo protocolo de aplicación con el que los envía la placa electrónica. Por cada mensaje recibido, la lectura de los sensores se almacenará en una base de datos junto con la fecha y hora para su posterior consulta. Además, el sistema tendrá puntos de acceso a la información mediante una API REST[5] que permitan la consulta de los datos desde una interfaz web SPA (*Single Page Application*)[6].

Interfaz de usuario. Se desarrollará una aplicación web que permita ver el histórico de los valores de humedad y temperatura del dispositivo mediante gráficos.

1.4. Metodología

Se ha utilizado la herramienta de planificación Notion[7], principalmente una lista de tareas jerarquizadas. En el caso de la escritura de la memoria, se creó un tablero, y se fueron definiendo los capítulos y secciones que quería que tuviera en un principio, marcando las secciones a medida que las iba escribiendo. Cuando me daba cuenta de que alguna sección definida era mejor fusionarla con otra, o me percataba de que debía añadir otra nueva, lo cambiaba en la lista de tareas. En el caso del diseño e implementación la metodología ha sido la misma, se definían tareas en una lista, a diferentes niveles de detalle. Por

5 API: «Es un conjunto de subrutinas, funciones y métodos que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción servicios disponibles para un programador para realizar ciertas tareas.» [1]
REST: Es un estilo de arquitectura software para sistemas hipermedia, por ejemplo web. Se caracteriza por ser una arquitectura cliente-servidor, sin estado y con una interfaz uniforme. [2]
Es una de las arquitecturas para transferencia de datos más utilizadas en la web.

6 SPA: Aplicación web que requiere una única carga de página, descargando en el primer acceso todo el código HTML, Javascript y CSS de todas las páginas. Evitando así actualizaciones de la página en el navegador, haciendo la experiencia de usuario más fluida. Los recursos necesarios se cargan dinámicamente de forma transparente para el usuario mediante peticiones asíncronas. [3]

7 https://www.notion.so

ejemplo, en un nivel “diseñar el dispositivo electrónico”, dentro “pensar los bloques funcionales”, etc. Y se iban marcando una vez completadas, para tener una visión general del estado del proyecto en todo momento. En el caso del software realizado (software embebido del dispositivo, servidor Node.js y aplicación web con React), se ha utilizado el sistema de control de versiones Git[8], con un repositorio privado para cada parte en la plataforma Bitbucket[9].

1.5. Contenido de la memoria

Pasamos a describir brevemente el contenido de los siguientes capítulos. En “estado del Arte” presentamos alternativas de conectividad inalámbrica y protocolos de la capa aplicación, y elegimos los más adecuados para el desarrollo del proyecto. En “diseño e implementación” mostramos la arquitectura general del sistema, y explicamos los trabajos realizados para el diseño del dispositivo electrónico, del sistema de información, de la interfaz de usuario y el despliegue de todo ello en la nube. En el apartado de “conclusiones” mostramos los aprendizajes y errores que se han cometido en el desarrollo del proyecto, y algunos trabajos futuros que se podrían desarrollar sobre la base del mismo. Por último, tenemos varios anexos en los que presentamos una prueba de concepto que se llevó a cabo, se hace una valoración de los costes del proyecto, entramos más en detalle en el diseño de los circuitos electrónicos del dispositivo y de un sensor que se ha diseñado, y adjuntamos un diagrama con la estructura de la base de datos.

8 Sistema de control de versiones distribuido, de código abierto. Es el más utilizado en la industria del software.
9 https://bitbucket.org/

2. Estado del arte

En este capítulo tratamos los dos aspectos de la rama telemática más importantes para el desarrollo del proyecto: las tecnologías de conectividad inalámbrica, y los protocolos de aplicación para el envío de pequeños paquetes de datos. Presentamos las tecnologías candidatas, y explicamos razonadamente la elección realizada.

2.1. Tecnologías de conectividad inalámbrica

Es necesario que el dispositivo electrónico sea capaz de enviar datos desde lugares remotos (fincas agrícolas), en los que no es viable utilizar conectividad a Internet mediante un cable, por lo que la alternativa más razonable es utilizar un sistema de conectividad inalámbrica. A continuación presentamos varias alternativas.

2.1.1. Redes celulares de telefonía móvil

Las redes de telefonía móvil celular permiten, desde su inicio, establecer conexiones de voz inalámbricas entre dispositivos móviles que se encuentran separados desde centímetros hasta miles de kilómetros. Posteriormente, con el estándar GSM se añadió soporte para servicio de mensajes de texto (SMS), y en evoluciones posteriores de GSM, se añadió soporte para envío y recepción de paquetes de datos que permitían la conexión a Internet (GPRS y EDGE)[1]. Más tarde, hacia el año 2001, se introdujeron las primeras redes comerciales de lo que se conoce como 3G (UMTS, HSPA, HSPA+ y CDMA2000)[2] que mejoraban la velocidad y la capacidad de usuarios en la red. En nuestros días las redes LTE[3] (4G) son el estándar más extendido, permiten velocidades de datos superiores a las de 3G y van dejando paso poco a poco a la siguiente evolución, el 5G, que cada vez está desplegado en más núcleos de población.

Hasta la aparición de LTE, la mayoría de dispositivos IoT que utilizaban redes móviles se conectaban mediante tecnología GSM (2G-2.5G), puesto que es la tecnología con una cobertura más extendida, y requiere dispositivos más baratos que las tecnologías posteriores. Sin embargo, sobre LTE se desarrollaron algunos protocolos específicos para dispositivos IoT que van ganando terreno poco a poco [8], como son LTE-M y NB-IoT, que per-

1 GPRS fue la primera forma de conexión a Internet a través de redes móviles, basado en GSM. Posteriormente se introdujo EDGE, que es una evolución que introduce mejoras en la velocidad de transferencia de datos. [4], [5]

2 Diferentes implementaciones de las redes de 3ª generación dependiendo de la región en el mundo. En Europa primero se implanto UMTS, y HSPA es una evolución de UMTS. [6]

3 «Estándar para comunicaciones inalámbricas de transmisión de datos de alta velocidad para teléfonos móviles y terminales de datos. Es una evolución de UMTS.» [7]

miten la conexión a la red con dispositivos más simples y económicos, al reducir la complejidad de los transceptores[4] necesarios respecto al LTE común. Además, estos protocolos introducen mecanismos para reducir el consumo de energía de los dispositivos. Por ejemplo, permiten reducir la frecuencia del mecanismo de *paging*[5] para mantener los dispositivos el máximo tiempo posible con el receptor y el transmisor apagados. [10]

Así pues, LTE-M y NB-IoT permiten conexiones móviles más eficientes energéticamente que GSM, y dispositivos con un precio similar o incluso menor. Además tienen mejores condiciones de propagación que LTE y que GSM. En el caso de NB-IoT, utiliza la tecnología de modulación *Ultra NarrowBand*[6], la cual otorga una muy buena tolerancia a la interferencia y el ruido. El principal problema es que la cobertura depende de las estaciones base LTE, las cuales todavía no están tan generalizadas en el medio rural como las de GSM, lo que hace difícil su uso como única forma de conectividad en fincas agrícolas.

Cabe señalar que las tecnologías mencionadas coexisten en la actualidad. En España existen redes tanto 2G, como 3G, 4G y 5G. Sin embargo, hay planes para el apagado progresivo de las redes 3G en primer lugar, y 2G posteriormente. Por lo que no parece una buena idea realizar un despliegue nuevo con dispositivos que dependan de estas tecnologías, ya que según los operadores de telefonía, en el año 2025 las estaciones base de 2G y 3G podrían haber desaparecido. [12]

2.1.2. Sigfox

Sigfox[7] es una tecnología propietaria, cuya red está bajo el control de una empresa con el mismo nombre de la tecnología. Está orientada para dispositivos IoT que envíen datos sin la necesidad de establecer y mantener conexiones de red. Tiene un protocolo muy ligero y orientado a dispositivos de bajo consumo energético, y permite que los costes de los módulos de comunicaciones sean más bajos que otras alternativas. Sigfox funciona en las

4 Transceptor es el nombre que recibe un dispositivo que combina un transmisor y un receptor.

5 El *paging* es el mecanismo mediante el cual la red móvil avisa a un dispositivo de que tiene algo que entregarle. Para ahorrar energía, se establece un acuerdo entre la estación base y el dispositivo móvil para que los mensajes de paging se envíen solo en determinados intervalos de tiempo, para poder mantener el receptor apagado el resto del tiempo. [9]

6 Son sistemas de comunicación en los que el canal tiene un ancho de banda muy estrecho. Sus receptores son muy selectivos y pueden rechazar el ruido y la interferencia permitiendo una relación señal-ruido aceptable con una señal recibida relativamente débil. [11]

7 https://www.sigfox.com/en

bandas de frecuencia ISM[8] entre 862 Mhz y 928 Mhz, dependiendo de la región (868Mhz en Europa) y al igual que NB-IoT, utiliza la tecnología de modulación *Ultra NarrowBand*.

Una de las desventajas es que tiene límites para el envío de paquetes. El tamaño máximo de un paquete es de 12 bytes (excluyendo cabeceras), y solo permite un máximo de 140 paquetes por día. Otra desventaja es la dependencia de una única empresa que posee el monopolio de la red a la que pueden conectarse los dispositivos desarrollados, y que los costes de conexión a la red son más elevados que los de los operadores de telefonía móvil, permitiendo además el envío de mucho menos volumen de datos.

2.1.3. Otras alternativas

Otras tecnologías de conectividad inalámbrica son WiFi, Bluetooth, tecnologías propietarias en bandas de frecuencias ISM, o tecnologías de redes malladas como ZigBee. Sin embargo, todas estas alternativas tienen el problema de que necesitan un *gateway* que utilice otra tecnología para conectar realmente con la red de Internet, como por ejemplo un *gateway* de GSM a WiFi. Dado que no planeamos instalar muchos dispositivos unos cerca de otros, esto resta flexibilidad a la solución, y la encarece, sin ninguna ventaja aparente, por lo que hemos descartado estas alternativas para el proyecto.

2.1.4. Elección final

Valorando las ventajas y desventajas de cada tecnología, la limitación de Sigfox tanto en la cantidad de paquetes como en el tamaño, hace que la solución sea poco flexible de cara a añadir sensores al dispositivo en el futuro. Suponiendo que utilizásemos un protocolo en el que reservemos 1 byte para tener 8 *flags*[9] que indiquen si se incluye la lectura de cada sensor en el paquete, o no, y utilizamos 2 bytes (16 bits) por cada dato, para 3 sensores ocuparíamos 7 bytes. Si añadimos dos bytes más, uno para el nivel de batería, y otro para el nivel de calidad de señal, ocuparíamos 9 de los 12 posibles. Lo cual deja muy limitada la expansión de nuevos sensores. Tampoco nos gusta depender de una empresa que tiene el monopolio de la red, puesto que en cualquier momento podría subir los precios, o prestar un peor servicio y la única solución sería volver a diseñar, fabricar y desplegar de nuevo todos los dispositivos de la red.

8 ISM: Bandas de radio reservadas para aplicaciones industriales, científicas y médicas, sin necesidad de pagar por su uso, siempre que se cumplan unos determinados criterios sobre el ciclo de trabajo para no bloquear continuamente las comunicaciones del resto de dispositivos del mismo tipo. Ejemplos comunes de bandas ISM en Europa son la banda de 2.4GHz utilizada por el wifi, o la banda de 868Mhz, con mejores condiciones de propagación a larga distancia.

9 «Un *flag* es uno o más bits que se utilizan para almacenar un valor binario que tiene asignado un significado.» [13]

La alternativa idónea sería NB-IoT y/o LTE-M, pero como hemos comentado la cobertura en España, aunque cada vez es mayor, todavía no es completa. Por ello hemos buscado un módulo de conectividad móvil capaz de conectarse a redes NB-IoT y LTE-M, pero con respaldo GSM para poder funcionar en aquellos lugares donde solo existe cobertura GSM, hasta que las nuevas redes lleguen a todo el territorio. Creemos que esta opción, junto con una SIM con un plan de datos específico para IoT es una solución económica, ya que el precio de los módulos de este tipo se encuentra entre 15-20€, y el precio de la conectividad por unos pocos MB es de menos de 1€ al mes. Además es una solución muy flexible, puesto que permite la adición de nuevos sensores, o el aumento de la frecuencia de transmisión de los datos, si fuese necesario.

2.2. Protocolos de aplicación: HTTP y MQTT

Necesitamos encontrar un protocolo de aplicación para el envío de pequeños paquetes de datos que contengan la lectura de los sensores. Debemos atender a criterios de eficiencia energética, para prolongar la autonomía del dispositivo, y de consumo de datos, los cuales cuestan dinero.

2.2.1. HTTP

Una alternativa para nuestro proyecto es disponer de una ruta en un servidor HTTP, que acepte peticiones POST. En esa ruta enviaríamos las peticiones desde los dispositivos remotos con los datos de los sensores en formato JSON[10], y la información se procesaría en el servidor para su almacenamiento.

Basándonos en la RFC 7230 [15], HTTP es un protocolo de la capa de aplicación de tipo cliente-servidor, para sistemas de información de hipermedia. Es un protocolo sin estado, es decir, trata cada petición independientemente de las peticiones anteriores. Y es el protocolo utilizado para el intercambio de información de la World Wide Web[11], así que da vida a los millones de páginas web accesibles a través de Internet. HTTP utiliza métodos de petición específicos para realizar diferentes acciones, algunos de estos son GET, para solicitar un recurso, POST, para añadir contenido o datos bajo un recurso existente o DELETE, para eliminar el recurso especificado.

10 «JSON es un formato ligero de intercambio de datos. Fácil de leer y escribir para humanos, y sencillo de interpretar y generar para máquinas.» [14]

11 https://developer.mozilla.org/es/docs/Glossary/World_Wide_Web

Normalmente las comunicaciones HTTP se realizan sobre conexiones TCP/IP[12], lo que permite comunicaciones fiables. El modo de funcionamiento consiste en que el cliente establece una conexión TCP con el servidor, y realiza una o varias peticiones de recursos, antes de cerrar la conexión TCP. Desde la versión HTTP 1.1, esta conexión puede ser persistente, lo que permite realizar múltiples peticiones al servidor mediante las URI de diferentes recursos antes de cerrar la conexión TCP, para ahorrar tráfico de datos y reducir la latencia. Normalmente los servidores de HTTP 1.1 tienen un valor de timeout configurable, a partir del cual no mantienen abierta una conexión persistente inactiva.

2.2.2. MQTT

Según su especificación OASIS [18], MQTT es un protocolo cliente-servidor de la capa de aplicación de tipo *publish/subscribe*. Es ligero, sencillo y diseñado para ser fácil de implementar. Estas características lo hacen ideal para el uso en entornos limitados en recursos computacionales y de red, como comunicaciones *Machine to Machine* (M2M) e *Internet of Things* (IoT), donde es necesaria una huella de código pequeña y el ancho de banda de red tiene un coste. El protocolo es agnóstico acerca del contenido del *payload*, por lo que permite enviar datos en formato texto o en binario.

El patrón de mensajería *publish/subscribe* consiste en que cuando alguien publica un mensaje no tiene que conocer al receptor, y no se lo envía a él, sino que lo envía a un broker, con el que tiene una conexión establecida previamente. Al publicar un mensaje, indica el *topic* en el que quiere registrar el mensaje y es el broker el que se encarga de hacer llegar el mensaje a todos aquellos clientes que están conectados, y suscritos al *topic* en cuestión. La forma de funcionamiento típica es sobre TCP/IP, estableciendo una conexión de larga vida para mantenerla abierta durante días, semanas o incluso meses, y así enviar el mínimo posible de datos por la red.

Por otro lado, MQTT establece tres niveles de calidad del servicio (QoS): "*at most once*", con el cual se acepta la pérdida de mensajes, cada uno se envía una sola vez sin realizar reenvíos. El nivel "*at least once*", que asegura la recepción del mensaje, pero podría enviar mensajes duplicados. Y el nivel "*exactly once*", que asegura que cada mensaje va a llegar, y que no va a llegar duplicado. Estos niveles de QoS tienen un coste de red progre-

12 Se refiere al uso conjunto del protocolo IP en la capa de red, y el protocolo TCP en la capa de transporte.
TCP: Es un protocolo de transporte orientado a transportar paquetes de manera fiable, ordenada y sin errores. [16]
IP: Es un protocolo de red encargado de transportar paquetes desde el origen hasta el destino. No es orientado a conexión, no tiene mecanismos de control y está basado en datagramas. En cada datagrama se incluye una cabecera con la dirección de destino, y los routers de la red se encargan de encaminar el paquete hacia el propietario de esa dirección. [17]

sivamente mayor conforme el nivel es más exigente por los mensajes de señalización que implican. En nuestro caso, es aceptable el nivel más bajo de QoS, ya que se enviarán lecturas de los sensores cada pocos minutos, por lo que no es crítico que se pierda algún mensaje con la lectura de los sensores, pero sí es muy importante tratar de reducir el consumo de datos en la red.

2.2.3. Comparativa

Pasamos a enumerar los puntos en común y diferencias principales entre MQTT y HTTP:

- MQTT es un protocolo binario desde su origen, lo que permite compactar las cabeceras y el *payload*, mientras que HTTP, es originalmente un protocolo textual, aunque a partir de la versión 2 también se convirtió en un protocolo binario. En este punto están igualados.
- HTTP tiene un modelo cliente-servidor de petición-respuesta en el que el servidor no puede realizar peticiones a un cliente. Por el contrario el modelo publicación-suscripción de MQTT permite enviar y suscribirse a mensajes por cualquier dispositivo, lo que facilita la descarga de configuración por los dispositivos sin necesidad de estar sondeando periódicamente un servidor a la espera de cambios. En este sentido MQTT es más adecuado para nuestra aplicación.
- Aunque tanto MQTT como HTTP permiten conexiones persistentes, MQTT incluye un mecanismo de latidos para mantener una conexión abierta si no se envía ningún mensaje con datos, para evitar que expire el *timeout*. Tanto la petición como la respuesta de este mecanismo de latido tienen un peso de 2 bytes + las cabeceras TCP/IP, por lo que son más ligeros que un mecanismo alternativo que pudiéramos implementar sobre HTTP, y además no tenemos que desarrollarlo nosotros.

  Además, los servidores HTTP, por lo general, suelen destinar un *thread* por cada conexión TCP establecida. Aunque es viable computacionalmente mantener miles de conexiones TCP abiertas, mantener miles de hilos y procesos del sistema abiertos compitiendo por la CPU y la memoria no es viable [19]. Por su parte, los servidores MQTT están pensados para mantener miles o incluso millones de conexiones concurrentes en un único servidor [20]. Por lo que, aunque teóricamente es posible utilizar HTTP con conexiones persistentes, si éstas van a estar abiertas continuamente, y tenemos cientos o miles de dispositivos, en la práctica los brokers

MQTT están mucho más preparados para este caso de uso, por lo que MQTT tiene ventaja en este punto.

- En cuanto al *overhead* de las cabeceras de ambos protocolos, MQTT es más ligero, ya que la cabecera mínima de una petición HTTP es de 26 bytes, mientras que en MQTT el *overhead* de publicar un mensaje con QoS 0 es de 15 bytes.
- En múltiples comparaciones entre HTTP y MQTT, como por ejemplo [21]–[23] podemos ver que MQTT disminuye el tiempo de respuesta y reduce el número de bytes en la red por cada mensaje enviado. Además reduce el consumo de energía, lo cual además de en el menor tamaño de las cabeceras, puede encontrar su explicación en que es un protocolo más ligero y cuya implementación es computacionalmente más sencilla, lo que permite que los dispositivos con restricciones en la capacidad de cómputo sean más eficientes enviando datos con el protocolo MQTT.
- MQTT tiene dos características únicas que son los diferentes niveles de QoS gracias a lo cual podemos publicar los mensajes más importantes con un nivel de calidad del servicio mayor, y los más redundantes (de datos enviados cada pocos segundos o minutos), con un nivel de calidad del servicio menor. La otra característica única es que al establecer una conexión con un broker MQTT, se puede establecer un mensaje de “última voluntad”, que se registrará automáticamente en el caso de la desconexión repentina de un cliente, lo cual permite monitorizar más fácilmente el estado de la conexión de los dispositivos.

En la Tabla 1 se muestra un resumen de la comparativa entre ambos protocolos. Por todo lo comentado anteriormente, vemos más ventajas al uso de MQTT frente a HTTP, por lo que el protocolo de aplicación elegido para nuestro sistema es MQTT.

Característica	HTTP	MQTT
Peticiones cliente-servidor	Unidireccional, solo del cliente al servidor	Bidireccional
Manejo de conexiones persistentes en el servidor	Posibles, pero no para miles de dispositivos	Pensado para ello. Cientos de miles de conexiones.
Overhead de datos	Intermedio	Más ligero
Tiempo de respuesta	Mayor	Menor
Consumo de energía	Mayor	Menor
Nivel de QoS	No tiene	3 niveles para elegir en función de la importancia de cada mensaje
Mecanismo de mensaje de última voluntad	No tiene	Configurable al establecer la conexión

Tabla 1: Resumen de la comparación entre HTTP y MQTT

3. Diseño e implementación

En este capítulo detallamos la arquitectura general del sistema, el software utilizado y describimos el proceso de diseño y desarrollo de cada una de sus partes por separado: dispositivo electrónico, sistema de información e interfaz de usuario.

3.1. Arquitectura general

Disponemos de un servidor conectado a Internet, en el que tenemos diversos servicios funcionando mediante contenedores Docker[1]:

- Una base de datos PostgreSQL para el almacenamiento de la información.
- Un broker MQTT, que es al que se conectan los dispositivos electrónicos a través de Internet para enviar la lectura de los sensores.
- Una aplicación Node.js[2], que incluye un cliente MQTT que se conecta al broker MQTT para recibir los datos de los sensores y almacenarlos en la base de datos. Esta aplicación Node.js también tiene una API REST a la que se conecta la aplicación web para solicitar la información que necesita en cada momento.
- Un proxy inverso[3] con el software Nginx, que es al que los usuarios del sistema realizan las peticiones web. Éste se encarga por un lado de servir los ficheros estáticos (HTML, JavaScript y CSS) que dan forma a la aplicación web React, y por otro lado redirige las peticiones a la dirección "/api/*" hacia la API REST de la aplicación Node.js.

Fuera del servidor, tenemos nuestro dispositivo electrónico con sensores, que se conecta a Internet mediante la red de telefonía móvil. Este dispositivo establece una conexión de larga duración con el broker MQTT, y le envía periódicamente un mensaje con las lecturas de los sensores, el porcentaje de batería y la calidad de la señal. En la Figura 1 se muestra un diagrama que ayuda a comprender la arquitectura del sistema de un vistazo.

1 Software de virtualización de aplicaciones mediante contenedores.
2 Entorno de ejecución de JavaScript en el lado del servidor.
3 «Es un tipo de servidor proxy que recupera recursos en nombre de un cliente desde uno o más servidores. Estos recursos se devuelven al cliente como si se originaran en el propio servidor web. A menudo se utilizan protegiendo los framework de aplicación de las debilidades de seguridad de HTTP.» [24]

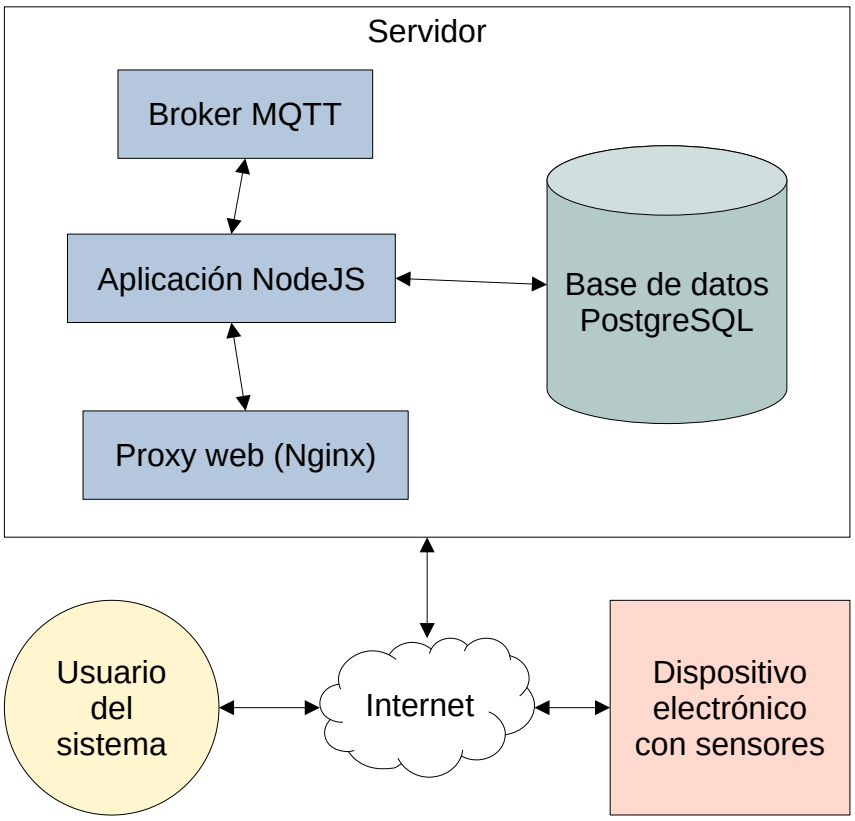

Figura 1: Diagrama de arquitectura

3.2. Software utilizado

Aquí presentamos una lista del principal software utilizado a lo largo del capítulo de diseño e implementación:

- **KiCad:** Software de código abierto para diseño de esquemáticos y PCB electrónicas.
- **Mbed Studio:** IDE[4] de desarrollo de software embebido para microcontroladores con núcleos ARM[5] con el sistema operativo ARM Mbed.
- **Visual studio code:** IDE de desarrollo web.
- **Docker:** Software de virtualización de aplicaciones mediante contenedores. Permite compartir y ejecutar entornos de desarrollo fácilmente, de forma mucho más ligera y eficiente que las máquinas virtuales tradicionales, al funcionar directamente so-

4 «Un IDE es una aplicación informática que proporciona servicios para facilitar el desarrollo de software.» [25]
5 Es una arquitectura de procesdores desarrollada por la empresa ARM Holdings. ARM es el conjunto de instrucciones de 32 y 64 bits más utilizado en unidades producidas. [26]

bre el *kernel* del sistema operativo host, en lugar de emular hardware y ejecutar un sistema operativo completo como en una máquina virtual [27].

- **Postman**: Permite realizar peticiones HTTP para testear API REST de forma sencilla y configurable. Útil para testear peticiones POST, PATCH, o DELETE, las cuales son difíciles de probar con un navegador web convencional, pero también GET o de cualquier otro tipo, ya que permite guardar las peticiones y ejecutarlas por lotes.
- **DBeaver:** Cliente de bases de datos SQL de código abierto, para la monitorización de la base de datos, y poder realizar consultas de prueba, o modificar registros manualmente.

3.3. Dispositivo electrónico

En esta sección detallamos la elección de una plataforma de desarrollo y las pruebas con la misma (3.3.1), la arquitectura de hardware diseñado (3.3.2) en base al cual construiremos una PCB, el diseño del esquemático del circuito necesario para obtener la funcionalidad que buscamos (3.3.3), cómo se ha llevado a cabo el diseño de las placas de circuito impresas (3.3.4), y la fabricación y montaje de componentes de las mismas (3.3.5). Por último, una explicación del diseño, implementación, y funcionamiento del software embebido en el dispositivo (3.3.6).

3.3.1. Plataforma de desarrollo

Tras haber trabajado con la plataforma Arduino con microcontroladores de 8 bits[6], conocer sus limitaciones, y buscando principalmente una plataforma que permitiera una mejor gestión de multitarea en *threads*, se barajó la posibilidad de usar el entorno Arduino con microcontroladores de 32 bits y núcleos ARM Cortex (como el Arduino Zero), o usar la serie de microcontroladores STM32 de STMicroelectronics con el sistema operativo ARM Mbed. Finalmente se optó por ARM Mbed frente a Arduino por ser un producto más completo, con librerías oficiales para la gestión de *threads*, entre otras cosas, y más orientado a desarrollar productos para llevarlos a producción, mientras que Arduino está más orientado a aficionados de la electrónica.

6 Los bits de un procesador determinan el tamaño de los registros y los números más grandes que pueden procesar en una sola operación. Esto implica que un MCU (microcontrolador) de 32 bits puede procesar datos grandes de manera más rápida que con 8 (con 8 bits solo podemos representar números enteros entre 0 y 255, y no pueden manejar números en coma flotante de forma nativa). Además los MCU de 32 bits suelen operar con frecuencias de reloj más rápidas, por lo que también son más rápidos que los de 8 bits. Sin embargo, los MCU de 8 bits siguen siendo útiles y baratos para tareas simples. [28]

Para realizar una primera aproximación del dispositivo se trabajó con la placa de desarrollo STMicroelectronics Nucleo L476RG que se muestra en la Figura 2. Se configuró el IDE Mbed Studio para cargar software a esta placa, y se probaron sus funciones realizando un montaje con el shield de Arduino para comunicaciones GSM y NB-IoT que se muestra en la Figura 3.

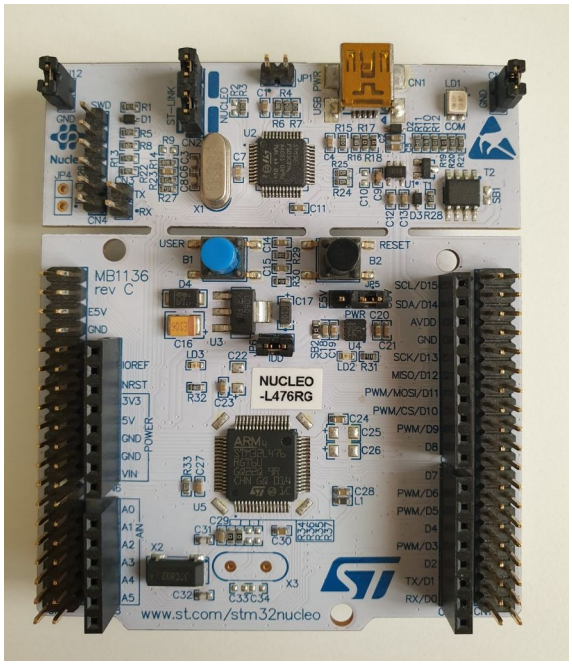

Figura 2: Placa de desarrollo Nucleo L476RG

Figura 3: Shield DFRobot SIM7000

Se desarrolló un driver para las comunicaciones entre el microcontrolador y el módulo SIM7000E haciendo uso de la librería Serial de ARM Mbed, con los comandos básicos para configurar el dispositivo, comprobar que se conecta a la red móvil, activar la conexión de datos GPRS, conectarse a un servidor MQTT de test, y enviar paquetes de datos al servidor MQTT teniendo una conexión abierta.

Posteriormente se modificó este driver basándonos en las librerías HAL[7] de STMicroelectronics del periférico LPUART[8] del microcontrolador en lugar de la librería Serial de ARM Mbed, puesto que ésta última no soporta los modos de baja energía de LPUART, ni mantener el dispositivo "dormido" y despertarlo a la llegada de nuevos datos por el puerto LPUART para su procesamiento.

7 Las librerías HAL (Hardware Abstraction Layer) de STMicroelectronics son librerías de alto nivel diseñadas por el fabricante para ofrecer una interacción más sencilla con los periféricos del microcontrolador, sin perder las principales opciones de configuración que tendríamos con las librerías de bajo nivel. [29]

8 *Low Power Universal Asynchronous Receiver/Transmitter*: es una interfaz de comunicación en serie, similar al UART pero capaz de funcionar a 9600 bps con un reloj externo de baja velocidad (32.768 kHz), lo cual permite reducir el consumo de energía del periférico. [30]

Una vez que teníamos funcionando un programa que hacía uso del módem SIM7000E para conectarse al broker MQTT de test de Eclipse Mosquitto[9] y enviar un mensaje periódicamente, se procedió a diseñar un circuito para fabricar una PCB que se ajustase de la mejor manera a los requisitos de nuestro dispositivo.

3.3.2. Arquitectura de hardware

En la Figura 4 vemos el diagrama de los bloques funcionales más importantes de la placa de circuito que hemos diseñado. Las líneas rojas denotan conexiones de alimentación, y las azules, conexiones de datos. Tenemos un receptáculo para una batería recargable extraíble de Li-Ion, de tipo 18650[10], conectado a un circuito de protección para evitar daños en el circuito y en la propia batería, si ésta se introduce con la polaridad invertida, y un circuito que gestiona la carga de la batería mediante una placa solar.

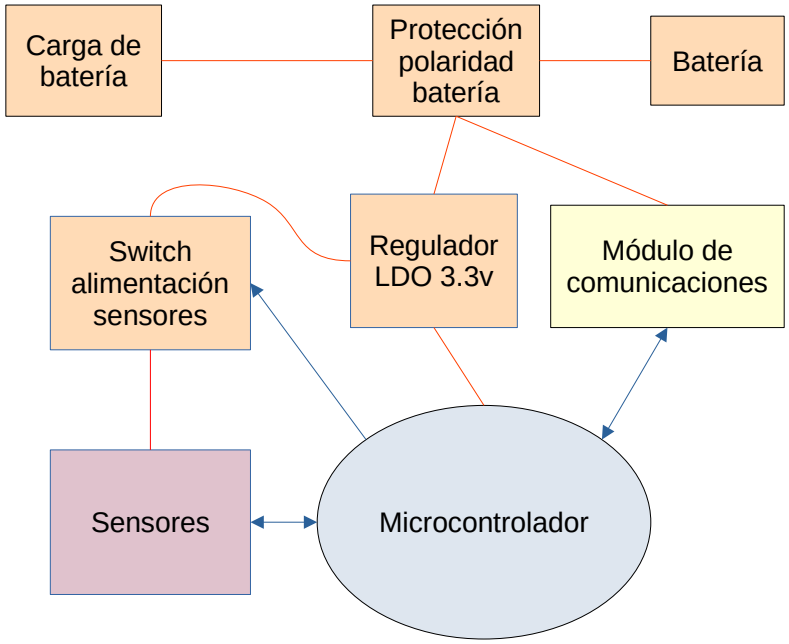

Figura 4: Diagrama de bloques

De la salida del circuito de protección de polaridad obtenemos un voltaje nominal de 3.6v, que llega hasta 4.2v cuando la batería está completamente cargada, que alimenta directamente al módulo de comunicaciones (que acepta alimentación entre 3.0V y 4.3V), y al re-

9 https://test.mosquitto.org/

10 El factor de forma 18650 de baterías recargables, generalmente de Li-Ion, es un estándar, como pueden ser AA o AAA. Recibe su nombre de las dimensiones de este tipo de baterías, cuya forma es un cilindro de 18mm de diámetro y 65mm de longitud. [31]

gulador de tensión LDO[11] con salida a 3.3v, puesto que éste es el voltaje al que se alimentan la mayoría de los componentes del circuito.

Del regulador de tensión se alimenta el microcontrolador, y tenemos una salida para la alimentación de los sensores que pasa por un *switch* digital, el cual permite cortar el paso de corriente para ahorrar energía cuando no se van a obtener lecturas de los sensores.

El microcontrolador tiene diversas conexiones de datos (lectura y escritura de pines digitales, I2C[12], UART) con el módulo de comunicaciones, con los sensores y con el control de alimentación de los sensores.

3.3.3. Diseño del esquemático

El diseño del circuito y la PCB se ha realizado con el software KiCad, el cual es Open Source, e incluye herramientas para realizar el esquemático, la PCB y mantenerlos sincronizados. En la Figura 5 se ve el detalle del editor de esquemático de KiCad. Se pueden encontrar más detalles de diseño del esquemático en el anexo “Detalles de diseño del dispositivo electrónico”.

11 Un regulador de tensión LDO (*Low Drop-Out*), o regulador de baja caída es un regulador lineal de corriente continua que puede regular el voltaje de salida cuando el voltaje de alimentación está muy cerca del de salida. La ventaja respecto a otras tipologías de reguladores es que no introducen ruido de conmutación, son más pequeños y su diseño es más simple. Por contra, disipan potencia proporcionalmente a la diferencia de voltaje entre la entrada y la salida. Por eso solo son apropiados para circuitos con ambos voltajes cercanos o de baja potencia. [32]

12 «I2C (*Inter-Integrated Circuit*) es un bus serie de datos desarrollado por Philips Semiconductors. Se utiliza principalmente para la comunicación interna entre diferentes partes de un circuito, por ejemplo, entre un controlador y circuitos periféricos integrados.» [33]

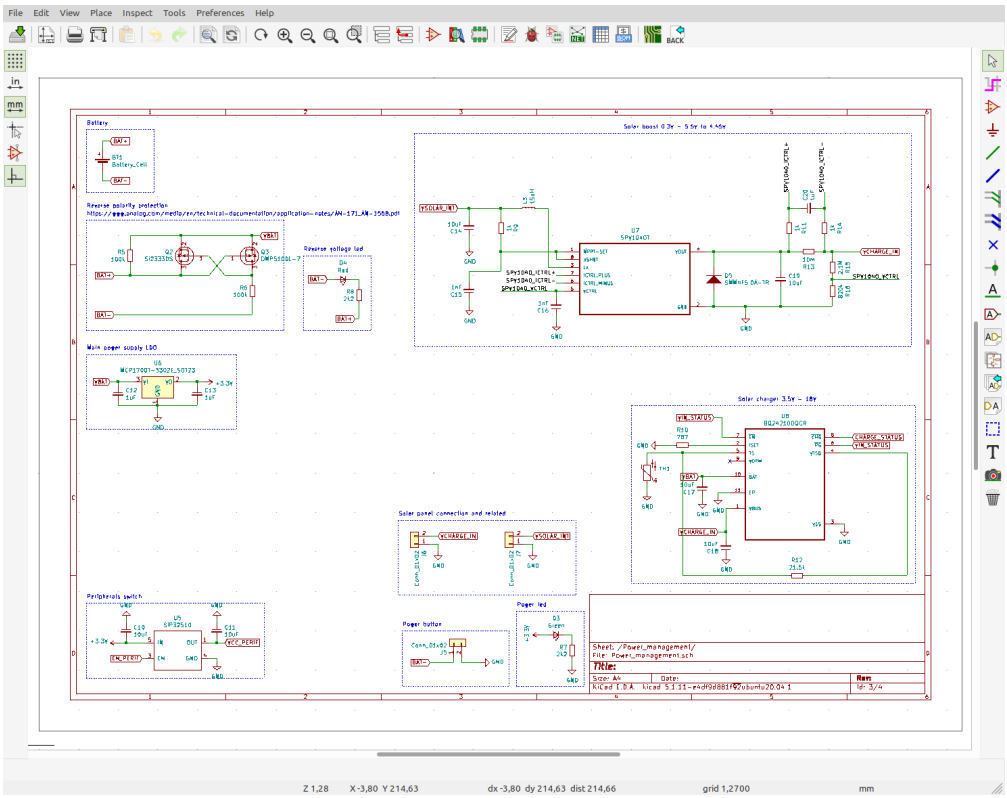

Figura 5: Detalle de la interfaz del CAD de esquemático de KiCad

3.3.4. Diseño de la PCB

Una vez terminado el diseño del esquemático se procedió a darle forma a una PCB basada en el diseño previo. Se realiza con componentes de montaje de superficie (SMD[13]) puesto que se dispone de un horno de soldadura y pistola de calor, con el objetivo de conseguir un dispositivo suficientemente pequeño para instalarlo en una carcasa eléctrica de 125mm*125mm.

Se consigue una placa de circuito de 100mm*80mm incluyendo en este el receptáculo de la batería. En la Figura 6 se muestra un render 3D de la PCB diseñada, obtenido con KiCad.

13 Los componentes electrónicos de montaje superficial (SMD) no atraviesan la placa de circuito impreso, las conexiones se realizan mediante contactos planos. Es la tecnología de montaje más utilizada actualmente. [34]

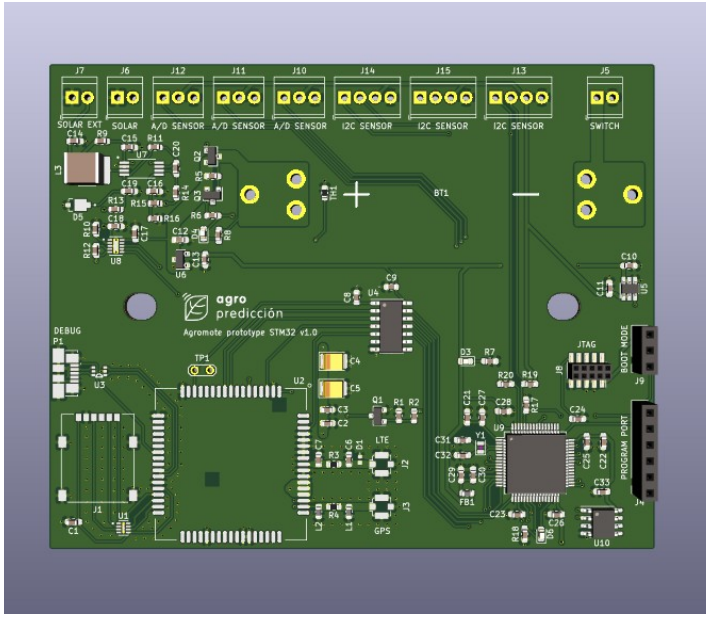

Figura 6: Render 3D de la PCB diseñada

3.3.5. Fabricación y montaje de la PCB

Con la PCB ya diseñada, se encargó su fabricación a la empresa china JLCPCB[14]. Junto con las PCB se encargó una plantilla (stencil[15]) para la pasta de soldadura. El resultado de la placa fabricada se muestra en las Figuras 7 y 8.

Se encargaron los componentes para el montaje de una PCB a las tiendas Mouser, Farnell, TME y LCSC. Se aplicó la pasta de soldadura con el *stencil*, y se colocaron los componentes SMD con unas pinzas. Tras soldarlos con el horno de soldadura T-962, se procedió a soldar los componentes de agujero pasante (THT[16]) con un soldador de la marca Baku.

14 https://jlcpcb.com/

15 Una plantilla de soldadura o "*paste stencil*" es una plantilla agujereada de forma precisa con las formas de los *pads* de una PCB en los que se van a situar componentes de montaje en superficie (SMD). Sirve para poder aplicar la pasta de soldadura en los lugares exactos y en la cantidad necesaria en los *pads* de la PCB. Se sujeta el *stencil* a la PCB para que los agujeros coincidan y no se mueva, y se aplica la pasta de soldadura con una espátula o una tarjeta de plástico. Después se retira la plantilla, y se sitúan los componentes en su lugar. Una vez hecho esto, se hornea la PCB con los componentes y la pasta de soldadura se funde y se solidifica dando lugar a las soldaduras en todos los *pads* de los componentes. [35]

16 La tecnología de agujeros pasantes es una tecnología de montaje de componentes en placas de circuito impreso, que utiliza los agujeros practicados en las placas de circuito para el montaje de los elementos electrónicos, creando puentes eléctricos entre ambas caras de la placa de montaje mediante un tubo conductor. [36]

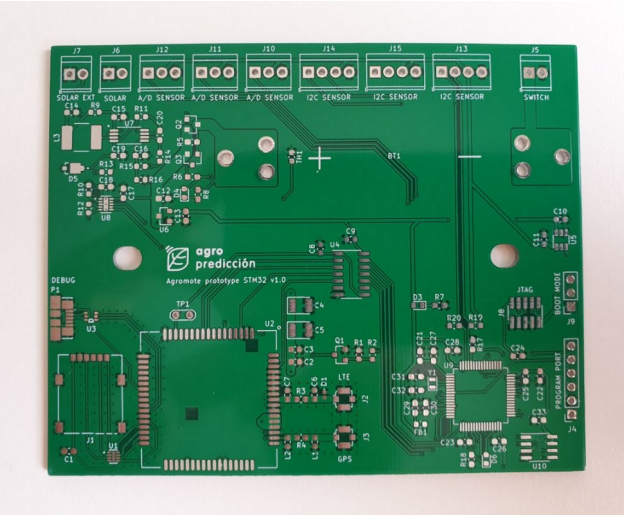

Figura 7: Vista anterior de la PCB fabricada

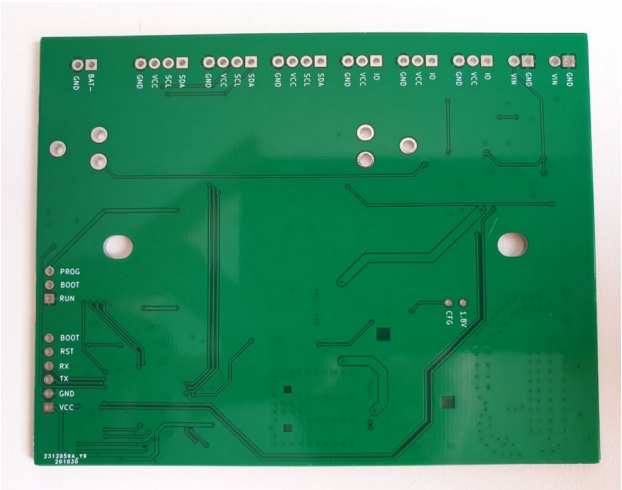

Figura 8: Vista posterior de la PCB fabricada

Posteriormente durante las pruebas se detectó un error en la huella del conector de debug del microcontrolador, que es el que permite cargar el programa. Se encontraba con los pines en espejo respecto a cómo deberían estar, por lo tanto hubo que soldar unos pequeños cables para poder trabajar con la placa. En la Figura 9 se muestra el resultado de la placa de circuito impreso con los componentes montados y con su batería, nano sim, debugger, conversor serial-usb, y los sensores conectados.

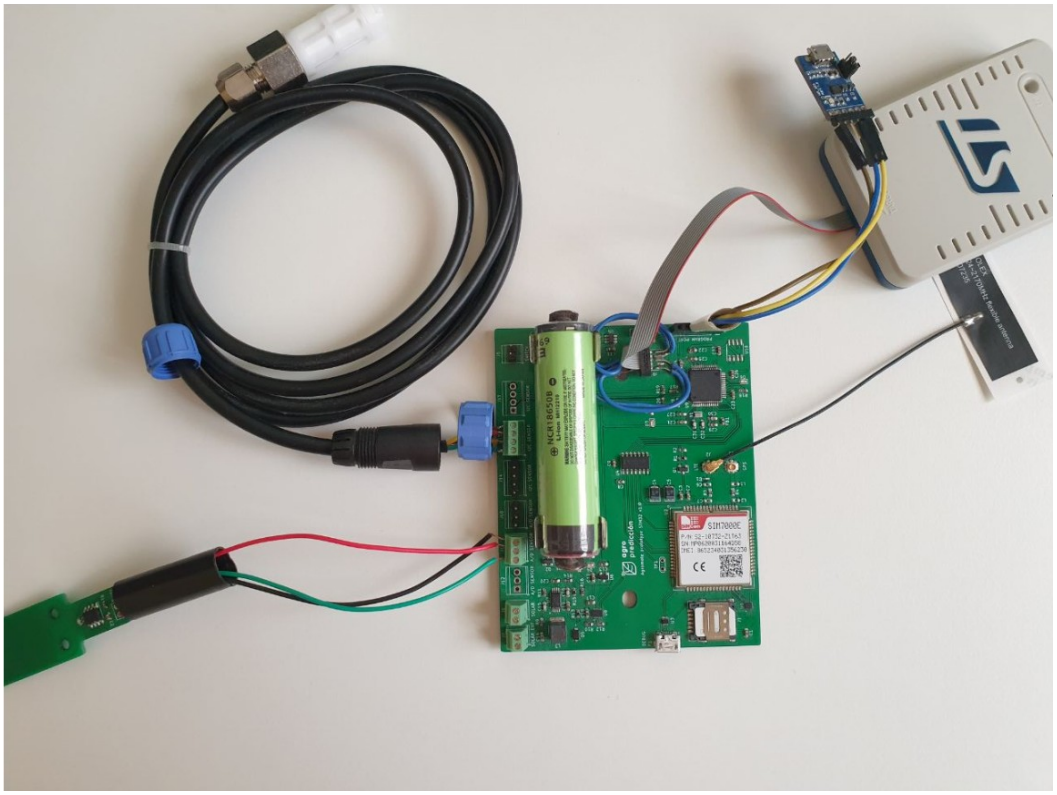

Figura 9: PCB con los componentes montados.

3.3.6. Software embebido

Tomando como base los archivos de definición de un microcontrolador de la misma familia que el utilizado (STM32L431RCT) se ha definido el mapa de pines y periféricos necesario para poder compilar el software para nuestro dispositivo con el sistema operativo ARM Mbed.

Manejo del módem SIM7000. Se ha desarrollado un driver para el periférico LPUART, con el cual configuramos este periférico para ser capaz de funcionar en modo “stop” del microcontrolador (uno de los modos de más bajo consumo), y despertar el dispositivo a la recepción de datos por el pin RX de este periférico. Este driver incluye una función para transmitir datos por el pin TX del periférico LPUART1, y recibe los datos por el pin RX en modo interrupción, pasando cada carácter a una función que pasamos como parámetro en la inicialización del driver.

Basada en el driver para el puerto LPUART, tenemos una clase ATHandler que abstrae los detalles de implementación para manejar comandos AT (el tipo de comandos que acepta el módem SIM7000). La clase ATHandler se encarga de inicializar el driver LPUART. También incluye un método para enviar un comando AT de forma configurable según el tipo, un método para esperar hasta recibir una respuesta AT determinada y un método para enviar un comando AT y devolver la respuesta recibida. Si se recibe un mensaje no esperado en cualquier momento (por ejemplo un aviso de SMS, o un mensaje MQTT del lado del servidor), se almacena en un buffer y se marca un flag para que el thread encargado de esta tarea procese el mensaje adecuadamente, que por el momento solamente se encarga de enviar el mensaje recibido por el puerto UART de depuración (debug).

Haciendo uso de la clase ATHandler tenemos una clase SIM7000 que implementa métodos para el envío de comandos al dispositivo, así como la gestión del encendido, apagado, dormir y despertar el dispositivo mediante los pines correspondientes. Incluye métodos para activar la conexión GPRS del módem, para leer el IMEI, obtener el método de conexión actual (sin servicio, GSM, LTE-M1, NB-IoT…), la calidad de la señal (RSSI[17]) o el porcentaje de carga de la batería conectada al módem, entre otras funciones. Esta clase SIM7000 es la interfaz para manejar el módem en nuestro programa principal.

17 RSSI: «El indicador de fuerza de la señal recibida es una escala de referencia para medir el nivel de potencia de las señales recibidas por un dispositivo en las redes inalámbricas.» [37]

Manejo del sensor SHT31. Basado en la librería *Open Source* para Arduino de la empresa Adafruit para este mismo sensor, y en el datasheet [38] del mismo, se ha desarrollado una clase que permite leer la temperatura y humedad del sensor, mediante el protocolo I2C.

Manejo del sensor de humectación de hoja. (Ver el anexo "Diseño de un sensor de humectación de hoja") La salida de este sensor, por diseño, es una señal cuadrada de frecuencia variable, en función de la cantidad de agua que se encuentra en la superficie del sensor, que es lo que pretendemos medir. Para medir la frecuencia de una señal cuadrada, tenemos el sensor conectado a un pin digital del microcontrolador. Y configuramos una interrupción[18] de subida en ese pin. Por lo que cada vez que el voltaje pasa de 0 a 3.3v se ejecuta la función de interrupción. Configuramos un timer de 500ms y la función de interrupción incrementa un contador. A la finalización del timer de 500ms, con el resultado del contador podemos obtener la frecuencia de la señal del sensor.

Para calibrar el sensor realizamos la medida con el sensor completamente seco, obteniendo 4680 Hz, y la medida con todas las pistas interdigitales del sensor sumergidas en un vaso de agua, obteniendo 2300 Hz. Por lo que obtenemos el porcentaje de humectación del sensor como 0% si la frecuencia es mayor de 4680 Hz, 100% si la frecuencia es menor de 2300 Hz. Si la lectura se encuentra entre estos dos valores, el porcentaje de humectación es el resultado de: (4680 – frecuencia) * 100 / (4680 – 2300)

Funcionamiento de la aplicación. La aplicación se organiza en *threads* independientes. Por un lado tenemos un *thread* que se encarga de encender un led durante 10ms (suficiente para ser visible) a intervalos de 3000ms para permitir detectar visualmente que el dispositivo está en funcionamiento. Se realiza de esta forma, con un parpadeo tan rápido, para tener un impacto mínimo en el consumo de energía. El ciclo de trabajo de este led es del 0,33%, lo cual para un led de 5mA de corriente consumida daría una media de 16,5 uA.

Por otro lado tenemos el thread principal, que enciende el switch digital para permitir el paso de la corriente que alimenta los sensores, espera 500ms para que los sensores se

18 «En informática, una interrupción es una señal recibida por el procesador, que le indica que debe interrumpir el curso de ejecución actual y pasar a ejecutar código específico para tratar esa situación.» [39]

estabilicen, y realiza una lectura de humedad, temperatura y humectación de hoja. Después apaga el switch digital de la alimentación de los sensores para ahorrar energía. Con estas medidas guardadas, despierta el módem, y obtiene de éste la lectura del porcentaje de la batería y el RSSI para conocer la calidad de la señal. Una vez hecho esto envía un mensaje MQTT al servidor configurado con todos los elementos leídos (temperatura, humedad, humectación de hoja, batería y calidad de señal) en formato JSON. La primera vez establece la conexión con el servidor MQTT. En las lecturas sucesivas reutiliza la misma conexión anterior para ahorrar batería y datos. Una vez terminado este proceso, se duerme el módem para ahorrar energía, y pone el thread en espera de la siguiente señal del timer, que está configurada cada 10 minutos. Por lo que la mayor parte del tiempo el dispositivo está en un modo de baja energía, ya que el sistema operativo duerme el dispositivo automáticamente si no lo usa ningún thread.

Además de los dos *threads* mencionados, tenemos un tercero para *debug* que se encarga de mostrar las estadísticas de tiempo en modo de energía *awake* (funcionando), *sleep*, y *deep sleep* (modos de sueño de bajo y muy bajo consumo respectivamente). Así como información sobre el tamaño del *heap*[19] de memoria, para controlar que no se desborde.

Carga de código al dispositivo. La carga del código al dispositivo se realiza desde el IDE Mbed Studio, mediante el dispositivo debugger STLink V3 del mismo fabricante que el microcontrolador (STMicroelectronics), a través del puerto de debug reservado en la placa a tal efecto. Por otro lado, en la placa de circuito tenemos un conector para UART, al cual tenemos conectado un conversor de USB a UART para monitorizar desde el ordenador, por ese puerto, los mensajes de salida del programa mediante el método “printf()”. En la Figura 10 se observa el detalle de la interfaz del IDE utilizado para el desarrollo y la carga de código al dispositivo electrónico.

19 El *heap* es la región de memoria dinámica donde se almacenan los datos que se crean durante la ejecución de un programa. Si ésta crece indefinidamente significa que la memoria no se está liberando correctamente. [40]

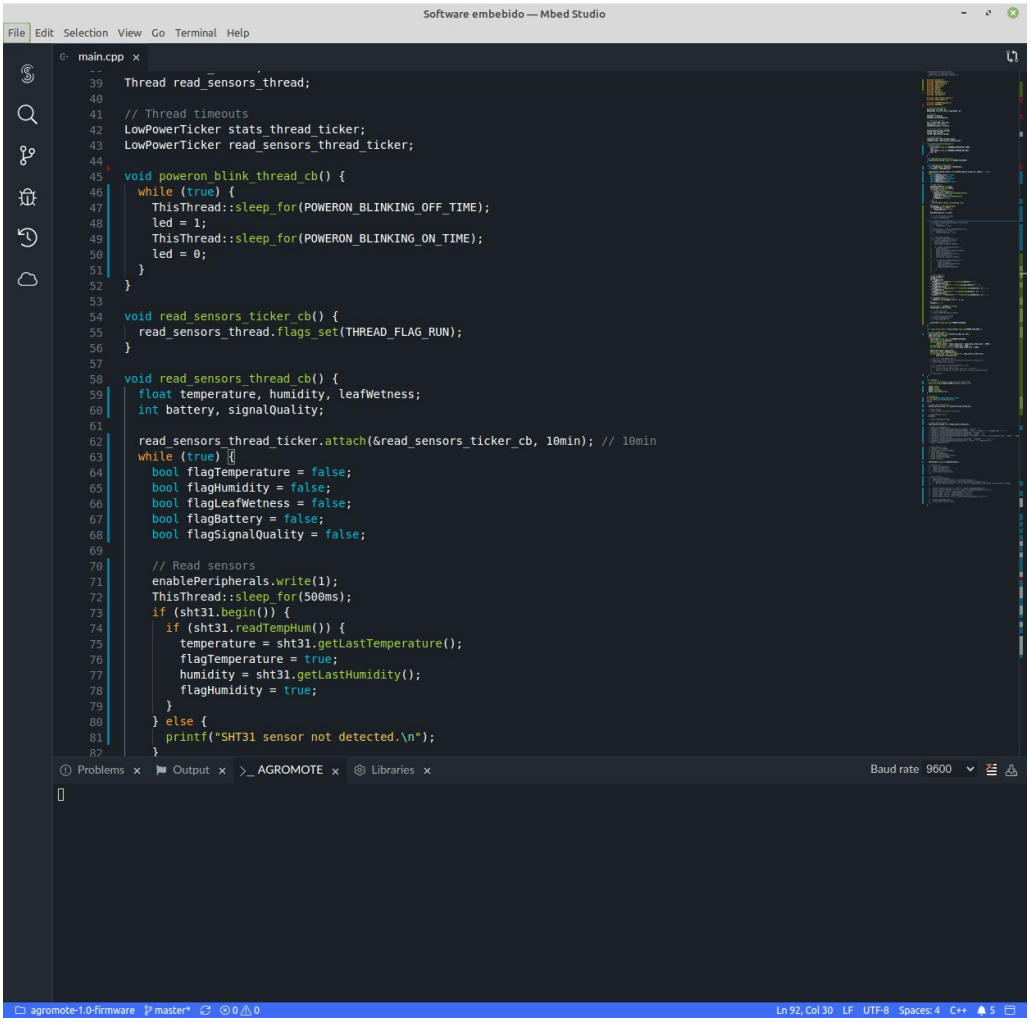

Figura 10: Interfaz del IDE Mbed Studio

3.4. Sistema de información

En esta sección explicamos las tecnologías, frameworks y librerías utilizadas más relevantes para el desarrollo del sistema de información (3.4.1), la configuración del entorno de desarrollo (3.4.2) y algunos detalles de la implementación del servidor Node.js (3.4.3).

3.4.1. Tecnologías utilizadas

Node.js. Es un entorno de código abierto basado en el motor de ejecución JavaScript V8 de Google, que permite la ejecución de programas escritos con este lenguaje fuera del navegador [41]. Puede utilizarse para aplicaciones de línea de comandos, para ejecutar código web en el lado del servidor, sustituyendo la función típica de PHP[20], o incluso para

20 «Lenguaje de programación de uso general, principalmente utilizado para desarrollo web. Su código se ejecuta en el lado del servidor.» [42]

aplicaciones de escritorio mediante el *framework* Electron[21]. En este caso lo utilizamos como lenguaje para un servidor web en el que desarrollamos una API REST.

NestJS. Es un framework de código abierto basado en Node.js, aunque permite código JavaScript, está pensado para su uso con TypeScript[22]. Utilizamos NestJS con su configuración por defecto, con Express.js como framework de servidor HTTP y escribiremos el código con TypeScript. La ventaja que introduce NestJS, frente a usar Express.js por nuestra cuenta, es que establece una arquitectura de la aplicación para facilitar el desarrollo de aplicaciones más mantenibles y con código más limpio [44].

PostgreSQL. Es una base de datos relacional de código abierto, que utiliza como lenguaje de consultas SQL. Permite la extensión mediante *plugins*.

TimescaleDB. Plugin para PostgreSQL. Permite mejorar el rendimiento de inserción en tablas con un gran volumen de datos, ya que las va dividiendo automáticamente en subtablas de tamaño configurable para evitar que crezcan indefinidamente, lo que ralentizaría la velocidad de inserción de nuevos datos. Todo este proceso se realiza de forma transparente para el usuario, las consultas se realizan sobre una única tabla, y en caso de necesitar filas de diferentes subtablas, la consulta se realiza como si la tabla fuese una tabla normal, pero Timescale se encarga de devolver todas las filas que corresponda. Se utiliza este plugin para la tabla de almacenamiento de los datos del sensor, con expectativa de recibir unas 50.000 lecturas por dispositivo al año, si tuviéramos 1000 dispositivos serían 50 millones de filas por año, por lo que a largo plazo se apreciaría una mejora de rendimiento importante. Se muestran las mejoras en el rendimiento por el uso de TimescaleDB en este documento de la bibliografía: [45].

Prisma. ORM[23] de código abierto para realizar consultas a la base de datos desde la aplicación Node.js. Es el ORM con mejor documentación, y con una forma de definir los modelos más limpia para Node.js. Además ofrece un gran soporte de tipos, y permite realizar consultas SQL crudas, pero protegidas contra inyección SQL.

EMQ X. Broker MQTT de código abierto de alto rendimiento.

ReactJS. Librería de JavaScript de código abierto, creada por Facebook, para construir interfaces de usuario avanzadas.

21 «*Framework* para crear aplicaciones de escritorio usando JavaScript HTML y CSS. Basado en Node.js» [43]

22 Superconjunto de JavaScript que añade tipado y otras mejoras.

23 Es una técnica de mapeo que consiste en mapear los datos de una base de datos relacional al sistema de tipos utilizado en el lenguaje de programación orientado a objetos. Llamamos ORM también al software que genera la abstracción de la base de datos siguiendo esta técnica. [46]

3.4.2. Entorno de desarrollo

Para el desarrollo de la aplicación necesitamos tener una instancia de la base de datos PostgreSQL[24], y una instancia del broker EMQ X. Para ello configuramos dos contenedores Docker mediante Docker Compose[25]. Esto es más cómodo que instalar el servidor de base de datos y el broker en nuestra máquina, y hace más fácil compartir el entorno de desarrollo con otros desarrolladores, si fuese necesario. Además permite que podamos llevar a cabo proyectos con diferentes versiones de base de datos por ejemplo, sin ningún problema, ya que podemos tener una instancia virtualizada de la base de datos para cada proyecto en el que estamos trabajando en ese momento.

3.4.3. Servidor Node.js

Se han seguido las directrices de arquitectura del *framework* NestJS. Se divide el código en módulos por cada funcionalidad, y dentro de cada módulo tenemos todo lo relativo a esa funcionalidad. Los módulos desarrollados más importantes son los siguientes:

- **Módulo de autenticación[26] y autorización[27] ("auth")**: Para la autenticación utilizamos la librería Passport.js[28]. Dentro de este módulo definimos las estrategias de validación de el intento de *login* mediante email y contraseña, y la validación de los JWT[29] que los usuarios logueados enviarán en cada petición. Un servicio[30] con métodos para validar las credenciales de un usuario, para generar un *token* JWT para un usuario (este es el que el usuario enviará en cada petición para mostrar que está autenticado) y para registrar un nuevo usuario. Y un controlador con: una ruta

24 En el anexo "Estructura de datos. Modelo entidad-relación" se presenta la estructura de datos utilizada.

25 Compose es una herramienta complementaria a Docker, que sirve para definir y ejecutar aplicaciones Docker multi-contenedor. Permite configurar varios contenedores en un archivo y arrancarlos y pararlos al mismo tiempo de forma sencilla. [47]

26 Autenticación es el acto de confirmar que alguien es quien dice ser. [48]

27 Autorización es el proceso por el cual un sistema decide si un usuario tiene permiso para acceder a un recurso. [49]

28 Librería de autenticación para ExpressJS. Permite diferentes tipos de autenticación, como usuario-contraseña, o validación de *JSON Web Tokens* (JWT), mediante un concepto que denominan "*strategies*". Cada "*strategy*" es un método de autenticación definido mediante una clase.

29 Los *JSON Web Tokens* son un método estandarizado y abierto para representar porciones de información de forma segura entre dos partes, basados en JSON. Normalmente un servidor, tras validar las credenciales de un usuario, genera un JWT, por ejemplo con el id del usuario, y los permisos del mismo como contenido. Lo firma con una clave secreta, y se lo envía al usuario. El usuario, envía este *token* en cada petición al servidor, y este lee el contenido y comprueba que la firma es correcta para comprobar qué usuario realiza la petición, y que realmente es quién dice ser. Si un usuario modificase el contenido para suplantar a otro, no podría generar una firma válida, ya que no conoce el secreto, por lo que el servidor detectaría que la integridad del mensaje se ha violado y rechazaría la petición. [50]

30 Los servicios son clases que se encargan de obtener los datos de la base de datos, y juntarlos de forma lógica para realizar una tarea específica, mientras que los controladores se encargan de orquestar las llamadas a los servicios, configuran las rutas de acceso HTTP a los recursos, y controlan la entrada del usuario y la respuesta que finalmente se le entrega a éste. Es una separación de ámbitos que permite escribir un código más limpio, al separar la parte relativa al servidor HTTP (controlador) y la parte relativa a la lógica de la aplicación (servicio).

para hacer *login* (validar el email y contraseña del usuario y en caso de ser correctos le envía una *cookie* HttpOnly[31] con el JWT), una ruta para registrar un usuario, en la que al enviar un objeto con el email, contraseña deseada, nombre y apellidos, se registra un usuario en el sistema, y otra para hacer *logout* que envía una *cookie* HttpOnly con contenido vacío y expiración instantánea para sobrescribir la *cookie* de autenticación en el navegador del cliente, y expirar en ese mismo instante.

- **Módulo de relaciones usuario-cliente ("memberships")**: la entidad *membership* relaciona un usuario con un cliente y le asigna un permiso sobre el cliente. Un cliente podría ser una empresa, o un autónomo, que tiene asociadas fincas, y "agromotes"[32], entre otras cosas. Cada *membership* define la relación que existe entre un cliente y un usuario, que puede ser "*ownership*", cuando un cliente ha sido creado por un usuario, "*access*" o "*management*", cuando el *owner* ha otorgado permisos de acceso o gestión respectivamente, a un usuario, que podría ser un empleado que trabaja para él. Éste es el módulo desde el que se puede ver a qué clientes tiene acceso un usuario.

- **Módulo de clientes ("clients")**: que nos permite crear un cliente, asignándole la "membresía" "*owner*" al usuario que lo crea (mediante la lectura del id de usuario de su *token* de autenticación al realizar la petición), y obtener la información de un cliente.

- **Módulo de fincas ("farms")**: que tiene rutas para la creación de una finca, asignándole parcelas mediante su número de parcela de SIGPAC[33] para su visualización en un mapa, así como ponerle un nombre para su identificación, o asociarlo a un "agromote".

- **Módulo de geometrías de SIGPAC ("sigpac-geometries")**: que consume una API de SIGPAC a la que se puede consultar un número de parcela para saber los números de "recinto" que tiene, y posteriormente con otra consulta a la API se puede descargar el polígono que define geográficamente a cada "recinto". De este modo podemos almacenar en nuestra base de datos lo que hemos llamado "farmplots", que contienen un número de parcela en formato *String*, y un "Multipolygon"

31 Una *cookie* HttpOnly es una *cookie* marcada como tal por el servidor. Estas *cookies* son inaccesibles desde el código JavaScript en el navegador, evitando así ataques *cross-site-scripting* (XSS).

32 "Agromote" es el nombre que le hemos dado dentro del dominio de la aplicación, al dispositivo electrónico con sensores.

33 Sistema de información geográfica de parcelas agrícolas de España, que permite identificar geográficamente las parcelas declaradas por los agricultores y ganaderos.

geográfico en formato GeoJSON[34] que define la forma de los recintos de una parcela, tal y como se pueden ver en SIGPAC. Posteriormente se pueden asignar tantos “farm-plots” como sea necesario a una finca, para poder visualizar todas las parcelas que conforman esa finca en un mapa.

- **Módulo “mqtt”**, que contiene un servicio que se conecta al broker MQTT como cliente y se suscribe al *topic* en el que los sensores envían los datos. Cada vez que llega un dato a un *topic*, este servicio llama a los servicios “agromote-status”, y “clima-sensor-data”, para el almacenamiento en base de datos del estado de conexión del dispositivo y de los datos recibidos por los sensores respectivamente.

Hay otros módulos, como uno para registrar “agromotes” en el sistema u obtener la información de uno. Otro para la gestión de los puntos geográficos de instalación de los agromotes (“installation-points”[35]), entre otros. Pero todos siguen una estructura similar con un “servicio” que es el que hace uso del ORM Prisma para realizar consultas a la base de datos, y un “controlador” que define la ruta de acceso en la API REST, los parámetros aceptados, y orquesta las llamadas necesarias a los servicios para dar una respuesta al usuario. En la Figura 11 se puede ver el detalle del IDE utilizado para el desarrollo del servidor Node.js y de la interfaz de usuario.

34 «GeoJSON es un formato estándar abierto diseñado para representar elementos geográficos sencillos, junto con sus atributos no espaciales, basado en JSON. Es un formato ampliamente utilizado en aplicaciones de cartografía en entornos web al permitir el intercambio de datos de manera rápida ligera y sencilla.» [51]

35 Los puntos geográficos de instalación son una entidad que define un punto geográfico mediante sus coordenadas, y un nombre familiar para el cliente para referirse a él. A este punto se asociará uno o varios “agromotes” y sirve para tener relacionadas las lecturas de los sensores con la posición geográfica en la que se han tomado.

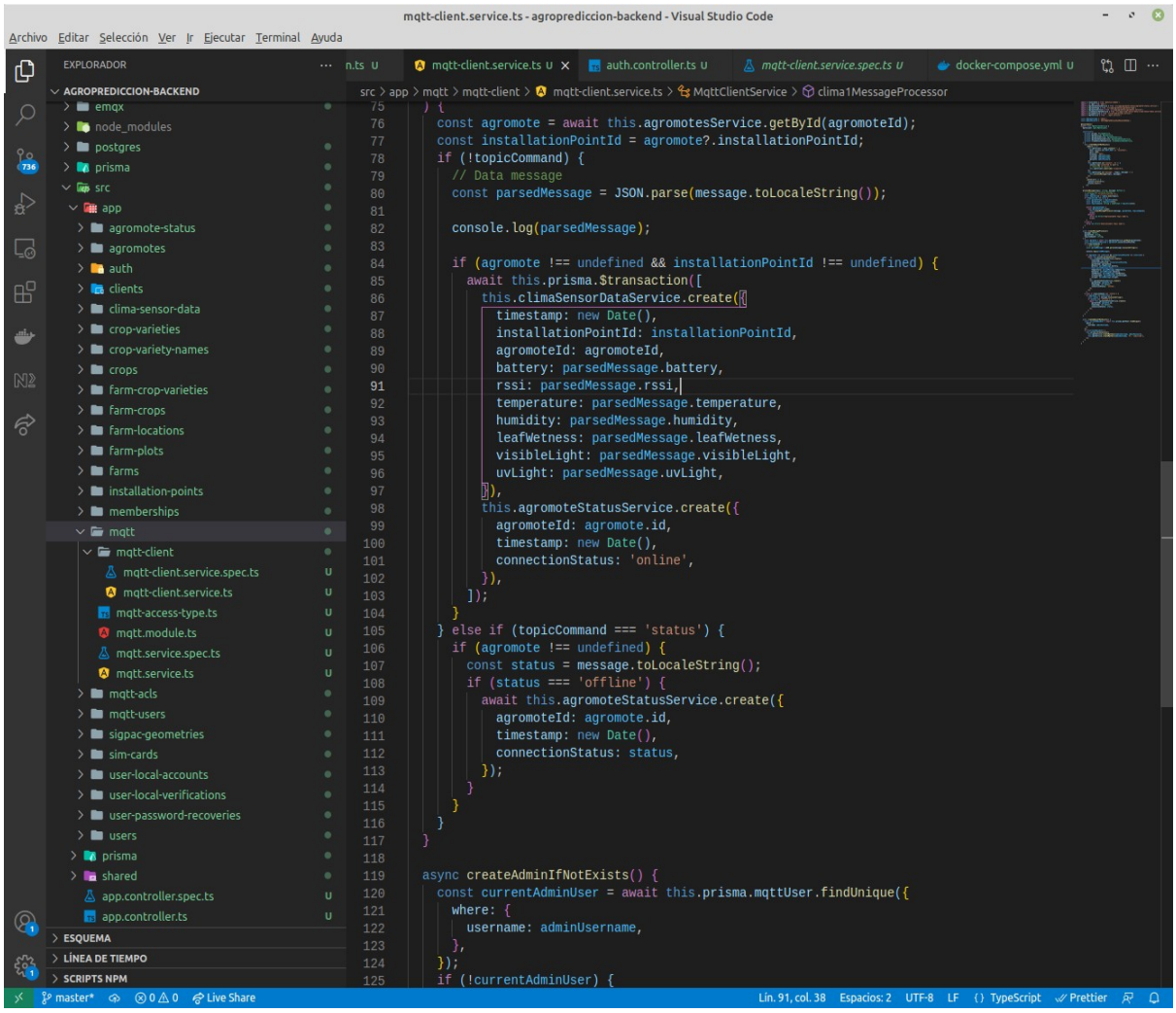

Figura 11: Detalle del IDE Visual Studio Code

3.5. Interfaz de usuario

La interfaz de usuario consiste en una aplicación web hecha con ReactJS y TypeScript. La estructura del proyecto separa la parte de lógica, donde tenemos interfaces[36] de los objetos que enviamos y recibimos del servidor, para tener autocompletado de tipos en el IDE, y servicios para realizar las peticiones HTTP necesarias a la API REST del servidor Node.js, mediante la librería Axios[37]. Por otro lado tenemos la parte donde describimos las pantallas de la aplicación, anidando las carpetas igual que las rutas de la aplicación. Dentro de cada subcarpeta se encuentran los componentes utilizados por esa pantalla, y los

36 En programación orientada a objetos una interfaz es una estructura que permite forzar ciertas propiedades en una clase. Es un "contrato", que define las propiedades y métodos que debe tener una clase que quiera implementarlo. [52]

37 Es una librería JavaScript que consiste en un cliente HTTP. Añade algunas funcionalidades útiles como interceptar peticiones y respuestas, o transformar las respuestas en JSON automáticamente, frente al uso de la API nativa para realizar peticiones HTTP desde JavaScript.

que son utilizados por varias, se encuentran en una carpeta de nivel superior. Por otro lado tenemos el *layout* (componente de maquetación) que comparten todas las pantallas de la aplicación excepto el *login*, esto es el menú superior, y el menú lateral, que es responsivo[38] para poder visualizarse en dispositivos móviles.

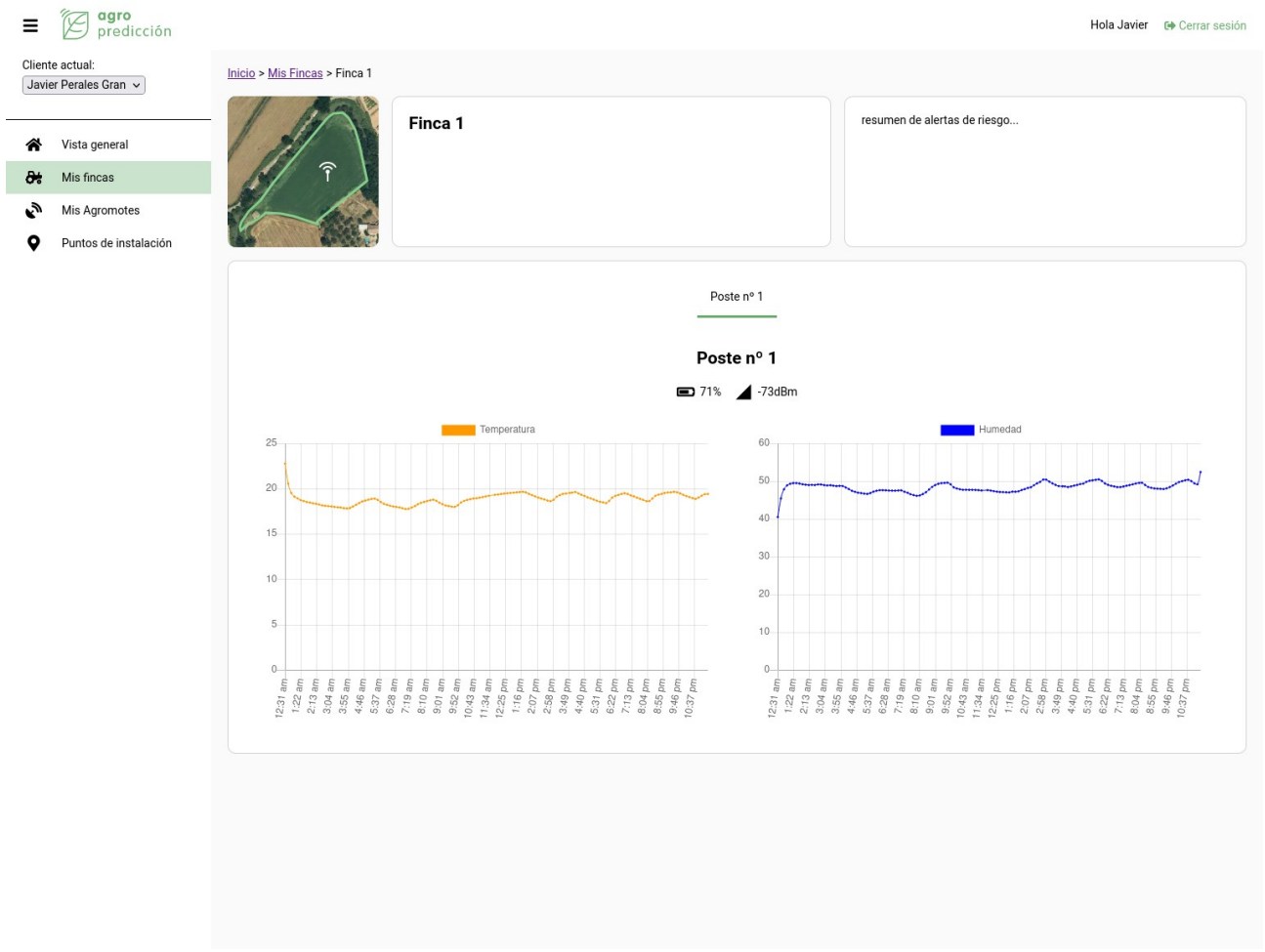

Figura 12: Detalle de la vista de una finca en la interfaz gráfica

Una vez vista brevemente la arquitectura, pasamos a describir el funcionamiento de la aplicación. Al entrar a la página principal, o a cualquier otra ruta, la aplicación comprueba en localStorage[39] si hay una sesión iniciada. En caso de haberla se va a la ruta indicada, en caso contrario se redirige directamente a la pantalla de *login* para iniciar sesión. En la pantalla de *login* se nos pide el email y la contraseña, de una cuenta que habremos crea-

38 El diseño responsivo es una filosofía de diseño y desarrollo web que busca adaptar la apariencia de las páginas web al dispositivo desde el que se visitan, ya sea tablet, móvil o un ordenador.

39 *Local storage* es una API de los navegadores web, que permite a las páginas web almacenar datos sobre una web, que permanecen almacenados entre diferentes sesiones de navegación. Estos datos solo pueden ser accedidos por código JavaScript ejecutado en la misma ruta. Por ejemplo, es útil para almacenar datos de sesiones persistentes. [53]

do mediante Postman (el registro de usuarios no está implementado en la interfaz de usuario). Al introducir los datos, se realiza una petición a la API REST y en caso de que sean correctos la aplicación almacena los datos del usuario en localStorage, así como la fecha de expiración del JWT recibido, para saber cuándo debe marcar en localStorage que no hay sesión iniciada porque ha caducado, y redirigir el navegador a la ruta raíz (“/”).

En la pantalla principal no se nos muestra nada normalmente, pero en caso de que no tengamos ningún cliente asociado, se nos muestra un mensaje que nos dice que no tenemos ningún cliente asociado, y un enlace a un formulario para crear uno. Como se puede ver en el menú lateral de la Figura 12, una vez que tenemos un cliente creado, aparece un selector con los clientes asociados y se nos selecciona automáticamente. Si tenemos más de un cliente podemos cambiar el selector y se actualizaría la pantalla en la que estemos en ese momento con los elementos relativos al cliente seleccionado. Además, la opción del selector de cliente se almacena en localStorage también, para que si salimos del navegador y volvemos a entrar sigamos con el cliente que dejamos marcado la última vez.

En el apartado puntos de instalación podemos crear uno mediante un formulario, introduciendo un nombre, y la latitud y longitud GPS donde se encuentra. En la visión general de los puntos de instalación vemos una cuadrícula con el nombre, las coordenadas y un pequeño mapa con un dibujo de un poste en el lugar de las coordenadas. En la visión de los “agromotes” vemos los que tenemos asociados, y una vista parecida a la de los puntos de instalación, solo que si no hay ningún punto de instalación asociado a un “agromote”, se muestra “sin asignar”.

Figura 13: Vista de una finca en la pantalla del listado de fincas de un cliente

En el apartado de fincas vemos un listado de las fincas que tenemos con su nombre y un pequeño mapa con la forma de las parcelas que tiene asociadas. También se muestra un botón para acceder a un formulario para añadir una nueva finca a ese cliente, en el cual

hay que introducir el nombre para la finca, y los números de parcela que le queremos asociar. En la Figura 13 se muestra el detalle de un elemento de la lista de la página de fincas.

Si hacemos clic en una finca del listado accedemos a la página de vista de una finca como la de la Figura 12, donde se muestra un mapa con el perímetro de las parcelas asociadas a esa finca, y los puntos geográficos de las localizaciones de "agromotes" que haya asociadas a esa finca también en el mapa. Debajo hay una vista con pestañas, con una pestaña por cada punto de instalación asociado, y en el contenido de la pestaña se muestra el nombre, los últimos valores recibidos de nivel de batería y RSSI, así como unos gráficos realizados con la librería Chart.js[40] del histórico de temperatura y humedad recibidos del "agromote" asociado.

Ese es el funcionamiento de la aplicación. Faltan algunas funcionalidades como poder asociar los "agromotes" a un punto de instalación, o un punto de instalación a una finca. Tareas que hay que realizar mediante peticiones con Postman (en el servidor Node.js sí está desarrollado), pero creemos que es una interfaz que cumple con los requisitos que habíamos marcado en la introducción.

3.6. Despliegue en la nube

Para desplegar el sistema se ha utilizado el servidor más barato del proveedor DigitalOcean[41]. Primero nos conectamos al servidor por SSH[42] y configuramos un usuario diferente del root siguiendo esta guía: [55]. Configuramos un firewall (ufw[43]) siguiendo esta guía: [57] permitiendo conexiones SSH, HTTP y MQTT (puertos 22, 80 y 1883) y lo activamos. Instalamos Docker siguiendo esta guía: [58]. Además configuramos el DNS[44] del dominio que vamos a utilizar (agroprediccion.es) para que resuelva el nombre en la IP de nuestro servidor.

Para el despliegue creamos un archivo docker-compose específico para producción en el que creamos un contenedor para ejecutar el servidor Node.js y otro con el proxy inverso

40 Librería JavaScript de código abierto para representación gráfica de datos. https://www.chartjs.org/
41 https://www.digitalocean.com/
42 «SSH es un protocolo cuya principal función es el acceso remoto a un servidor por medio de un canal seguro en el que toda la información está cifrada» [54]
43 «Es un cortafuegos diseñado para ser de fácil uso desarrollado por Ubuntu. Utiliza la línea de comandos para configurar las reglas de paso usando un pequeño número de comandos simples.» [56]
44 «El DNS es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP. Su función más importante es traducir nombres inteligibles para las personas en identificadores binarios asociados con equipos conectados a la red.» [59]

Nginx, que se encargará de servir los archivos de la aplicación ReactJS, y de redirigir las peticiones realizadas a la ruta “/api” a la API REST del servidor Node.js. Además de los contenedores de PostgreSQL y el broker EMQ X. Por otro lado, ponemos a punto los archivos de configuración que se cargarán en cada contenedor para configurar todos los servicios debidamente. Generamos el HTML, CSS y JavaScript comprimido, listo para su uso en producción de la aplicación ReactJS, y en el proyecto del servidor Node.js transpilamos[45] el código TypeScript a JavaScript, que será lo que ejecutemos en producción.

Una vez que tenemos todo lo anterior, nos conectamos mediante el software Filezilla[46] al servidor de DigitalOcean por SFTP[47] para cargar los archivos de configuración de los contenedores Docker, el archivo docker-compose, y las carpetas con los archivos de producción de la aplicación ReactJS y del servidor Node.js, y arrancamos todos los contenedores comprobando que todo funciona correctamente. Ya solo nos queda modificar el código del dispositivo electrónico para conectarse con nuestro broker MQTT en lugar del de test, cargamos el nuevo código al dispositivo con el usuario y la contraseña que hemos configurado en el servidor y la nueva URL del servidor, y tras configurar en la aplicación un nuevo usuario, un cliente, añadir una finca y asociar el “agromote”, comprobamos que en la interfaz web se muestran las nuevas lecturas de los sensores con el paso de los minutos, por lo que tenemos la arquitectura descrita en el capítulo 3.1, funcionando en la nube con éxito.

45 Transpilar es un caso especial de compilación por el cual se genera código en un lenguaje a partir de código en otro lenguaje, teniendo ambos lenguajes el mismo nivel de abstracción.

46 Es un cliente FTP y SFTP. Sirve para el intercambio de archivos con un servidor FTP o SSH.

47 «SFTP es un protocolo que permite realizar operaciones sobre archivos remotos. Similar a FTP, basado en SSH.» [60]

4. Conclusiones

Durante el desarrollo del proyecto, se han llevado a cabo tareas de documentación sobre protocolos, tecnologías, diseño electrónico, y programación, entre otros campos. En este capítulo se presentan los aprendizajes obtenidos, el impacto del proyecto, y posibles mejoras o desarrollos futuros que se apoyan en el presente trabajo.

4.1. Aprendizajes y errores

Los principales aprendizajes han sido sobre diseño de electrónica, he aprendido a comparar y elegir componentes electrónicos, he profundizado en el diseño de circuitos, en temas como el uso de los condensadores de desacoplo, en protocolos de comunicación como UART, o I2C. He profundizado en el diseño de placas de circuito, pasando del esquemático a dibujar las pistas, teniendo en cuenta el grosor de las mismas según la corriente que debe atravesarlas, o para que la impedancia de la pista al conector de la antena esté adaptada, y teniendo en cuenta evitar los ángulos rectos para que las pistas no actúen como antenas indeseadas, captando o emitiendo ruido electromagnético. He cometido innumerables fallos en los diseños electrónicos: desde olvidar enrutar[1] una pista, hasta colocar un conector con las conexiones invertidas, pasando por elegir un chip integrado equivocadamente, no consiguiendo el funcionamiento deseado, lo cual me ha traído numerosos aprendizajes que no podría enumerar en estas líneas. Por otro lado, también he aprendido sobre el protocolo MQTT, el broker EMQ X, y he podido poner el práctica el uso de algunas tecnologías de desarrollo web que ya conocía por mi trabajo en el sector.

4.2. Trabajos futuros

El primero de los trabajos a realizar sería resolver un par de errores del diseño del dispositivo electrónico: el conector invertido, y el circuito de carga de la batería, en el cual se eligió un integrado inadecuado. Por otro lado, habría que terminar el desarrollo de la interfaz de usuario, que por cuestiones de tiempo, y porque escapa el alcance del proyecto ha quedado con una funcionalidad un poco incompleta, pero debería mejorarse mucho para poder llevar el producto al mercado.

1 Trazar el camino de una conexión eléctrica al diseñar una placa de circuito impreso.

Otro posible desarrollo sería una aplicación móvil en la que poder recibir las alertas de riesgo mediante notificaciones *push*[2]. Esto sería esencial también para llevar el producto al mercado, ya que es la forma más inmediata de poder avisar a los agricultores de que su cultivo está en riesgo.

4.3. Reflexión final

Creo que el proyecto ha servido como prueba de concepto para demostrar que es posible conseguir un sistema de bajo coste para monitorizar los cultivos agrícolas. En el aspecto personal estoy muy contento con los aprendizajes obtenidos, han sido muy enriquecedores para mi formación y confío en que útiles para mi futuro profesional. Por último, comentar que el proyecto sigue en desarrollo, que se construye sobre el trabajo aquí presentado, y espero que pronto tengamos un producto maduro que sirva para ayudar a los agricultores a cuidar la salud de sus cultivos.

2 «La tecnología *push* es una forma de comunicación a través de internet en la que la petición de envío tiene origen en el servidor, al contrario de la tecnología *pull*, que tiene origen en el cliente.» [61]

Bibliografía

[1] «Interfaz de programación de aplicaciones». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

[2] «Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)». https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (accedido oct. 27, 2021).

[3] David Flanagan, *JavaScript – The Definitive Guide*, 5th de. O'Reilly, 2006.

[4] «Servicio general de paquetes vía radio». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Servicio_general_de_paquetes_v%C3%ADa_radio

[5] «Enhanced Data Rates for GSM Evolution». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Enhanced_Data_Rates_for_GSM_Evolution

[6] «Telefonía móvil 3G». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Telefon%C3%Ada_m%C3%B3vil_3G

[7] «LTE (telecomunicaciones)». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/LTE_(telecomunicaciones)

[8] «The IoT device market continues to expand, especially in China». https://on5g.es/en/the-iot-device-market-continues-to-expand-especially-in-china/ (accedido nov. 08, 2021).

[9] «Paging LTE». Accedido: nov. 21, 2021. [En línea]. Disponible en: https://www.sharetechnote.com/html/Paging_LTE.html

[10] «Protocolo NB-IoT y su PSM». https://programmerclick.com/article/81241515818/ (accedido nov. 21, 2021).

[11] «Ultra Narrowband». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Ultra_Narrowband

[12] «Ya queda menos para que apaguen el 2G y 3G». https://www.adslzone.net/noticias/operadores/2g-3g-apagado-datos-2021/ (accedido nov. 08, 2021).

[13] «Flag». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Flag

[14] «Introducción a JSON». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.json.org/json-es.html

[15] «RFC 7230». [En línea]. Disponible en: https://datatracker.ietf.org/doc/html/rfc7230

[16] «TCP». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Glossary/TCP

[17] «¿Qué es el protocolo IP?» Accedido: nov. 22, 2021. [En línea]. Disponible en: https://redesteleco.com/que_es_protocolo_ip/

[18] «MQTT Version 5.0». Accedido: nov. 11, 2021. [En línea]. Disponible en: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[19] «HTTP Keepalive Connections and Web Performance». https://www.nginx.com/blog/http-keepalives-and-web-performance/ (accedido nov. 13, 2021).

[20] «EMQ Concurrent connection test». Accedido: nov. 13, 2021. [En línea]. Disponible en: https://emq-xmeter-benchmark-en.readthedocs.io/en/latest/connection.html

[21] «HTTP vs MQTT: A tale of two IoT protocols». https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols (accedido nov. 11, 2021).

[22] «MQTT vs HTTP: ¿qué protocolo es mejor para IoT?» https://borrowbits.com/2020/04/mqtt-vs-http-que-protocolo-es-mejor-para-iot/ (accedido nov. 11, 2021).

[23] «Reasons and Peculiarities of Choosing MQTT Protocol for Your IoT Devices». https://www.integrasources.com/blog/mqtt-protocol-iot-devices/ (accedido nov. 11, 2021).

[24] «Proxy inverso». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Proxy_inverso

[25] «Entorno de desarrollo integrado». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

[26] «Arquitectura ARM». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Arquitectura_ARM

[27] «Docker (software)». Accedido: nov. 15, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Docker_(software)

[28] «MCU de 8 bits vs. De 32 bits: cómo elegir el microcontrolador adecuado para el diseño de tu PCB». https://resources.altium.com/es/p/8-bit-vs-32-bit-mcu-choosing-right-microcontroller-your-pcb-design (accedido nov. 22, 2021).

[29] «Description of STM32L4/L4+ HAL and low-layer drivers». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.st.com/resource/en/user_manual/dm00173145-description-of-stm32l4l4-hal-and-lowlayer-drivers-stmicroelectronics.pdf

[30] «STM32L4 – LPUART». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.st.com/content/ccc/resource/training/technical/product_training/3e/e7/ff/6f/b4/11/43/a4/STM32L4_Peripheral_LPUART.pdf/files/STM32L4_Peripheral_LPUART.pdf/jcr:content/translations/en.STM32L4_Peripheral_LPUART.pdf

[31] «What is an 18650 Battery?» Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.18650batterystore.com/collections/18650-batteries

[32] «Regulador de baja caída (LDO)». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Regulador_de_baja_ca%C3%Adda_(LDO)

[33] «I2C». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/I%C2%B2C

[34] «Tecnología de montaje superficial». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Tecnolog%C3%ADa_de_montaje_superficial

[35] «Solder Paste Stenciling». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.sparkfun.com/tutorials/58

[36] «Tecnología de agujeros pasantes». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Tecnolog%C3%ADa_de_agujeros_pasantes

[37] «Indicador de fuerza de la señal recibida». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Indicador_de_fuerza_de_la_se%C3%B1al_recibida

[38] «Datasheet SHT3x». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.mouser.com/datasheet/2/682/Sensirion_Humidity_Sensors_SHT3x_Datasheet_digital-971521.pdf

[39] «Interrupción». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Interrupci%C3%B3n

[40] «El heap y la memoria dinámica». Accedido: nov. 22, 2021. [En línea]. Disponible en: http://www.it.uc3m.es/pbasanta/asng/course_notes/ch06s03.html

[41] «Node.js». Accedido: nov. 15, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Node.js

[42] «PHP». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/PHP

[43] «Documentación de ElectronJS». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.electronjs.org/es/docs/latest

[44] «NestJS Docs». Accedido: nov. 15, 2021. [En línea]. Disponible en: https://docs.nestjs.com/

[45] «Why Use TimescaleDB over relational databases?» Accedido: nov. 15, 2021. [En línea]. Disponible en: https://docs.timescale.com/timescaledb/latest/overview/how-does-it-compare/timescaledb-vs-postgres/#much-higher-ingest-rates

[46] «Asignación objeto-relacional». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Asignaci%C3%B3n_objeto-relacional

[47] «Overview of Docker Compose». Accedido: nov. 15, 2021. [En línea]. Disponible en: https://docs.docker.com/compose/

[48] «Autenticación». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Autenticaci%C3%B3n

[49] «Autorización». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Autorizaci%C3%B3n

[50] «Introduction to JSON Web Tokens». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://jwt.io/introduction

[51] «GeoJSON». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/GeoJSON

[52] «Interfaces in Object Oriented Programming Languages». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.cs.utah.edu/~germain/PPS/Topics/interfaces.html

[53] «Window localStorage». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/API/Window/localStorage

[54] «Secure Shell». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Secure_Shell

[55] «Configuración inicial del servidor con Ubuntu 20.04». https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04-es (accedido nov. 21, 2021).

[56] «Uncomplicated Firewall». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Uncomplicated_Firewall

[57] «UFW Essentials: Common Firewall Rules and Commands». https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands (accedido nov. 21, 2021).

[58] «How To Install and Use Docker on Ubuntu 20.04». https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04 (accedido nov. 21, 2021).

[59] «Sistema de nombres de dominio». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Sistema_de_nombres_de_dominio

[60] «SSH File Transfer Protocol». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/SSH_File_Transfer_Protocol

[61] «Tecnología push». Accedido: nov. 23, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Tecnolog%C3%ADa_push

[62] «Learn, Why your circuit stops working after sometime | Dry Solder Joints». https://www.raviyp.com/learn-why-your-circuit-stops-working-after-sometime-dry-solder-joints/ (accedido nov. 12, 2021).

[63] «Reverse Voltage Protection for Battery Chargers». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.analog.com/media/en/technical-documentation/application-notes/AN-171_AN-1568.pdf

[64] «Si2333DS Datasheet». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.vishay.com/docs/72023/72023.pdf

[65] «DMP510DL Datasheet». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.diodes.com/assets/Datasheets/DMP510DL.pdf

[66] «Up to 5 W solar battery charger for single-cell Li-ion and Li-Pol batteries based on the SPV1040 and L6924D». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.st.com/resource/en/data_brief/steval-isv012v1.pdf

[67] «STM32L431xx Datasheet». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://eu.mouser.com/datasheet/2/389/dm00257211-1798949.pdf

[68] «SIM7000 Hardware Design». Accedido: nov. 22, 2021. [En línea]. Disponible en: https://www.waveshare.com/w/upload/6/61/SIM7000_Hardware_Design_V1.04.pdf

[69] Luigi Degiorgis, «Design of a low-power, low-cost leaf wetness sensor for smart agriculture application», Politecnico di Torino. [En línea]. Disponible en: https://webthesis.-biblio.polito.it/10992/1/tesi.pdf

[70] «Condensador variable». http://joseal-medicionesindustriales.blogspot.com/2008/06/211-condensador-variable.html (accedido nov. 16, 2021).

[71] «Condensador eléctrico». Accedido: nov. 16, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Condensador_el%C3%A9ctrico

[72] «Constante dieléctrica». Accedido: nov. 16, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Constante_diel%C3%A9ctrica

[73] «Hypertext Transfer Protocol (HTTP)». Accedido: nov. 11, 2021. [En línea]. Disponible en: https://www.extrahop.com/resources/protocols/http/

[74] «Generalidades del protocolo HTTP». Accedido: nov. 11, 2021. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/HTTP/Overview

Índice de figuras

Índice de tablas

Anexo I: Prueba de concepto

En este anexo comentamos los trabajos más relevantes realizados desde la concepción de la idea del proyecto, hasta antes de comenzar el prototipo detallado en esta memoria, puesto que han servido como base para aprender a diseñar placas de circuito impreso, y los errores cometidos han servido para enfocar algunas cosas de diferente forma.

I.1. Prototipo con Arduino

Se comenzó trabajando con un Arduino UNO, una *shield* con el módulo GSM SIM900 para dotar de conectividad móvil al Arduino, y el sensor de humedad y temperatura Aosong AM2305. Se utilizó la librería de Arduino pública TinyGSM[1] para manejar el módulo GSM, y la librería de Adafruit para sensores DHTxx[2], para leer datos del sensor, ya que el AM2305 implementa el mismo protocolo de comunicación que el DHT22. Se montó un prototipo sencillo capaz de enviar la lectura del sensor a un servidor mediante una petición POST vía HTTP con el contenido en formato JSON.

I.2. Arduino personalizado

Estudiando formas de alimentar un Arduino mediante batería y usar los modos de bajo consumo del procesador se llegó a la conclusión de que las placas de desarrollo de Arduino, como tal, no están preparadas para eso, ya que incluyen componentes como el regulador de voltaje, que tienen un consumo demasiado elevado en modo inactivo, lo cual agotaría la batería en cuestión de días por si solo. Por tal motivo decidí comenzar a estudiar diseño de placas de circuito impresas, para diseñar y ensamblar un *Arduino personalizado*, que integrase el módulo GSM, un regulador de tensión adecuado para ser alimentado con una batería LiPo o Li-Ion y un integrado para la carga de la propia batería mediante una placa solar.

Tras varias iteraciones fallidas del proceso de diseño, encargo de fabricación de las PCB a China, y soldadura de los componentes, se consiguió un prototipo funcional en el que se podía cargar y ejecutar código desde el IDE de Arduino, así como obtener la misma funcionalidad que había conseguido con el Arduino original. Es decir, se podía conectar a Internet mediante un módulo GSM similar al SIM900, en este caso el SIM800C, leer datos de un sensor de temperatura y humedad, en este caso el HTU21D.

1 https://github.com/vshymanskyy/TinyGSM
2 https://github.com/adafruit/DHT-sensor-library

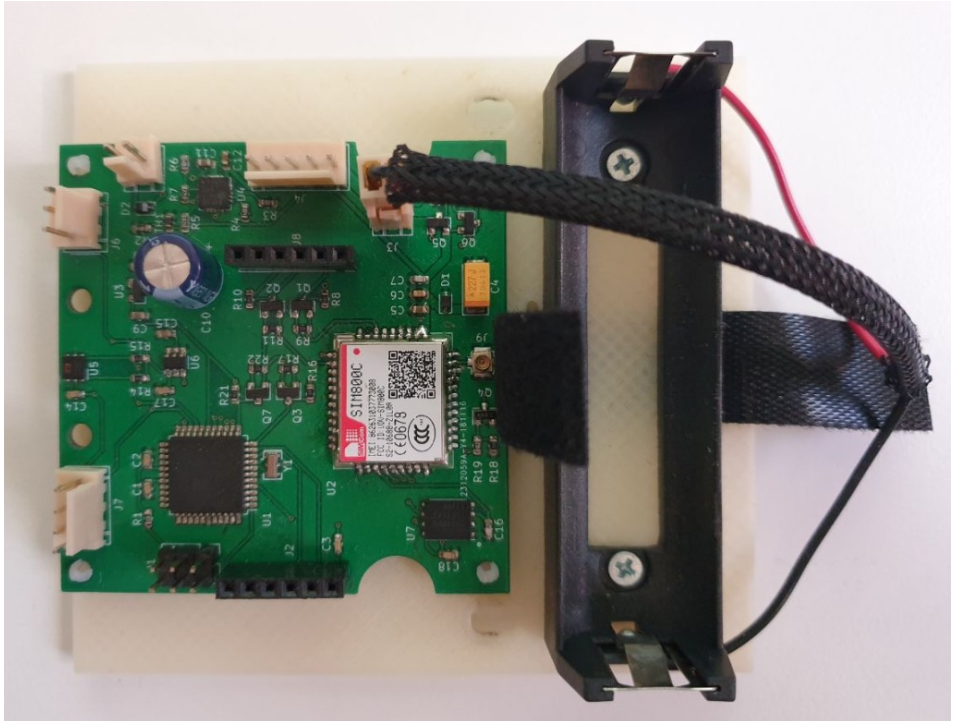

Figura 14: Prototipo personalizado basado en Arduino

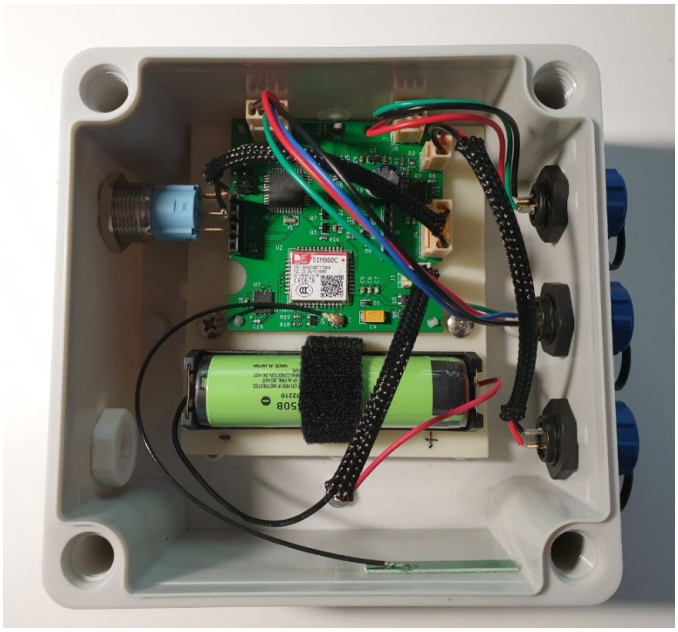

Figura 15: Prototipo electrónico montado en su carcasa para su instalación en el campo

I.3. Pruebas de campo

En la Figura 14 se ve el prototipo mencionado anteriormente, montado sobre una base impresa en 3D para adaptarlo a una carcasa genérica. Este dispositivo se ideó inicialmente para su montaje en otra carcasa, por eso tiene el corte de dos semicircunferencias en sus

bordes que no tiene utilidad aparente. Finalmente, fue más fácil conseguir unas carcasas con otra forma, un poco más grandes, y se diseñó en 3D una base sobre la que anclar la PCB y el soporte de la batería, y que a su vez tuviera agujeros coincidentes con los de la carcasa, para poder anclar todo a ésta. También eran dispositivos pensados para conectar sensores externos mediante conectores resistentes al agua, como se aprecia en la Figura 15.

Una vez montados, se pusieron a funcionar en el campo varios de estos sensores conectados a una placa solar para cargarlos. Como se puede ver en la Figura 16, los dispositivos no envejecieron bien. El dispositivo de la imagen llevaba instalado más de un año, y es cierto que nunca consiguió entrar el agua en su interior, tampoco hay ningún signo de corrosión en el interior, por lo que el montaje y las carcasas cumplieron su función muy bien. Sin embargo, hubo diversos problemas, el primero y evidente a simple vista es que el sensor de humectación de hoja se oxidaba, y sucedía a las pocas semanas. Se explica más en detalle la evolución del proceso de diseño de este sensor en el anexo "Diseño de un sensor de humectación de hoja". Esta oxidación deterioraba consigo la lectura del sensor, por lo que se volvía inservible.

Figura 16: Dispositivo sensor en el campo tras más de 1 año instalado

Por otro lado, algunos de los dispositivos empezaron a dejar de funcionar. Afortunadamente se había dispuesto un botón para encender y apagar los dispositivos fácilmente, por lo que se podían reiniciar, lo cual los devolvía a la vida. Era un problema de fiabilidad del software, que en ciertas situaciones hacía cosas "raras". Sin embargo, al cabo de las semanas y los meses, algunos de los dispositivos no reaccionaban a los reinicios, a pesar de que las últimas lecturas que habían dejado no mostraban que la batería se hubiera agotado. Tras traerlos al laboratorio y ver qué sucedía, algunos de ellos volvían a funcionar en el entorno de laboratorio casi milagrosamente, lo cual puso muy difícil el proceso de depuración para saber qué era lo que estaba pasando. Tras investigar acerca de lo que podía estar sucediendo, descubrí una entrada de un blog muy interesante: "*Learn, Why your circuit stops working after sometime | Dry Solder Joints*"[3], que explicaba exactamente el problema que estaba teniendo, y es que había problemas con las soldaduras. Además de que el equipamiento que se había utilizado para realizar las soldaduras no era el más profesional posible, había otra causa aún más probable: se había usado una pasta de soldadura que llevaba mucho tiempo almacenada, lo que podía haber provocado que hubiera soldaduras frías, que generaban los problemas de fiabilidad en el hardware del dispositivo. Ésto, unido a los problemas de software sufridos, llevó a tratar de desarrollar un dispositivo mejorado en todos los sentidos, reutilizando algunos de los subcircuitos que habían funcionado bien y tratando de mejorar otros aspectos, eliminando fuentes de problemas y actualizando la conectividad, ante un apagón de las redes GSM cada vez más cercano.

3 https://www.raviyp.com/learn-why-your-circuit-stops-working-after-sometime-dry-solder-joints/ [62]

Anexo II: Detalles de diseño del dispositivo electrónico

En este anexo se detallan las decisiones de diseño más importantes, y el funcionamiento de algunos circuitos. No se hace público el diseño completo del dispositivo por tratarse de un proyecto realizado en una empresa, que quiere preservar el secreto industrial.

II.1. Circuitos de alimentación

En cuanto a la alimentación del dispositivo, se ha dispuesto un receptáculo para batería Li-Ion de tipo 18650. Para proteger el circuito y la batería, ante un eventual fallo al insertarla en el receptáculo con la polaridad invertida, se ha utilizado el circuito que se muestra en la Figura 17, que permite el paso de la corriente en ambos sentidos (cuando la batería entrega corriente hacia VBAT, y cuando la batería recibe corriente de VBAT porque está siendo cargada) cuando la batería está con la polaridad correcta. Y corta el paso de la corriente también en ambos sentidos cuando la batería está con la polaridad invertida. Una vez montado el prototipo se ha comprobado en laboratorio que al colocar la batería con la polaridad invertida, el voltaje en VBAT era mayor de -0.3v, por lo que se puede considerar que el circuito funciona como se esperaba. Por supuesto, tras hacer esto ni la batería ni el circuito han sufrido daños aparentes.

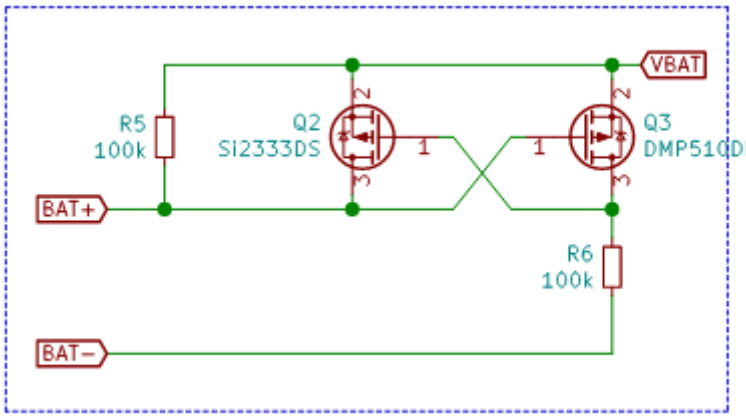

Figura 17: Circuito de protección de polaridad invertida

El circuito está basado en la nota de aplicación 171 de Analog Devices [63]. Se han seleccionado los MOSFET con características más similares que se han podido encontrar. El circuito consiste en dos MOSFET de canal P, el Q3 que detecta la polaridad invertida, y el

Q2, que bloquea el paso de corriente y que posee una R_{DS} muy baja en la región ohmica (42mΩ)[4] para reducir la caída de tensión cuando la batería está polarizada correctamente.

Si la batería está invertida, y suponemos una R_L del circuito mucho menor que R_5, el V_{GS} en Q3 es aproximadamente el voltaje de la batería invertido (-3.6V), y aún menor si el circuito de carga está activo. -3.6V < V_{TH} = -2V, por lo que Q3 conduce, haciendo V_{GS} = 0 en Q2, no permitiendo el paso de la corriente entre BAT+ y VBAT. En el caso de que la batería esté correctamente colocada, V_{GS} = 0 en Q3, por lo que en Q2 V_{GS} = -3.6V < V_{TH} = -1V, así que Q2 está en región óhmica, y deja pasar la corriente entre BAT+ y VBAT. Además de este circuito de protección, se ha añadido un diodo led rojo para advertir de que la batería no está instalada correctamente, que se polariza cuando la batería está con la polaridad invertida y permanece apagado cuando la batería está correctamente instalada.

En cuanto a la alimentación del microcontrolador, los sensores, y otros componentes, por sencillez, se ha optado por un regulador de tensión LDO de 3.3v con una caída de tensión típica de menos de 200mV, por lo que sirve para nuestra aplicación si asumimos que la batería no caiga de su valor nominal de 3.6v. Concretamente se ha optado por el MCP1700 de la empresa Microchip, que además de la baja caída mencionada, tiene una corriente inactiva de tan solo 1.6uA, por lo que el impacto en la vida de la batería cuando el dispositivo está dormido es mínimo.

El subcircuito de la carga de la batería consta de dos partes, un recolector de energía con algoritmo MPPT: SPV1040T de STMicroelectronics, que se encarga de recolectar la energía de una placa solar con tensión de salida entre 0.3V y 5.5V y elevar el voltaje hasta el valor configurado mediante unas resistencias. Con el algoritmo MPPT integrado, este chip es capaz de optimizar la potencia obtenida de la placa solar, haciéndola trabajar en el punto de máxima potencia, ajustando la carga aparente para la placa solar. Se seleccionaron las resistencias que ajustan el voltaje de salida para sacar 4.45V.

Esta salida se conecta con un circuito de carga de batería basado en el integrado BQ24210DQCR de Texas Instruments. Tras realizar las pruebas de este circuito y examinar a fondo el *datasheet*, se llegó a la conclusión de que fue una mala elección, ya que se malentendió una parte de la documentación acerca de la capacidad de ajustar la corriente de carga, pero no era de esa forma. Por lo que el circuito de carga solo funciona de manera efectiva cuando la potencia entregada por la placa solar es mayor de cierto umbral. Si

4 Este valor R_{DS}, y los valores de V_{TH} se han obtenido de los *datasheet* de los MOSFET utilizados [64], [65].

se volviera a diseñar hoy, escogería el integrado L6924D de STMicroelectronics, que es más adecuado para ese propósito, como se muestra en esta nota de aplicación: [66]

II.2. Microcontrolador y comunicaciones

El microcontrolador elegido es el STM32L431RCT de STMicroelectronics. La elección se ha basado en las pruebas realizadas previamente con una placa de desarrollo que tenía un microcontrolador de la misma familia. Se ha elegido éste atendiendo a los periféricos que incluía, memoria de programa y memoria RAM, ya que tras desarrollar el software inicial se observó que el microcontrolador utilizado (que era el más completo de la gama), tenía muchas cosas que no estábamos utilizando, por lo que se buscó uno menos sobre dimensionado para reducir un poco el coste. Todo lo relacionado con el micro, como los condensadores de desacoplo, el cristal para el oscilador, y los conectores necesarios para su uso, se han elegido siguiendo las recomendaciones del fabricante en el *datasheet* [67], por lo que no tienen mucho interés.

Como se ha mencionado en otras partes del trabajo, el módulo utilizado para las comunicaciones móviles es el SIM7000, concretamente la versión europea SIM7000E. El diseño alrededor de este módulo se ha realizado siguiendo las guías del fabricante SimCom [68]. Se ha utilizado un receptáculo para nano-sim genérico, de una marca china, junto con un filtro EMI integrado de la marca Texas Instruments, específico para su uso en las líneas de datos de una SIM. Dado que el microcontrolador de nuestro circuito se alimenta a 3.3v, y los pines digitales del módulo SIM7000 funcionan a 1.8v, hemos utilizado el conversor de nivel TXB0104D con 4 canales de Texas Instruments, para las conexiones de recepción y transmisión UART, y las conexiones digitales para conocer el estado del módulo y para activar el modo sleep.

Anexo III: Diseño de un sensor de humectación de hoja

La medición de la humectación de hoja es muy importante para algunos modelos de predicción de enfermedades fúngicas, ya que la cantidad de horas que las hojas permanecen húmedas en ciertas condiciones de temperatura, pueden marcar la diferencia entre la proliferación masiva de hongos, o que no haya ningún problema.

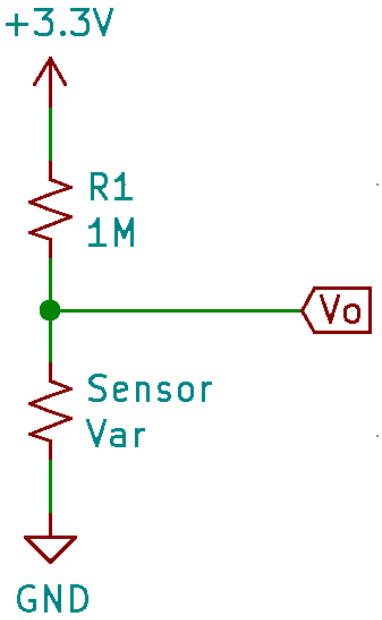

Figura 18: Divisor resistivo para sensor de humectación

Figura 19: Sensor resistivo de humectación de hoja

En un primer momento del desarrollo se probó un sensor de tipo resistivo, que consistía en un patrón interdigital, concretamente el sensor de la Figura 19, fabricado por Hobby-Boards[5] El funcionamiento de este sensor se basa en la variación de la resistencia entre los dos bornes. En seco, esta resistencia es idealmente infinito, pero en presencia de rocío o escarcha, el agua permite cierto paso de la corriente entre las pistas, por lo que la resistencia disminuye.

Tomando como referencia el circuito de la Figura 18, si leemos el voltaje Vo cuando está seco, este será de 3.3v, puesto que no pasa corriente por el sensor y no hay caída de tensión. Si suponemos que el sensor sumergido en agua tiene una resistencia de 100k, la lectura en Vo en esas condiciones sería 0,09*3,3v = 0,3v. De este modo podemos calibrar

5 https://hobbyboards.com/ - En la actualidad, el sitio web se encuentra inactivo, y ya no fabrican ni venden este producto.

los diferentes estados del sensor que entregarán una lectura Vo entre 0,3v y 3,3v. Tras probar el sensor en laboratorio, y validar su funcionamiento, se diseñó uno casi igual, pero con el divisor resistivo incorporado, y un agujero para su montaje más alejado del elemento sensor. Éste es el que finalmente se instaló en el campo en las primeras pruebas.

Sin embargo, este sensor tenía importantes problemas. Por un lado su comportamiento estaba muy lejos de ser lineal, pero lo realmente grave es que las pistas, a pesar de estar chapadas en oro, se oxidaban a los pocos días de estar a la intemperie, y la capa de óxido variaba las lecturas del sensor, lo que lo volvía inservible. Por todo ello, investigando sobre sensores comerciales de este tipo, se vio que son de tipo capacitivo, y no resistivo. También se vio que el coste era de en torno a 100€ la unidad, lo cual se escapaba del presupuesto, por lo que decidimos diseñar uno, basado en el resultado de una tesis de máster del Politecnico di Torino [69].

Un sensor de humedad capacitivo se basa en el principio de funcionamiento de un condensador. Las pistas eléctricas tienen una forma similar a las del sensor resistivo, sin embargo están cubiertas de un material aislante, como por ejemplo la máscara de soldadura. Estas pistas interdigitales actúan como las placas de un condensador. Al ser las placas del condensador de un grosor mínimo, el efecto de borde[6] tiene mucho peso en la capacidad total del condensador, y precisamente nos aprovechamos de este efecto, ya que la superficie que se encuentra encima de lo que sería el condensador ideal, que es la superficie del sensor, es precisamente donde queremos medir la presencia de agua, y gracias al efecto de borde, en presencia de agua, la capacidad del condensador se incrementa notablemente, ya que la constante dieléctrica del agua ($\varepsilon_r = 78,5$) es muy superior a la del aire ($\varepsilon_r = 1,0$). [71], [72]

Una vez que tenemos un condensador variable, podemos utilizarlo en el circuito RC de un oscilador, y leer la frecuencia de salida para detectar las variaciones de la humedad encima del sensor. En la Figura 20 se puede observar el circuito utilizado para la medición de la capacidad del condensador variable, que permite detectar la presencia de agua en la superficie del sensor. La conexión nombrada como C+ es uno de los polos del condensador, y el otro está conectado a GND. Como se puede observar hay un condensador C1 de tipo NP0, cuya capacidad varía lo mínimo posible con la temperatura, en paralelo con el condensador sensor. Este condensador C1 tiene un valor decidido tras haber probado el

6 El efecto de borde de un condensador, es un efecto, normalmente indeseado, que se produce en los bordes de un condensador de placas paralelas, y que consiste en el desvío de las líneas de campo eléctrico en el borde de las placas, haciendo que las líneas no se comporten de forma perpendicular a dichas placas. [70]

sensor sin ningún condensador montado, y su propósito es aumentar la capacidad del circuito RC del oscilador, para que trabaje en la región lineal, puesto que si la capacidad del condensador del circuito RC es demasiado pequeña, el circuito trabaja en una región no lineal y la variación de capacidad no se reflejaría linealmente en la variación de frecuencia.

La capacidad era de apenas unos pF en seco, y según el *datasheet*[7], con una resistencia de 1MOhm, el comportamiento empieza a ser lineal a partir de los 100pF aproximadamente, por lo que necesitábamos añadir un condensador en paralelo para sumar las capacidades y hacer trabajar al oscilador en la región lineal.

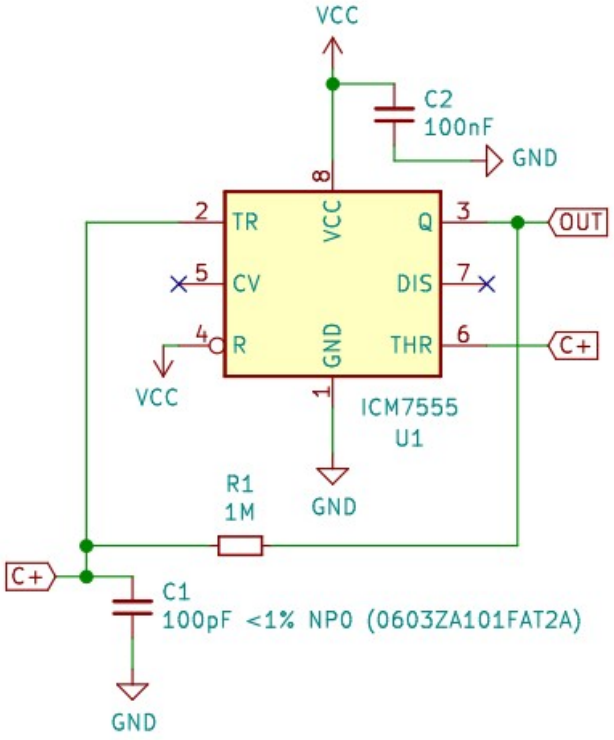

Figura 20: Circuito oscilador ICM7555 para la medición de capacidad de un condensador variable

A la salida del circuito, el punto eléctrico OUT, tenemos una onda cuadrada con frecuencia variable, en función de la cantidad de agua líquida que se encuentra en la superficie del sensor. Al medir la frecuencia con un dispositivo electrónico, podemos calibrar la frecuencia cuando el sensor está completamente seco, y cuando el sensor está completa-

7 https://www.renesas.com/us/en/document/dst/icm7555-icm7556-datasheet

mente cubierto de agua, para poder obtener una medición del porcentaje de humectación de las hojas, tal y como hemos detallado en la sección 3.3.6.

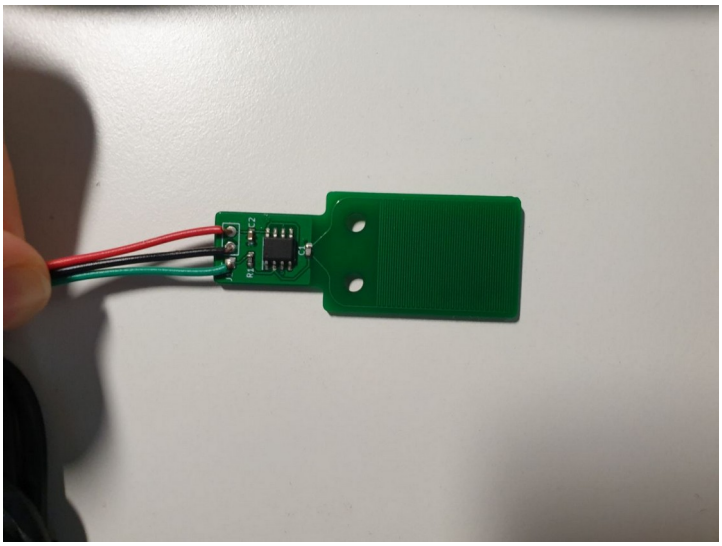

Figura 21: Sensor de humectación de hoja definitivo

En la Figura 21 se puede ver el diseño del sensor definitivo. Cabe señalar que la parte más estrecha, donde se encuentra el chip integrado y los condensadores y la resistencia del circuito RC, van cubiertos por un trozo de tubo termoretráctil con adhesivo, para proteger la electrónica de la entrada de agua y otros agentes, pero se muestra así para que se puedan apreciar los componentes del circuito.

Anexo IV: Valoración de costes

En algunas partes de este trabajo se ha mencionado que buscábamos conseguir un dispositivo de bajo coste. El objetivo inicial era un coste máximo de entre 100-150€ por dispositivo completo con sensores, para poder competir a un precio atractivo con las alternativas existentes. En este anexo detallamos algunos de los costes más importantes del dispositivo y vemos si estamos cerca de nuestro objetivo inicial o no.

Basado en precios reales pagados a proveedores en Alibaba, una carcasa genérica con agujeros personalizados con cortes por CNC[8] costaron unos 11€ por unidad. Un botón resistente al agua para encender y apagar el dispositivo, 3€ por unidad. Los conectores exteriores resistentes al agua cuestan 4€ por pareja (macho y hembra), por lo que serían 12€ por dispositivo, considerando la conexión de 2 sensores externos y una placa solar. El sensor de humectación de hoja diseñado se estima en 10€, incluyendo el coste de fabricación de la placa de circuito, componentes, cable y termoretráctil, y el sensor de temperatura y humedad ya encapsulado para ser resistente al agua se ha comprado por 15€ cada unidad. Placas solares de 1W protegidas con estructura de aluminio y vidrio, con soporte para instalación se han comprado por 12€ cada unidad. Calculando el total del coste de los componentes electrónicos utilizados, y sumando 15€ para el coste de la PCB y el montaje de los componentes, estimamos aproximadamente 60€.

Sumando todo lo anterior tendríamos un total de 123€. A estos costes habría que añadir el coste de montar la placa de circuito en la carcasa, de la instalación de cables interiores para conectar los conectores exteriores con la placa, y del montaje de los conectores externos en los cables de los sensores y la placa solar. Es posible que añadiendo estos costes sobrepasemos ligeramente los 150€ marcados como máximo. Sin embargo, hay que tener en cuenta que las compras indicadas se han realizado en lotes pequeños, de entre 10-30 unidades, por lo que no nos hemos beneficiado de economías de escala, y hemos pagado un sobre coste unitario muy alto por los costes de envío. Por lo que, considero que quizá recortando en algunos costes, como el módulo de conectividad inalámbrica o la placa solar, y aprovechando las economías de escala fabricando lotes de 100 unidades o más, sería perfectamente posible conseguir fabricar un dispositivo con la funcionalidad deseada por menos de 150€.

8 El Control Numérico por Computador se refiere a sistemas que operan máquinas mediante comandos programados. Normalmente son máquinas capaces de realizar tareas con mucha precisión.

Anexo V: Estructura de datos. Modelo entidad-relación

Se ha modelado la estructura de los datos utilizando la sintaxis del ORM Prisma, en un archivo schema.prisma. En la Figura 22 se muestra el diagrama de entidad-relación que hemos obtenido.

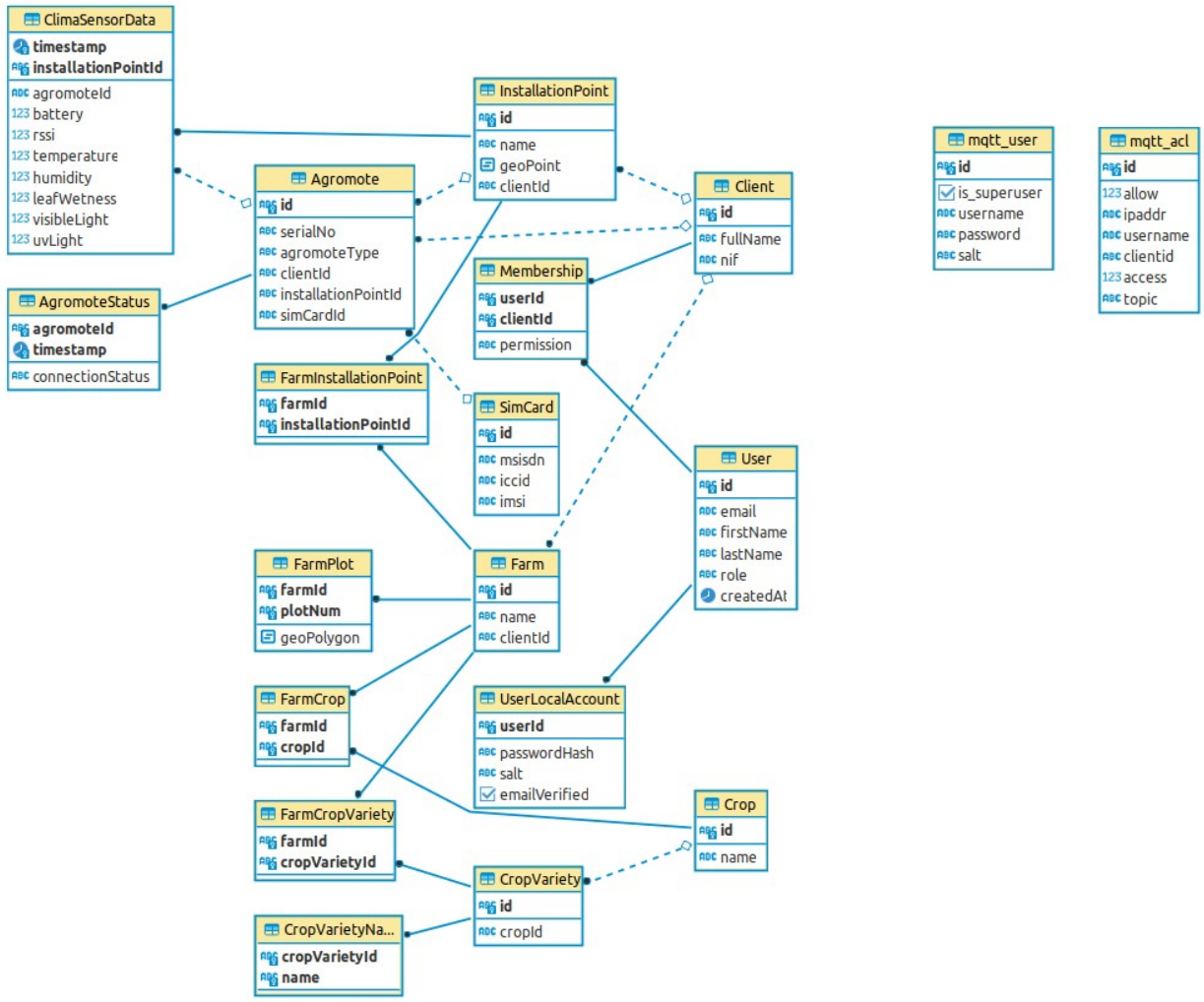

Figura 22: Diagrama entidad relación de la base de datos