Escola Tècnica Superior d'Enginyeria
Electrònica i Informàtica La Salle

Treball Final de Màster

Màster Universitari en Enginyeria de Telecomunicació

Monitorización de sistemas de alimentación ininterrumpida

(Uninterruptible power supply monitoring)

Lluc Boix

Agustin Zaballos Diego

ACTA DE L'EXAMEN
DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D.

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

Resumen del proyecto

Habitualmente, en el mundo de la transformación digital de las ciudades, se requiere de suministros constantes de electricidad en ubicaciones muy dispersas de un territorio. Desgraciadamente, en la mayoría de municipios de nuestro país, no se dispone de buenas infraestructuras para proporcionar este requerimiento. Para hacer frente a esta problemática, una solución bastante habitual es la instalación de sistemas de alimentación ininterrumpida sobrepuestos a la red de alumbrado público.

Entonces, como cualquier otro sistema eléctrico complejo, resulta ser susceptible a fallos. Es aquí donde nace la necesidad de conocer la probabilidad de posibles interrupciones de este suministro, especialmente en casos donde la pérdida de electricidad puede suponer un problema grave, como es el caso de una red de videovigilancia. Esta es la idea fundamental sobre la que se desarrolla este trabajo.

Así pues, este proyecto tiene por objetivo la implementación de un sistema que sea capaz de predecir, detectar e identificar posibles fallos en el suministro eléctrico de una red de videovigilancia. Además, al tratarse de ser un proyecto desarrollada para una empresa instaladora de sistemas de alimentación ininterrumpida, además de procurar el cumplimiento de los requerimientos de la empresa, se buscará ofrecer un valor añadido al servicio, haciendo uso de las posibilidades que ofrece el proyecto.

Resum del projecte

Habitualment, en el món de la transformació digital de les ciutats, es requereixen subministraments constants elèctrics en ubicacions molt disperses d'un territori. Malauradament, en la majoria de municipis del nostre país, no es disposa de bones infraestructures per proporcionar aquest subministrament. Per fer front a aquesta problemàtica, una solució força habitual és la instal·lació de sistemes d'alimentació ininterrompuda sobreposats a la xarxa d'enllumenat públic.

Aleshores, com qualsevol altre sistema elèctric complex, resulta ser susceptible a errors. És aquí on neix la necessitat de conèixer la probabilitat de possibles interrupcions d'aquest subministrament, especialment en casos on la pèrdua d'electricitat pot suposar un problema greu, com és el cas d'una xarxa de videovigilància. Aquesta és la idea fonamental sobre la qual es desenvolupa aquest treball.

Així doncs, aquest projecte té per objectiu la implementació d'un sistema que sigui capaç de predir, detectar i identificar possibles falles en el subministrament elèctric d'una xarxa de videovigilància. Per altre banda, al tractar-se d'un projecte desenvolupat per una empresa instal·ladora de sistemes d'alimentació ininterrompuda, a més de procurar el compliment dels requeriments de l'empresa, es buscarà oferir un valor afegit al servei, fent ús de les possibilitats que ofereix el projecte.

Abstract

Usually, in the world of digital transformation of cities, constant power supplies are required in widely dispersed locations. Unfortunately, a large number of towns in our country do not have a good infrastructure to provide this supply, for this reason, it is a common solution to install uninterruptible power supplies over the public lighting network.

Then, like any other complex electrical system, it becomes susceptible to failures. So, knowing the probability of failure of this supply will be particularly important, especially in cases where a power failure can be a serious problem, such as video surveillance systems. This is the fundamental idea of this work.

Thus, this project aims to implement a system capable of predicting, detecting and identifying possible failures in power supply of a camera network of video surveillance. In addition, since it is a project for an uninterruptible power supplies installer. Apart from respecting the fulfilment of the requirements of the company, it will seek to offer added value to the service making use of the possibilities given by the project.

Acrónimos

- Internet of Things(IoT): Internet de las cosas
- State of Charge(SOC): Estado de carga
- Plug and Play(PnP): Enchufar y usar
- Simple Network Management Protocol(SNMP): Protocolo simple de administración de red
- System on Chip(SoC): Sistema en un chip
- Integrated Development Environment(IDE): Entorno de desarrollo integrado
- Depth of Discharge(DOD): Profundidad de descarga
- Quality Assurance(QA): Aseguramento de la calidad
- Prove of Concept(PoC): Prueba de concepto
- Master en Ingenieria de Telecomunicación(MET)
- JavaScript Object Notation(JSON): Notación de objeto de JavaScript
- General Purpose Input/Output(GPIO): Entrada/salida de proposito general
- End to End(E2E): Punta a punta
- Analogic to Digital(A/D): Analógico a digital
- Global Positioning System(GPS): Sistema de posicionamiento global
- Open Circuit Voltage(OCV): Voltage en circuito abierto
- Light Emitting Diode(LED): Diodo de emisión de luz
- Inter-Integrated Circuit(I2C): Circuito Inter-Integrado
- Pulse Width Modulation(PWM): Modulación por amplitud de pulso
- Structured Query Language(SQL): Lenguaje de consulta Estructurada
- Node Package Manager(NPM): Gestor de paquetes node
- Secure Shell(SSH): Interfaz segura
- Hypertext transfer protocol(HTTP): Protocolo de transferencia de hypertexto
- Model View Controller(MVC): Modelo Vista Controlador
- Uniform Resource Locator(URL): localizador de recursos uniforme
- Universal Serial Bus(USB): bus en serie universal
- Advanced Encryption Standard(AES): Encriptacion estandar avanzada

Agradecimientos

En primer lugar, la primera persona a la que le quisiera agradecer su intervención en este proyecto es el director del trabajo Agustín Zaballos Diego. Su dedicación docente, la exigencia en los detalles y la proximidad de cara al alumno me han ayudando a desarrollar el trabajo muy satisfactoriamente. Además, gracias a él también he aprendido a valorar otros aspectos tan importantes como la organización y la redacción.

En segundo lugar, también quiero dar las gracias a la empresa Sintelec S.L. por darme la oportunidad de desarrollar este proyecto y también por darme plena libertad de decisión sobre el diseño del en todas las fases.

Paralelamente, también quiero agradecer a mi familia que siempre me han apoyado todas las dificultades que se me han ido presentando a lo largo de la carrera.

Por último, quisiera recordar a todos los amigos y compañeros que también me han echado una mano.

Contenido

1 Introducción

1.1 Presentación del problema

La infraestructura urbana de la mayoría de ciudades del mundo carece de una buena disponibilidad de suministro eléctrico constante orientado a sistemas municipales. Si bien es cierto que muchos municipios disponen de este tipo de redes, estos despliegues son muy discretos limitándose a usos muy concretos, como la alimentación de semáforos, quedando su disponibilidad muy reducida. Por otra parte existe otro tipo de redes con mas extensión pero con un suministro más limitado, como la del alumbrado público, donde el suministro queda reducido a pocas horas durante la noche.

Con la tendencia actual de la digitalización de las ciudades, esto supone un problema ya que se requiere un suministro constante de corriente eléctrica para la alimentación de todo tipo de dispositivos electrónicos. Además, como estos dispositivos se distribuyen de forma bastante uniforme, se requiere de soluciones que puedan cubrir cualquier punto de un municipio.

Actualmente, existen varias soluciones para hacer frente a este problema, una de ellas es alimentar los dispositivos mediante acumuladores, pero esta medida solo es valida para dispositivos de baja demanda energética. En otros casos como puntos de carga para vehículos eléctricos, se ha desplegado redes dedicadas a esta finalidad, pero esto puede suponer un inconveniente en cuanto a tiempo y dinero. Finalmente, en lugares donde se tiene acceso a la red del alumbrado público, se ha hecho uso de sistemas de alimentación ininterrumpida basada en baterías, donde se almacena energía durante la noche para poder estar disponible durante el día.

A modo de ejemplo, en la figura 1 se puede observar un semáforo para peatones alimentado con este tipo de sistemas; como un despliegue de una red de electricidad puede no ser óptimo, se ha optado por colocar un sistema de alimentación ininterrumpida que se alimenta con la red de alumbrado público que se encuentra en la misma farola.

Figure 1. Semáforo peatonal alimentado con un sistema de alimentación ininterrumpida – Fuente: Wikipedia

Estos sistemas de alimentación ininterrumpida, actualmente tienen un uso muy extendido, subministrando corriente desde semáforos hasta complejas redes de video vigilancia. En este último caso, la garantía de un suministro constante y sin cortes es de vital importancia. En este sentido, se requiere de un sistema de control y monitorización.

1.2 Antecedentes

Nos encontramos ante dos ayuntamientos municipales del Maresme, concretamente los de Cabrera de Mar y Sant Vicenç de Montalt, que poseen redes de cámaras de video vigilancia para la policía local. Más concretamente, en las que solo disponen de acceso a la red de alumbrado público, la empresa Sintelec SL tiene la concesión del abastecimiento de corriente eléctrica de forma ininterrumpida para estas cámaras.

Actualmente, existen seis de estos puntos instalados y mantenidos por la empresa. Básicamente, todos están compuestos por un interruptor diferencial a modo de protección, un cargador de baterías, una batería y, finalmente, un inversor para proporcionar 220V a las cámaras. A día de hoy se encuentran totalmente operativos.

En la figura 2 se puede observar uno de estos puntos. Principalmente, están compuestos por un armario que alberga todos los sistemas electrónicos, y el mástil situado al lado de la carretera, que contiene las cámaras de videovigilancia y la antena del radioenlace.

Figure 2. Punto de videovigilancia situado en Sant Vicenç de Montalt – Fuente: Propia

Durante toda su etapa de funcionamiento, se han observado fallas de funcionamiento con el inconveniente que esto supone. Las causas son diversas; En primer lugar y con más frecuencia en episodios de tormentas eléctricas, los interruptores diferenciales se desarman de forma totalmente impredecible, provocando un corte de suministro pasadas unas horas. En otros casos y de forma menos frecuente, simplemente se deja de recibir suministro eléctrico a través de la red por algún problema en la red de alumbrado público. Finalmente y como caso menos frecuente, algún componente del sistema de alimentación falla y deja de subministrar corriente.

Todos estos casos expuestos suponen una vulnerabilidad de la seguridad de la ciudad ya que el sistema de videovigilancia deja de funcionar. Es aquí donde nace la necesidad de anticipación al fallo de suministro y también de conocer las causas de estos posibles fallos. Como existe una gran cantidad de emplazamientos y muy dispersos, es necesaria algún método remoto para obtener esta información.

1.3 Objetivo del trabajo

Para este trabajo se plantean una serie de objetivos a cumplir, en primer lugar se presentarán los que pertenecen a requerimientos técnicos. A continuación, como se trata de un trabajo final de máster, se describirán los objetivos académicos. Finalmente, se concluirá el apartado nombrando los objetivos relacionados con el servicio que se quiere ofrecer.

1.3.1 Objetivos técnicos

En primer lugar, se propone el objetivo principal; Consiste en diseñar un sistema que cumpla con las necesidades de la empresa interesada, esto es: la predicción, la detección y la identificación de fallos en los sistemas de alimentación ininterrumpida ya existentes. La verificación de este punto se determinará en función de si el sistema es capaz de minimizar la probabilidad de fallo de suministro eléctrico a la red de sistemas de alimentación ininterrumpida.

Como segundo objetivo, se pretende usar las ultimas tendencias tecnológicas. Este punto implica el uso de Software y Hardware de fuente abierta, ya que es una de las principales tendencias a nivel global. Además, aunque sea un trabajo desarrollado únicamente por una persona, se usará el control de versiones GIT. Este tipo de desarrollo, a parte de tener muchas ventajas en trabajo en grupo, permite tener un histórico de los cambios en el código.

Siguiendo con el segundo objetivo, se pretende usar frameworks de ultima generación. Esto tiene numerosas ventajas sobre el sistema; En primer lugar, el sistema quedará obsoleto tecnológicamente mucho mas tarde, por lo tanto, dispondrá de soporte durante mas tiempo. Por otra parte, los frameworks de ultima generación son lo que gozan de una mayor comunidad de desarrollo y, en consecuencia, es mas sencillo y fácil de programar con ellos. Como último punto de más relevancia, son los que más compatibilidad y retro-compatibilidad ofrecen frente otros frameworks.

El siguiente objetivo consiste en dotar al sistema de cierta robustez. Esto implica que, en caso de caída de la red de fallo de suministro, el sistema sea capaz de volverse a poner en funcionamiento de forma totalmente autónoma. Por otra parte, al tratarse de un sistema que probablemente use software de desarrollo propio, es decir, susceptible a fallos, se procurará también dotarlo de cierta robustez frente a posibles defectos del código que puedan ocasionar excepciones. Esto es muy importante en sistemas autónomos, ya que si se interrumpe un proceso, en vez de pararse el proceso se debe reiniciar sin ningún tipo de intervención humana.

Otro objetivo muy importante es la seguridad del sistema. Al tratar con información sensible, es muy importante impedir que terceros puedan acceder a los datos internos. A modo de ejemplo, si valores como el voltaje de la batería cae en manos de personas o entidades

delincuentes, esto podría suponer una vulnerabilidad a la seguridad del pueblo, ya que esta información podría ser usada para determinar una franja horaria en la que las cámaras de videovigilancia no funcionen debido a el agotamiento de la batería.

Como último objetivo dentro de los técnicos, se procurará un diseño que permita tanto escalabilidad vertical como horizontal. Esto es muy importante en una infraestructura de IoT, ya que si en un futuro se requiere ampliar el sistema con mas nodos o más usuarios (escalabilidad horizontal), se pueda hacer sin tener que rediseñar por completo el sistema. De la misma forma, si en un futuro se requiere ampliar el sistema con alguna funcionalidad adicional en los nodos (escalabilidad vertical), como un procesado en función del tiempo, se pueda hacer sin un rediseño completo o un costoso despliegue.

1.3.2 Objetivos académicos

A nivel académico, se proponen también una serie de objetivos. El primero de estos puntos es ser capaz de crear un sistema E2E completo. Esto implica crear un sistema que involucre conceptos desde hardware de bajo nivel como podría ser un divisor resistivo. Pasando también por microcontroladores y microcomputadores, por ejemplo una Raspberry Pi con un microservicio Back-End programado a medida. Llegando finalmente a una aplicación Front-End que esté totalmente orientada a un usuario genérico, es decir, que valore entornos fáciles de usar con la información muy simplificada y bien estructurada.

Otro objetivo que se propone a nivel académico, es poder consolidar y ampliar conocimientos ya adquiridos en el Máster de Ingeniería de Telecomunicación. En estos estudios, se han adquirido conocimientos de programación de lenguajes de bajo nivel con Arduino en una asignatura cuatrimestral, donde se han visto muchos conceptos avanzados de programación en microcontroladores, por lo que en este trabajo se espera poderlos consolidar e incluso poder verlos desde otra perspectiva.

Además, en el MET también se profundizó bastante en aplicaciones Front-End en una asignatura de desarrollo para aplicaciones móviles en Android. En este punto se pretende no solamente consolidar los conceptos de programación Front-End adquiridos, sino ampliarlos usando alguna otra tecnología.

Como ultimo objetivo de los académicos y a modo de ampliación, se pretende ver un concepto que no se ha tocado en el master, se trata de la arquitectura de microservicios. Este aspecto, más propio de másteres de informática que de telecomunicaciones, es muy importante de cara a la creación de proyectos de IoT. Se trata de la parte que dota a los nodos de un sistema IoT de inteligencia y de cómo interactúan entre si. Es aspecto muy importante porque conecta los elementos del sistema a Internet.

1.3.3 Objetivos de servicio

Como se ha descrito en el apartado de presentación del problema, en numerosos municipios pequeños se controlan los accesos mediante sistemas de videovigilancia, la mayoría alimentados con sistemas de alimentación ininterrumpida, como el servicio que ofrece Sintelec S.L. El listado de empresas que se dedican a este servicio de suministro eléctrico es muy largo, y todas se enfrentan al mismo problema: los cortes eléctricos sin previo aviso.

Dado que a priori no existe un solución comercial extendida, todas estas empresas se encuentran en la misma situación que Sintelec: se detecta que hay un problema en el punto de suministro cuando las cámaras de videovigilancia ya han dejado de funcionar. En esta situación existe claramente una demanda bastante extendida y generalizada de este tipo de servicio para un destacable número de empresas.

Consecuentemente, el último objetivo que se plantea en este proyecto, es el diseño de un servicio que, no solo cubra las necesidades de Sintelec, sino que pueda ser extendido a cualquier otra empresa con la misma problemática.

1.4 Estructura de la memoria

La memoria se estructura en un total de ocho capítulos, el capítulo 2 hace referencia a todas las tecnologías actuales que guardan relación con la problemática que presenta este proyecto; En primer lugar, se analizarán las soluciones ya implementadas que hacen frente de manera directa a los requerimientos. Por otra parte y ya mas enfocado a una solución más personalizada, se estudiarán otras tecnologías no aplicables de forma directa, sino que pueden ser usadas para la implementación de un sistema para cumplir con el objetivo.

Seguidamente, el apartado 3 narra la prueba de concepto que se realiza en el proyecto, dicha prueba tiene dos grandes objetivos; Por una parte se determinará el alcance y las posibilidades del proyecto, por otra y en función de estas conclusiones, se propondrán una serie de desarrollos posteriores para concluir el diseño del sistema.

A continuación, en el apartado 4 se desarrollarán estas funcionalidades adicionales que darán por finalizado el sistema. Después, de presentará este diseño en el apartado 5 con la ayuda de diversos tipos de diagrama y con una prueba de estés donde se garantizará el funcionamiento bajo las demandas del sistema previstas.

Una vez presentado el sistema, en el capitulo 6 se presenta el despliegue final. Este apartado incluirá tanto el montaje de cada uno de los nodos como el despliegue de ellos en los puntos a monitorizar. Seguidamente en el capítulo 7 se detallarán los presupuestos relacionados con el desarrollo del proyecto.

Para finalizar la memoria, en el apartado 8 se narran las conclusiones y el trabajo futuro. Finalmente, se incluyen la bibliografía y los apéndices para concluir el documento.

1.5 Requerimientos del proyecto

A continuación se analizaran todos los requerimientos del proyecto, ya que tienen un papel muy importante para determinar el alcance del mismo. En primer lugar, se estudiaran todos los que están relacionados con la empresa interesada en su implementación. Finalmente, se analizarán todos los referentes a los proyectos de esta naturaleza.

1.5.1 Requerimientos por parte de la empresa

El proyecto debe cumplir dos requerimientos por parte de la empresa interesada; el primero es que tiene que evitar que la batería se agote completamente, hecho que provocaría que las cámaras de video vigilancia dejen de funcionar. Finalmente, tiene que ser eficiente económicamente para ser ventajoso frente otras tecnologías ya existentes.

Por otra parte, el proyecto permite ciertas funcionalidades adicionales que podrían dar un valor añadido al actual sistema de alimentación ininterrumpida, siempre y cuando se respeten los requerimientos del mismo.

1.5.2 Condiciones de entorno

Al tratarse de un proyecto de Internet of Things, existen una serie de condicionantes para el diseño del sistema. Esto es debido a que el emplazamiento del sistema son armarios instalados en la calle, protegidos de lluvia, pero totalmente expuestas a humedades y temperaturas que pueden ir des de negativas en invierno, hasta valores bastante elevados en días soleados de verano.

Estas condiciones, no solamente son importantes para tener en cuenta en el diseño de los dispositivos que irán alojados en los armarios, sino que también pueden ser medidos y, posteriormente, tenerlos en cuenta para realizar estudios de meteorología o la estimación de la vida útil de la batería.

Por su naturaleza, cada uno de los puestos donde se encuentran las baterías a monitorizar, se encuentran cerca de las cámaras de videovigilancia. Por lo que se pueden realizar muchas más mediciones para ofrecer un valor añadido de cara al cliente. Por ejemplo, si desde el mismo armario se captura el sonido, se estaría ofreciendo un nuevo servicio que complementaría el ya actual servicio de video.

Finalmente, también se debe tener en cuenta el coste económico, ya que el coste de cada elemento individual a instalar resultará multiplicado por el número de emplazamientos a monitorizar. Este coste podría tratarse de un valor muy elevado, hecho que podría disparar el presupuesto del proyecto.

2 Tecnologías actuales aplicables

Actualmente, existen varias soluciones al mercado que hacen frente al problema que se plantea en este proyecto y son Plug and Play, es decir, que se conectan y con un mínimo de configuración ya son funcionales. Este será el primer punto que se analizará en este apartado.

Por otra parte, en base a las conclusiones que se obtendrán de este primer punto, se presentarán una serie de tecnologías actuales que, si bien son de un propósito más general, tienen un papel relevante respecto las necesidades que plantean estos sistemas y pueden ser usados para una solución mas personalizada. Siguiendo esta línea, se realizará un estudio del hardware relacionado con los requerimientos proyecto, y después, se analizarán las soluciones software actuales mas adecuadas.

En ambas partes, se justificarán las elecciones hardware y software que se han considerado más oportunas. Finalmente, se planteará un listado de casos de uso a implementar.

2.1 Soluciones ya existentes PnP

La mayoría proporcionan una solución completa, es decir, un hardware y un software que es capaz de hacer frente al requerimiento más básico del proyecto, esto es, monitorizar el estado actual de la batería y de forma completamente remota. Pero entonces surgen varios inconvenientes, el primero es el precio, con valores mínimos de 100 USD la unidad. Después, también existe el problema de que sus interfaces no permiten ningún tipo de implementación de funcionalidades personalizadas, por lo que se cubriría única y exclusivamente este requerimiento.

A continuación, se analizan algunas de las soluciones que se ha considerado que tienen más relevancia y, finalmente, se concluirá el apartado razonando porqué se ha decidido prescindir de estas soluciones.

2.1.1 SiteMonitor Base Unit II, de PacketFlux Technologies

Se trata de un sistema de monitorización con conectividad Ethernet, permite hasta el seguimiento de dos fuentes de voltaje desde 12 a 56VDC, por lo que se ajustaría a una batería de plomo-ácido perfectamente. Además, es capaz de monitorizar temperaturas y controlar de forma remota su propio relé interno, su precio está en la línea de este tipo de productos: 99.95$. Dispone únicamente de Ethernet para su conexión a Internet.

En la figura 3 se puede ver el dispositivo, muy pequeño y fácil de instalar, pero con poca protección frente a condiciones de polvo o lluvia.

Figure 3. SiteMonitor Base Unit II Fuente: store.packetflux.com

2.1.2 Color Control GX, de Victron Energy

Este sistema está diseñado para trabajar con otros componentes de la misma marca, por este motivo tiene conexiones propietarias para ser enlazado con otros módulos Victron Energy. La principal ventaja de este dispositivo si es usado en este escenario es que monitoriza todo tipo de parámetros desde el cargador hasta el inversor. Dispone únicamente de Ethernet para su conexión a Internet. El inconveniente es que requiere realizar toda la instalación con los dispositivos compatibles, además de su precio, superando los 500€.

En la figura 4 se puede ver el producto, tecnológicamente muy avanzado pero poco orientado a ser montado en armarios remotos.

Figure 4. Color Control GX Fuente: victronenergy.com

2.1.3 PowerAgent de Alpha

PowerAgent es un sistema avanzado de monitorización que se centra en el estado de las células de una batería, permite controlar hasta un total de 240 células. A diferencia de los otros sistemas, éste es capaz de obtener parámetros mas específicos como temperatura de cada una de las células, corriente de descarga y corriente de rizado. Además, permite establecer alarmas vía SNMP. Igual que los dos anteriores, dispone únicamente de Ethernet para su conexión a Internet.

Figure 5. PowerAgent

2.1.4 SPM-200 de NEWMAR

Esta, juntamente con el SiteMonitor, son las soluciones que mas se ajustan a las necesidades del proyecto, ya que en su funcionamiento se contempla la monitorización completa de

sistemas de alimentación ininterrumpida. Igual que los anteriores, dispone únicamente de Ethernet para su conexión a Internet

Su interfaz web permite consultar el estado de los distintos equipos electrónicos, como la batería, el inversor y el cargador. Dispone de un histórico de hasta 30 días. La funcionalidad mas interesante de este dispositivo es la personalización de las alarmas, pudiendo llegar a ser configurados hasta 50 avisos mediante e-mail o llamada telefónica.

Finalmente, su principal inconveniente es el precio, superando los 900$ por dispositivo.

Figure 6. SPM-200 Fuente: poweringthenetwork.com

2.1.5 MeteoHub de SmartBedded

Finalmente, se analizará MeteoHub. Ofrece una solución distinta respecto a los otros desarrolladores; SmartBedded comercializa un software base llamado MeteoHub y el usuario es quien elige en cual de sus opciones hardware ejecutarlo, una de ellas es Raspberry Pi.

MeteoHub está orientado a la monitorización de datos meteorológicos y que, además, permite la recolección de datos adicionales como podría ser el voltaje de baterías. El inconveniente es que su funcionamiento requiere de una licencia que, en caso de ser usado en Raspberry Pi, parte de los 120$ en su versión más básica.

2.1.6 Conclusiones

Como se ha podido ver en este apartado, actualmente existen soluciones muy validas que permiten la monitorización de baterías con alarmas para la prevención de fallos en el sistema de suministro, incluso se ha podido observar que hasta contemplan parámetros adicionales como la monitorización de los elementos auxiliares del sistema de alimentación ininterrumpida.

Por otra parte, suponiendo usar la mas sencilla de las soluciones, implicaría un coste de 100€ por nodo mas la mano de obra que supondría el montaje, implementación y mantenimiento del sistema en la empresa. Además de otras desventajas como la imposibilidad de tener una interfaz en español o la imposibilidad de ampliar funcionalidades en un futuro.

Dadas todas estas circunstancias, se ha decidido implementar el proyecto desde cero, es decir, sin partir de ninguna solución comercial ya implementada para tal finalidad. De esta forma no

solo se podrá reducir gastos, sino que además se podrá implementar una solución totalmente adaptada al problema real.

2.2 Tecnologías hardware

En este apartado se compone por tres partes. Primeramente, se presentarán los microcontroladores sobre el cual se plantearán los desarrollos. En segundo lugar, se estudiará qué tecnologías aplicar sobre estas placas de desarrollo con el fin de cumplir con los requerimiento del proyecto, esto es, asegurar la disponibilidad de la batería y dar un valor añadido al proyecto. Finalmente, en base a todas las tecnologías anteriormente analizadas, se estudiaran distintas arquitecturas hardware.

2.2.1 Microcontroladores

Los dos microcontroladores planteados son Arduino y Raspberry Pi. Se han considerado estos porque ambos gozan de mucha popularidad y disponen de una extensa comunidad de desarrolladores. Además, son complementarios y en cooperación pueden cubrir un gran rango de funcionalidades.

2.2.1.1 Arduino

Arduino es una placa basada en un microcontrolador, que básicamente es un circuito donde se le pueden grabar instrucciones. Entonces, mediante su propio IDE, se puede desarrollar programar que permitan interactuar con los interfaces de la placa. Tanto el Hardware como el Software de la marca son de fuente abierta. Además, Arduino dispone de multitud de accesorios como los Shields, que básicamente son módulos adicionales que permiten añadir funcionalidades como conectividad Ethernet, GPS y WiFi entre otros.

Los interfaces que tiene, permiten la captura de señales tanto digitales como analógicas, ya que su microcontrolador dispone de conversores A/D. Además, algunos de sus interfaces digitales pueden ser programados como salidas completamente personalizables, esto permite interaccionar con dispositivos como simples LEDs o actuadores. En la figura 7 se puede observar una representación de la placa Arduino UNO.

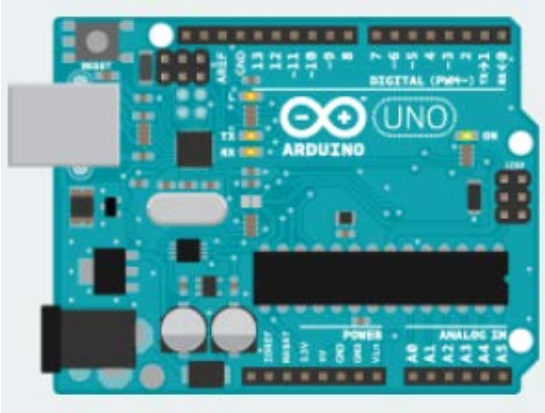


Figure 7. Placa de desarrollo Arduino UNO - Fuente: arduino.cc

2.2.1.2 Raspberry Pi

Raspberry Pi es una placa computadora, su funcionamiento es como el de un ordenador pero con una potencia, consumos y costes mas reducidos. La fundación Raspberry, responsable de su desarrollo, mantiene su sistema operativo oficial: el Raspbian, que básicamente es una distribución Linux Debian adaptada a este pequeño ordenador.

Dispone de las conectividades digitales propias de un ordenador, e incluso una serie de pines GPIO (pines de entrada/salida de propósito general), pero no dispone de ningún tipo de conversión analógico/digital como si tiene Arduino.

En la figura 8 se puede ve la placa Raspberry Pi, con todos los elementos de un ordenador, siendo un poco mas grande que una tarjeta de crédito.

Figure 8. Raspberry Pi 3 Model B+ - Fuente: Fundación Raspberry

Tiene propósitos muy distintos a Arduino; Mientras que el microcontrolador de Arduino tiene la potencia de procesamiento y la memoria suficientes para ejecutar códigos muy sencillos que permiten realizar funcionalidades muy básicas con sus interfaces analógicas y digitales, la Raspberry Pi dispone de un SoC que es capaz de ejecutar un sistema operativo y realizar tareas tan pesadas como reproducir Streamings de 4K o complejas funciones de red.

2.2.2 Soluciones hardware para obtención de datos

2.2.2.1 Tecnologías relacionadas con la disponibilidad de la batería.

En este proyecto se cuenta con baterías de plomo-ácido de 12V, pero también podría ser que en un futuro la alimentación se realice con otro tipo de suministros, el dispositivo de medición debe poder adaptarse a cualquier tipo de alimentación.

En primer lugar, es conveniente conocer el estado de carga (SOC) de la batería, ya que es un indicador que está completamente relacionado con el suministro; si la batería tiene carga, hay suministro a las cámaras de seguridad, en caso contrario, no hay suministro.

En términos generales, para medir el SOC (Chang, 2013) de cualquier batería, una buena aproximación consiste en medir el voltaje en circuito abierto (OCV) en bornes de la batería. Después, existe una relación entre voltaje y SOC que varía dependiendo del tipo de batería: en el caso de las baterías plomo-ácido es lineal, pero para otras tecnologías como litio sigue la relación del gráfico de la figura 9.

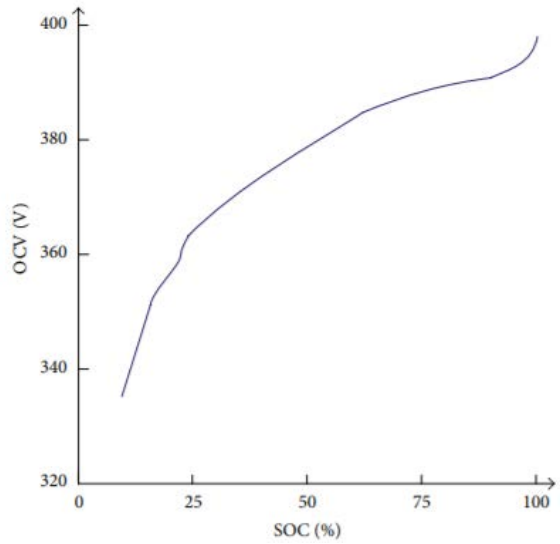


Figure 9. Gráfico que relaciona el SOC y voltaje en circuito abierto para una batería de Litio - Fuente: (Chang, 2013)

Por otra parte, para casos no lineales como el de la figura 9, resulta fácil calcular la SOC a partir del OCV mediante procesamiento software. Posteriormente, conociendo la SOC y un histórico de este se pueden calcular otros parámetros, como tiempo restante de la batería y vida útil.

Para el caso específico de baterías de plomo-ácido, adicionalmente existe un parámetro determinante de la vida útil, este parámetro es la profundidad de descarga (DOD). Resulta que en este tipo de baterías, existe una correlación entre la profundidad de descarga y el número de ciclos de carga y descarga, esto implica que cuando más se descarga una batería en su ciclo de carga-descarga, más corta será su vida útil. En una publicación online de una empresa instaladora (EnergyMatters, 2018) se ha determinado que un DOD del 50% es el más óptimo de almacenamiento frente al coste.

Además, para conocer la disponibilidad de la batería se puede considerar otras variables, una de ellas a tener en cuenta es la tensión en la entrada del cargador de baterías y a la salida del inversor. Estos dos parámetros informan si el sistema de alimentación recibe y/o proporciona suministro o no. De esta forma se permite detectar y identificar la ubicación de las averías de forma remota.

Como todos estos parámetros descritos anteriormente son voltajes, se pueden obtener mediante dispositivos de medida analógicos. Para este proyecto se considerará la plataforma Arduino, ya que dispone de varias entradas analógicas con una resolución de medida más que suficiente para medir el voltaje de la betería. El único inconveniente es que la cuantificación se realiza de 0 a 5V, por lo que se requerirá de un circuito adaptador previo que comprima la señal.

Finalmente, se considerará conocer el estado del interruptor diferencial que se encuentra en el mismo armario. Esto permitirá determinar el origen de posibles fallos en el suministro eléctrico e incluso volverlo a rearmar mediante un actuador mecánico. Para conocer su estado se puede hacer mediante dos métodos; o bien con la obtención de una imagen y su procesamiento, o bien mediante un sensor de posición.

Para conocer su estado mediante un sensor de posición, existen en el mercado distintos tipos de pulsadores que pueden incorporarse en el interruptor diferencial, pero el más adecuado es el interruptor de final de carrera. Puede ir colocado de forma que cuando baje el diferencial cierre el circuito. El montaje puede ser un poco complicado ya que tiene que ser montado sobre los actuales interruptores diferenciales sin que impida el rearmamento del mismo. Finalmente, tiene como ventaja que puede ser conectado directamente en una entrada digital de Arduino.

Por otra parte, si se desea conocer el estado mediante imagen, se debe desarrollar un sistema con suficiente potencia de cálculo y que ofrezca robustez ante falta de iluminación. Para casos de reconocimiento de colores y/o formas como la posición de un interruptor diferencial, el estudio de (González, 2016) relata que sería conveniente usar la librería OpenCV juntamente con un dispositivo que pueda soportarla como Raspberry Pi. El inconveniente, como se puede ver en el estudio, es la implementación, ya que los sistemas de reconocimiento de imagen son difíciles de desarrollar y posiblemente se escape del alcance de este proyecto.

Finalmente, asumiendo que se conoce que el interruptor diferencial está desarmado, es posible volverlo a rearmar con un actuador mecánico. Por la naturaleza de los interruptores diferenciales, se necesita uno del tipo lineal y capaz de ejercer fuerzas desde 150N hasta 1000N. Adicionalmente, es fácil del controlar mediante la plataforma Arduino. En la figura 10 se puede observar un actuador de 500N apto para la plataforma Arduino.

Figure 10. Actuador lineal – Fuente: pololu.com

2.2.2.2 Tecnologías de soporte a la disponibilidad de la batería

Una vez se puede asegurar el suministro de electricidad a las cameras de video vigilancia, se puede aprovechar el plan de despliegue y la actual plataforma para añadir funcionalidades de soporte a la principal.

En primer lugar, se estudiará como estimar la vida útil de la batería. Para esta estimación, la temperatura es un parámetro directamente relacionado con el porcentaje de vida útil que se pierde al cabo de un año (Bhatt, 2013). Como se explica en el artículo, con una temperatura constante de 35 grados la batería puede experimentar una reducción de su vida útil a la mitad, y con 44 grados, se reduce un 83%.

En la plataforma Arduino existe una multitud de métodos para medir la temperatura. Para este proyecto se considera el modulo de la figura 11. Se trata de un sensor digital de temperatura y humedad que integra el estándar de comunicación I2C. Además, la librería oficial Adafruit de Arduino permite obtener directamente los valores de temperatura y humedad sin tener que implementar nada.

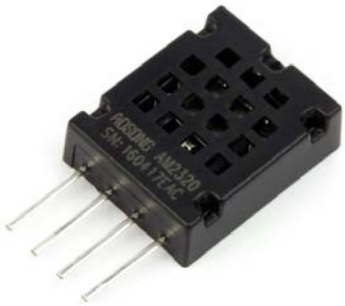


Figure 11. Módulo AM2320 – Fuente: akizukidenshi.com (fabricante)

En segundo lugar, considerando que en cada punto remoto se encuentran una serie de dispositivos conectados a la red, como la cámara de video vigilancia, se puede realizar un control y/o gestión de estos elementos. Como control se puede hacer un ping para asegurar que el dispositivo se encuentra en la red y como gestión se puede mandar comandos mediante SSH como por ejemplo encender/apagar modo visión nocturna.

Esta funcionalidad requiere únicamente de funciones de red por lo que la Raspberry Pi por si sola ya sería capaz de realizar la tarea.

2.2.2.3 Tecnologías para dar un valor añadido

Como estos armarios se encuentran situados cerca de las cámaras de video vigilancia, puede ser valorable complementar las funcionalidades de la grabación de imágenes. Una buena opción es la captura y la emisión de sonidos.

Esta funcionalidad se puede implementar con Raspberry Pi y un dispositivo de entrada de sonido como un micrófono USB y, finalmente, un dispositivo de salida de sonido amplificado como un altavoz exponencial o una sirena, ambos de montaje en farola. En las figuras 12 y 13 se pueden ver el altavoz y la sirena, respectivamente. Ambos de montaje en mástil.

Figure 12. Altavoz exponencial - Fuente: farnell.com

Figure 13. Sirena - Fuente: https://cpc.farnell.com/

Otra consideración, por la naturaleza del proyecto, es que el despliegue de estos armarios es a nivel provincial, por lo que se tendrá que abastecer a varios puntos distribuidos de forma aleatoria, con lo que se cubre bastante territorio. En estas circunstancias, se puede hacer una adquisición de datos como el posicionamiento y, con los datos de temperatura y humedad que ya se tienen, se puede hacer un estudio meteorológico a nivel municipal.

Por este motivo se ha considerado la incorporación de un módulo GPS de bajo coste, concretamente el NEO6MV2, mostrado en la figura 14. Se trata de un dispositivo muy popular con comunicación serie y alimentación de 3.3V que se integra muy fácilmente con Arduino.

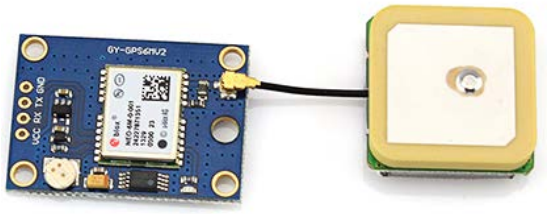


Figure 14. Modulo NEO6MV2 integrado en placa de desarrollo con antena cerámica – Fuente: Fritzing Project

Por otra parte, cada vez más ayuntamientos desean monitorizar datos de polución; como es el caso de Barcelona, donde están siendo pioneros en controles de contaminación. Esto sería fácilmente aplicable a este proyecto, ya que cada estación puede proporcionar, además de datos meteorológicos, información sobre la calidad del aire.

Sensores como el MQ-135 representado en la figura 14, son utilizados en la plataforma Arduino para determinar la calidad del aire y presencia de varios elementos nocivos.

Figure 15. Sensor MQ-135 con circuito adaptador I2C – Fuente: olimex.com

2.2.3 Conectividad

Como en la mayoría de proyectos de IoT, es un requerimiento indispensable dotar a los nodos de conexión a Internet, por este motivo se ha realizado una selección de tecnologías de conectividad con relevancia en IoT. Es importante tener en cuenta que la ubicación de los nodos es muy dispersa, por lo que por razones de optimización de despliegue únicamente se tendrán en cuenta las tecnologías inalámbricas.

A continuación se presentará una serie de tecnologías inalámbricas que ha considerado que tienen relevancia en este proyecto.

2.2.3.1 ZigBee

ZigBee es una tecnología inalámbrica que se caracteriza principalmente por tener el menor consumo energético posible con el fin de maximizar la vida de la batería. Además, goza de gran robustez, es decir, seguridad en sus comunicaciones y una muy buena escalabilidad, ya que permite topologías en estrella, malla y árbol. En este sentido se ajusta bien a la temática de este proyecto, ya que por su naturaleza es una topología de malla y además, los nodos se encuentran alimentados con batería, por lo que interesa maximizar su autonomía.

Por otra parte, otros aspectos quedan un poco sacrificado, esto resulta en tasas de transferencia de datos bajas, con un máximo de 250kbps y cortos alcances, con máximos que se sitúan entre 10 y 100 metros. Como se puede ver en la figura 16, los módulos ZigBee son pequeños y normalmente integran la misma antena en el dispositivo.

Figure 16. Módulo ZigBee XBee – Fuente: sparkfun.com

2.2.3.2 WiFi

Las redes WiFi destacan por su popularidad, siendo instaladas en multitud de entornos, tanto domésticos como profesionales. Dispone de buenas tasas de transferencia de archivos, con cifras habituales de 150Mbps y alcances moderados, del orden de 50 metros. Goza de buena escalabilidad, soportando multitud de dispositivos en una misma red con poca degradación de la calidad.

Como puntos negativos, los dispositivos que operan con redes WiFi tienen consumos energéticos elevados, por lo que se reduce bastante la autonomía de los sistemas alimentados con batería. Como se puede ver en la figura 17, los módulos WiFi tienen un tamaño contenido y, de la misma forma que con ZigBee, también pueden integrar la antena.

Figure 17. Módulo WiFi ESP8266 - Fuente: sparkfun.com

2.2.3.3 WiMAX

WiMAX es otra tecnología inalámbrica pero, a diferencia de las otras, destaca por sus amplias coberturas, llegando a cubrir distancias de hasta 70 km. Esta característica es muy interesante en este proyecto, ya que la ubicación de implementación de este proyecto, el Maresme, está caracterizado por estar situado en el Litoral, una sierra donde la mayoría de sus municipios disponen de visión directa. En estas circunstancias, solo con un punto de acceso WiMAX, se ofrece cobertura para la mayoría de nodos.

En la figura 18 se puede ver que, a diferencia de las otras tecnologías presentadas, los módulos son bastante aparatosos y su implementación resulta más dificultosa que las otras tecnologías.

Figure 18. Módulo WiMAX directivo – Fuente: wikipedia.org

Por otra parte, dispone de buenas tasas de transferencia de archivos. Siendo normal disponer de radioenlaces de 30Mbps constantes. Como desventajas, solo se puede establecer conexión en puntos con visión directa con el punto de acceso, además, cuenta con unos consumos energéticos bastante elevados.

2.2.3.4 Bluetooth

Esta tecnología entra en el grupo de conexiones de bajo consumo energético, en consecuencia cuenta también con bajas tasas de transferencia y alcances moderados. Como se puede apreciar en la figura 19, tiene tamaño reducido igual que las otras tecnologías de su categoría.

Figure 19. Módulo Bluetooth HC06 – Fuente: rs-online.com

Es bastante similar al ZigBee, pero soporta menos nodos y tiene el consumo eléctrico un poco mas elevado. En cambio tiene tasas de transferencia mas altas, con máximos de hasta 3000 kbps frente a los 250 kbps del ZigBee.

2.2.3.5 Red de telefonía móvil

Esta opción destaca por su alta disponibilidad por todo el territorio nacional. Como principal ventaja dispone de una alta tasa de transferencia de datos, llegando hasta los 10Mbps en el caso de la tecnología 4G. Además, a diferencia de las otras, es la opción con más proveedores de servicio, por lo que se puede encontrar planes de datos muy personalizados a precios muy competitivos. Además, los módulos tienen un tamaño reducido tal i como se puede ver en el ejemplo de la figura 20, siendo un módulo 3G con antena externa.

Figure 20. Módulo 3G SIM5300E – Fuente: rs-online.com

Como punto negativo, cualquier de sus tecnologías tienen unos consumos energéticos algo elevados.

2.2.3.6 LoRaWAN

Se trata de una tecnología de bajo consumo, pero a diferencia de las anteriormente presentadas, esta presenta unos alcances bastante mas amplios, con cifras de 2 a 5 km en el caso de entornos urbanizados y de hasta 15 km en largas extensiones sin obstáculos. Como se puede ver en el módulo de la figura 21, los módulos son de tamaño bastante reducido.

Figure 21. Módulo LoRaWAN MB-LR1272 – Fuente: shop.sodaq.com

En referencia a las tasas de transferencia de datos, esta es la tecnología menos favorable, con velocidades que se sitúan por debajo de los 50 kbps. Se trata de una tecnología para transferir documentos muy ligeros pero en extensiones amplias y con un bajo consumo.

2.2.3.7 Sigfox

Finalmente, se presenta una de las conexiones con menor consumo energético de todas. Teniendo alcances que se sitúan entre el WiFi y el WiMAX, tiene consumos que no llegan a los

100 micro vatios. Esto maximiza al máximo la autonomía de la batería pero sacrificando la velocidad de transferencia de datos, que en este caso se encuentra al orden de centenares de bits por segundo. Como se puede observar en la figura 22, los módulos son realmente pequeños.

Figure 22. Módulo Sigfox – Fuente: farnell.com

2.2.3.8 Conclusión

En este proyecto, los nodos están ubicados de forma dispersa, por lo que la topología que mas se ajusta es en forma de malla. Por otra parte, los nodos están alimentados con baterías, pero estas son de 12V y 100Ah y se cargan a diario, por lo que ninguna de las anteriores tecnologías supondría un problema para la autonomía. En cuanto a tasa de transferencia de archivos, los nodos requieren de altas tasas por el Streaming de las cámaras de videovigilancia.

Después de analizar las características técnicas de las tecnologías inalámbricas anteriormente listadas, se ha considerado que las únicas que pueden ser válidas son el WiFi, el WiMAX y la red de telefonía móvil. Como todos los puntos ya disponen de conexión WiMAX, se aprovechará la infraestructura ya existente, pero para nuevos puntos donde no sea posible un radioenlace con el punto de acceso WiMAX se optará por un plan de datos 4G.

2.2.4 Arquitecturas óptimas para la gestión de la información

En el apartado anterior, se ha analizado un conjunto de magnitudes que pueden ser de interés. Después de hacer una valoración completa, se observa que los parámetros de medida más básicos pueden ser cubiertos simplemente utilizando un Arduino con conectividad a Internet, posteriormente se analizará este caso.

Adicionalmente, si se desean funcionalidades más avanzadas que incluyan algún tipo de procesamiento o diseñar un sistema que permita escalabilidad vertical, se requerirá de un microcomputador más potente como la Raspberry Pi para complementar el Arduino.

2.2.4.1 Células basadas en Arduino

En este primer caso, si se considera usar Arduino en solitario como plataforma de adquisición de datos, cumpliría perfectamente con los dos objetivos del proyecto, es decir, monitorizar el estado de la batería y con un coste muy reducido. Pero se debe tener en cuenta que éste tiene unas funcionalidades bastante limitadas.

Como desventajas, en primer lugar se tiene unas funcionalidades de red muy básicas, además presenta un procesamiento de datos poco potente. Por otra parte, como no dispone de una memoria no volátil pensada para almacenar ficheros pesados, las posibilidades de persistencia de datos son muy limitadas. Por otra parte, implica más sencillez en cuanto a implementación

y, consecuentemente, presenta menor probabilidad de fallos, y por lo general, menor consumo energético.

Para la implementación, se tiene que hacer uso de un Shield de Ethernet y un Shield PCB adicional donde implementar los circuitos de alimentación y de adaptación en los sensores.

2.2.4.2 Células basadas en Arduino y Raspberry Pi

En este segundo caso, se considera usar de forma adicional una Raspberry Pi porque tiene una gran ventaja: su considerable aumento de potencial, permitiendo implementar una topología tipo Fog, es decir, que cada elemento de la malla sea capaz de procesar de forma totalmente independiente. Además, permite una buena escalabilidad, tanto vertical como horizontal, pudiendo realizar cambios en el software de forma remota, hecho que Arduino por sí solo no es capaz de hacer. Finalmente, los nodos podrían beneficiarse también de una buena capacidad de procesamiento de datos y dotarlos de persistencia de datos.

Por si sola, Raspberry Pi no puede hacer frente en solitario a magnitudes analógicas como voltajes y otros sensores por lo que necesita como complemento algún tipo de conversión analógico-digital. Por este motivo, se considera implementar una unión Arduino-Raspberry Pi. Por otra parte, existe una gran comunidad de desarrolladores que implementan proyectos en este mismo conjunto.

Por otra parte, este tipo de implementación tiene una serie de desventajas, en primer lugar resulta en una configuración costosa de implementar, también aumenta la probabilidad de fallo del sistema al contener mas elementos. Consecuentemente también existe un mayor consumo energético y complicidad de implementación.

2.3 Tecnologías Software

Una vez analizadas todas las tecnologías hardware que pueden ser de interés para el proyecto, es necesaria la elección de un software para poderlo operar. Para su elección, dada la actual tendencia, se ha procurado usar tecnologías punteras de código abierto. Esto tiene muchas ventajas como una gran comunidad de desarrolladores, gratuidad del servicio y un alto grado de seguridad.

A continuación se van a presentar y justificar distintas soluciones software que pueden ser aplicadas a las dos partes que componen el proyecto: el Back-End y el Front-End.

La situación actual del Back-End es que nos encontramos ante dos plataformas que necesitan su software específico; el Arduino y la Raspberry Pi.

2.3.1 Back-End

La capa de software Back-End tiene una función muy importante; se trata del nexo de unión entre el hardware del nodo y la aplicación Front-End que se ejecutará en los dispositivos finales. Este software esta dividido en dos partes, la correspondiente a Arduino y la correspondiente a Raspberry Pi.

En este apartado, primeramente se analizará el software mas conveniente para ser implementado en Arduino y se nombrarán algunas consideraciones para desarrollarlo.

Finalmente, se justificará qué arquitectura se debe implementar y, siguiendo esta línea, se propondrán las tecnologías mas adecuadas para su uso en este proyecto.

2.3.1.1 Arduino

Esta plataforma se caracteriza por no tener ni una abundante potencia de calculo ni gran capacidad de almacenamiento de datos, además de no permitir actualizaciones en caliente. En este sentido se reducirá su funcionalidad a únicamente la adquisición de datos y su posterior envío a través de su interface serie, sin ningún tipo de procesamiento.

El microcontrolador Arduino ha sido programado mediante su propio IDE, que es completamente libre. Adicionalmente, se ha usado librerías tanto internas como externas a la propia plataforma Arduino con el fin de poder interaccionar con los distintos periféricos hardware. A continuación de listarán todas las librerías externas.

En primer lugar se ha usado la librería AM2320.h, su función es interactuar con el sensor de temperatura y humedad, la librería proviene de Adafruit y es de código libre. También se ha usado la librería TinyGPS++.h, en este caso es para comunicarse con el sensor de posicionamiento GPS neo 6m.

2.3.1.2 Raspberry Pi

2.3.1.3 Sistema operativo

Esta otra plataforma permite muchas posibilidades, pero para ello se tiene que escoger el software adecuado. En primer lugar, se tiene que elegir el sistema operativo. El mas recomendable según el fabricante y la comunidad de desarrolladores es el oficial de la fundación Raspberry: Raspbian. Se trata de una distribución Linux basada en Debian. Entonces, a partir de aquí se diseñará toda la lógica de aplicación.

2.3.1.4 Software de servidor

Para implementar todas las funcionalidades Back-End en la Raspberry Pi, interesa un framework que pueda albergar todos los requerimientos, así como permitir una buena escalabilidad y mantenimiento. En este sentido existe multitud de entornos de capa de servidor que puedan servir para la Raspberry Pi como Perl, ASP.NET y Node.js, pero debido principalmente a que el microcontrolador es un computador un poco limitado como servidor, Node.js es la opción mas recomendable ya que es muy ligero y eficiente.

Por otra parte, Node.js tiene una serie de características que lo ponen en una situación ventajosa respecto otras tecnologías. En primer lugar, la plataforma permite concurrencia de distintos procesos. Además, permite usar módulos mediante un gestor de paquetes propio: el Node Package Manager, que permite instalar y usar multitud de programas que se albergan en su propio repositorio. Gracias a los siguientes módulos se ha logrado una buena integración de todos los requerimientos en un único proyecto Node.js.

Finalmente, como el entorno de ejecución es totalmente remoto con posibilidades limitadas de gestión presencial, se necesita un gestor para el proceso Node.js que, en caso de algún tipo de fallo o malfuncionamiento, el mismo gestos sea capaz de reiniciar el proceso.

En este sentido el gestor estrella de Node es PM2, que en su versión gratuita permite la gestión autónoma del servidor. Y en su versión de pago permite una monitorización y gestión avanzadas.

En la figura 23, se puede observar el portal de gestión, donde a modo de ejemplo se muestran todos los procesos Node.js de un servidor con datos de importancia como veces que el proceso ha sido reiniciado, uso de la memoria y uso del procesador.

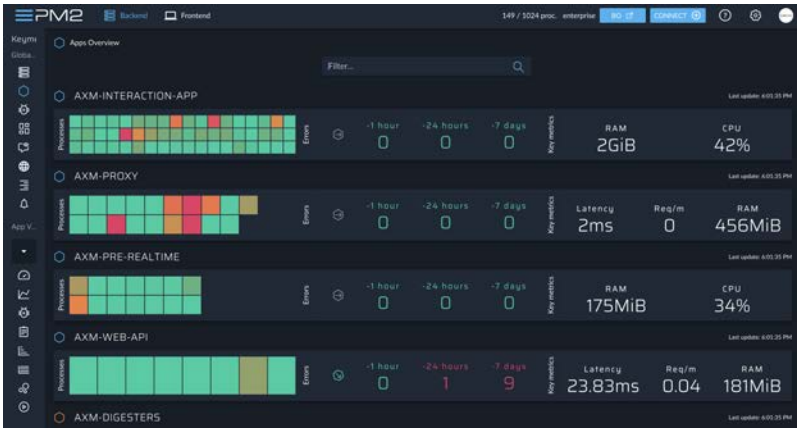


Figure 23. Ejemplo del gestor de procesos PM2 – Fuente: PM2

2.3.1.5 Servidor HTTP

Como primer requerimiento, es indispensable establecer algún microservicio para poder interaccionar con el nodo desde el exterior. En este caso, donde se requiere exponer datos muy concretos como lecturas de sensores, es una practica bastante habitual exponer esta información a través de un documento JSON accesible a través del protocolo http. NPM dispone del módulo Express, un framework para desarrollar aplicaciones web minimalistas, en este caso es de gran utilidad ya que permite responder a eventos http de forma muy sencilla.

2.3.1.6 Servicio de túnel

En segundo lugar, es muy importante tener accesibilidad desde cualquier punto de Internet. Más concretamente dando acceso al puerto 22 donde se encontrará la conexión SSH de la Raspberry Pi, de esta forma se podrá realizar tareas administrativas de forma remota. En segundo lugar, se debe dar acceso también al puerto 80, que corresponde al protocolo HTTP se podrá interactuar con el microservicio mencionado el párrafo anterior. En este sentido NPM tiene un módulo llamado Ngrok: se trata de un túnel que permite exponer a Internet múltiples conexiones hacia distintos puertos del Localhost de la Raspberry Pi sin limitantes como Firewalls o Routers.

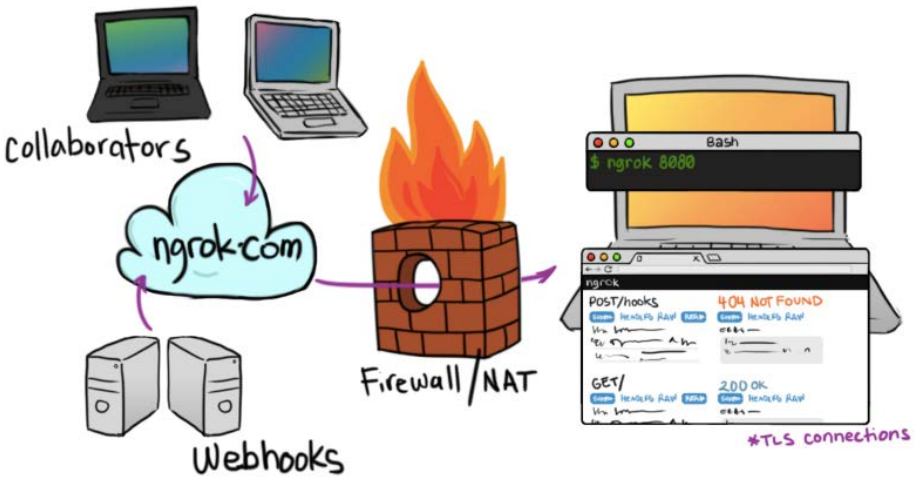


Figure 24. Esquema de funcionamiento de Ngrok – Fuente: ngrok.com

El túnel de Ngrok es una alternativa a los conocidos servicios de DNS dinámicos, como NO-IP.com, o Dyndns.com. Este tipo de servicios, basan su funcionamiento en una redirección de un puerto desde el Router hacia el dispositivo final situado dentro de la red local, por lo que requiere de al menos una configuración en Router, y esto, rompe el principio de Plug and Play. Ngrok, como se puede ver en la figura 24, hacer el redireccionado de forma totalmente interna dentro del host, por lo que solo con tener acceso a internet es suficiente para establecer el túnel, por esto se ha escogido esta opción.

2.3.1.7 Servicio de persistencia de datos

Para disponer de un histórico de datos como por ejemplo, el valor del voltaje de la batería en función del tiempo, es necesario algún tipo de sistema de almacenamiento de los valores. Según el articulo de (Valenzuela, 2018) en la revista digital Medium, la base de datos es el sistema mas adecuado para microcontroladores Raspberry Pi que adquieren datos de sensores.

Además, en el articulo se explica que en la actualidad existe multitud de estas soluciones, pero en este caso se necesita algo muy concreto: como solo se va a almacenar valores de sensor en función de variables temporales, no es necesario disponer de bases de datos muy extensas ni muy potentes, además, se debe tener en cuenta que como más liviana sea más óptima va a ser para el microcontrolador Raspberry Pi.

En este punto, se ha considerado la bases de datos SQLite, por ser la que mas se ajustan a este proyecto, además, es de las mas populares en Raspberry Pi. Otro punto a favor de esta base de datos, es que es algo mas sencilla de poner en funcionamiento que otras bases de datos mas complejas como PostgreSQL.

2.3.1.8 Servicio de notificación

Por otra parte, desde un punto de vista un poco más funcional, se requiere poder notificar al usuario de alguna forma. Una solución bastante sencilla es mediante un servicio de mensajería instantánea. NPM dispone de un paquete del Bot del Telegram, donde se puede programar

una lógica de interacción con el usuario, permitiendo enviar mensajes delante de alguna circunstancia o responder de forma predefinida ante cualquier mensaje dirigido a él por parte del usuario. De esta forma, desde cualquier dispositivo con Telegram se puede obtener información del nodo sin necesidad de acceder a complejas direcciones URL.

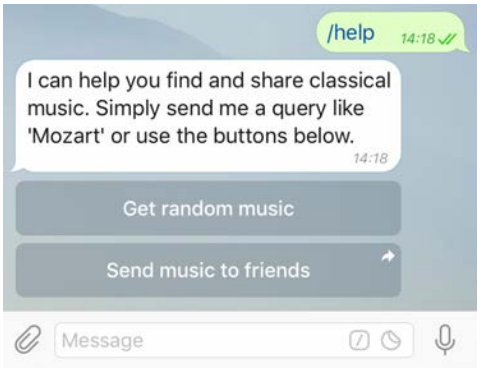


Figure 25. Ejemplo de un Bot de Telegram – Fuente: Telegram API

Como se puede observar en la figura 25, el funcionamiento del bot consiste en procesar la información que empieza con una barra, entonces, responde lo que haya sido programado, con la posibilidad de añadir respuestas predeterminadas.

2.3.2 Front-End

Para que el usuario final pueda interactuar con el sistema se ha considerado crear una aplicación multiplataforma, de esta forma podrá ser usada tanto en dispositivos móviles de cualquier tipo como en tabletas como en ordenadores. En la actualidad existe multitud de frameworks para desarrollar aplicaciones web, de las cuales se destaca Angular, React y Vue por ser los mas populares de ultima generación.

En la revista digital Codearmy (Martin, 2019), se hace una comparación de ambas y, para casos generales, se concluye el artículo mencionando a Angular como mejor opción, ya que integra todo lo necesario para un desarrollo básico de una aplicación web; enrutamiento, templates (html) ya incluso testing en el mismo package. Consecuentemente, se ha escogido implementar una Web App basada en Angular 6, ya que es un framework que por sus características se ajusta perfectamente a las necesidades del proyecto.

El factor mas importante para su elección es que al tratarse de un lenguaje MVC, se ajusta perfectamente a este proyecto de IoT ya que permite separar perfectamente la estructura de las células, como la adquisición de sus datos y finalmente su visualización. Por otra parte al ser un framework muy popular y de ultima generación goza de muy buen soporte en internet.

Finalmente, centrando el enfoque en la parte práctica, es importante considerar que está diseñado para poder crear aplicaciones web con un bajo esfuerzo pero permitiendo muchas funcionalidades y gran escalabilidad de las aplicaciones.

2.3.3 Cifrado E2E

El cifrado extremo a extremo, es un sistema donde únicamente el emisor y el receptor pueden leer los mensajes. Esto evita espías potenciales, incluidos los ISP, de esta forma se evita que datos confidenciales puedan ser obtenidos por terceras personas.

En la actualidad existe una gran cantidad de algoritmos de cifrado orientados a servicios de red como Blowfish, Twofish, RIJNDAEL, Serpent o AES y RC6 (Tovar, 2010). Además, se ha priorizando la facilidad de integración en el sistema formado por Node.js en la parte de Back-End y Angular 6 en la parte de Front-End. Entonces, se ha optado por usar una encriptación tipo AES, ya que se trata de una de las mas usadas en este tipo de aplicaciones, además, según (Pousa, 2011) se trata de un cifrado calificado como seguro por Estados Unidos y sin ataques eficientes conocidos con éxito.

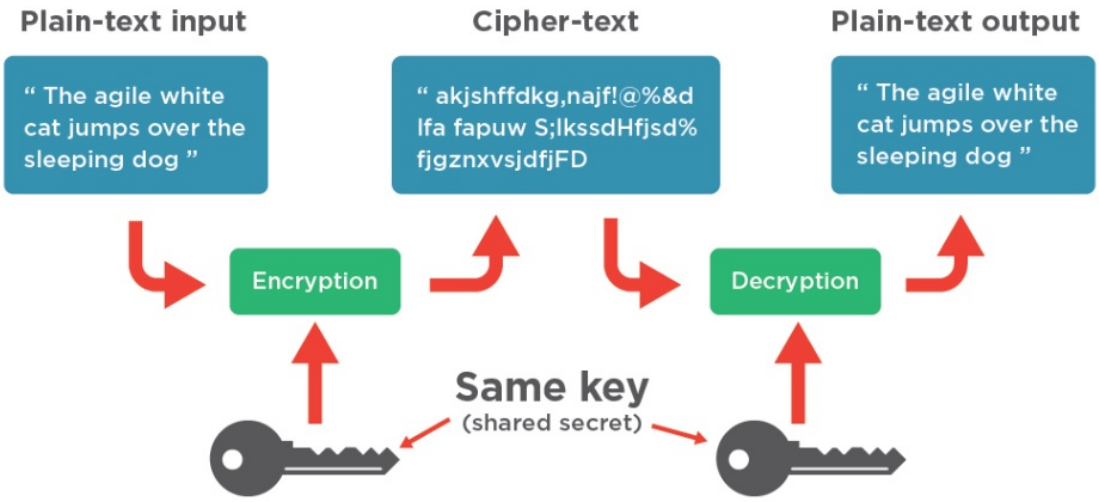


Figure 26. Esquema de funcionamiento del cifrado AES – Fuente: Transcend

Como se pude observar en la figura 26, el estándar de cifrado AES, funciona a partir de una clave de cifrado y un contenido a cifrar. Una vez convertidos los datos, estos son inteligibles y no pueden ser leídos por ninguna persona que no sea la destinataria, que dispone de la misma clave de cifrado. Afortunadamente, tanto Node.js como Angular disponen de una multitud de librerías que permiten el cifrado y el descifrado únicamente aportando el mensaje, la clave y parámetros de configuración.

2.4 Casos de uso

Valoradas todas las posibles tecnologías presentadas anteriormente, se considerará los siguientes casos de uso para implementar en este proyecto.

- ***Medición y predicción de fallos en sistema de alimentación:*** Se realizarán mediciones directas: voltaje de batería e indirectas: temperatura y humedad. Se considera obviar el voltaje del suministro y el estado del interruptor diferencial, ya que mediante el SOC de la batería y una procesado de software se puede determinar las fases de carga y de descarga en todo momento, indicando si hay fallo de suministro. Tampoco se considera la interacción con él, ya que diseñar un sistema mecánico para rearmarlo es muy dificultoso de implementar y con una fiabilidad baja.

- ***Adquisición de datos complementarios para dar valor añadido:*** Como se ha comentado, se considerará adquirir los siguientes datos de interés: sonido mediante micrófono, posicionamiento mediante GPS y calidad del aire mediante un sensor.
- ***Gestión y monitorización de los distintos elementos de la célula:*** Se monitorizará todos los elementos conectados a la red de cada célula, que el mismo dispositivo de monitoreo sea capaz de detectar todos los elementos e identificarlos. Posteriormente se podrá vincular su página de configuración propia del fabricante y la posibilidad de poner avisos configurables si alguno de ellos cae. La principal idea es unificar la gestión de todos los elementos de una célula en una sola entidad.
- ***Interfaz de usuario para gestionar todos los anteriores casos de uso***: Se plantea diseñar una interfaz intuitiva y accesible desde cualquier punto y cualquier dispositivo. Con esta interfaz de podrá acceder a toda la información y configuraciones. Debe adaptarse perfectamente a cualquier tipo de usuario. Inicialmente está pensada para los técnicos de la empresa de mantenimiento.

3 Prueba de Concepto

3.1 Introducción

Para estudiar el potencial, viabilidad del proyecto y los casos de uso planteados anteriormente, se ha realizado una prueba de concepto. Dicha prueba consiste en un entorno completo E2E (End to End), donde básicamente se ha construido un sistema desde la obtención de datos hasta el usuario final, pero con la diferencia de ubicarse en un entorno controlado y con una infraestructura limitada a un solo nodo.

Para su ejecución, se ha trabajado de forma paralela en sus distintos componentes, que básicamente son 3: el nodo, el software Back-End que se encuentra en la célula, y finalmente el software Front-End albergado en un servidor externo de Hosting.

3.2 Antecedentes

En primera instancia se construyó una célula basada únicamente en un Arduino que él mismo gestionaba los datos de los sensores y los ponía disponibles a internet mediante su propia Shield de Ethernet. Posteriormente, se realizaba un procesamiento de datos de forma centralizada en el Cloud, pero como se comentó en el apartado anterior, esta solución resultaba en un sistema con posibilidades de interacción con la el nodo muy limitadas, con poca potencia de cálculo y poca escalabilidad.

En la primera reunión con el supervisor del trabajo de desestimó esta opción dando lugar a otra arquitectura mucho mas atractiva; el Fog Computing. Este concepto se basa en distribuir la capacidad de procesamiento a todas la células y minimizar la centralización. Para lograr esto se consideró dotar las células de la placa computadora de bajo coste Raspberry Pi.

3.3 Montaje de la célula

3.3.1 Hardware

Para la llevar a cabo esta prueba de concepto, se ha construido una único nodo de forma sencilla, integrando todos los elementos propuestos para los distintos casos de uso que se recogen en el apartado Tecnologías Actuales Aplicables. En este sentido se ha dotado a la célula de sensores de temperatura, humedad, calidad del aire, posicionamiento mediante GPS, voltaje de la batería y, finalmente, micrófono.

Para simplificar el montaje de la célula en cada punto, se ha considerado alimentar los equipos electrónicos mediante línea de entrada también usada para la adquisición del voltaje de la batería, pero en este caso con un circuito adicional estabilizador PWM. Además, se han integrado todo los componentes dentro de una caja IP 66 para un buen grado de protección.

En el Anexo, se puede observar el circuito principal del nodo, ubicado en la Shield PCB de Arduino. Este circuito tiene dos grandes finalidades: por una parte se pretende alimentar a los sensores, placa Arduino y placa Raspberry Pi a través de la batería, por lo que el circuito protegerá contra posibles inversiones de polaridad y además estabilizará el voltaje a 5V constantes.

En este sentido, como se puede observar en el fragmento de la figura 27, la señal de entrada proveniente de la batería en primer lugar encuentra un fusible de protección, esto es debido a que el gran amperaje que ofrece la batería podría dar lugar a situaciones catastróficas en caso de cortocircuito. A continuación se encuentra un diodo, este diodo protege el circuito en caso de inversiones de polaridad.

Una vez protegido el circuito, el condensador electrolítico C1 dará suficiente estabilidad para dos propósitos: En primer lugar se obtendrán valores estables de voltaje para ser capturados a través de Arduino, esta señal será previamente reducida mediante el divisor de tensión formado por R1 y R2. Esto es debido a que los puertos analógicos de Arduino pueden cuantificar voltajes de hasta 5V. En segundo lugar se requiere este voltaje estable para poder ser convertido a 5V mediante un conversor PWM.

Finalmente, este voltaje de salida del regulador volverá a ser estabilizado con un condensador de las mismas características que el anterior y se pondrá un LED indicador de estado para montar en la carcasa.

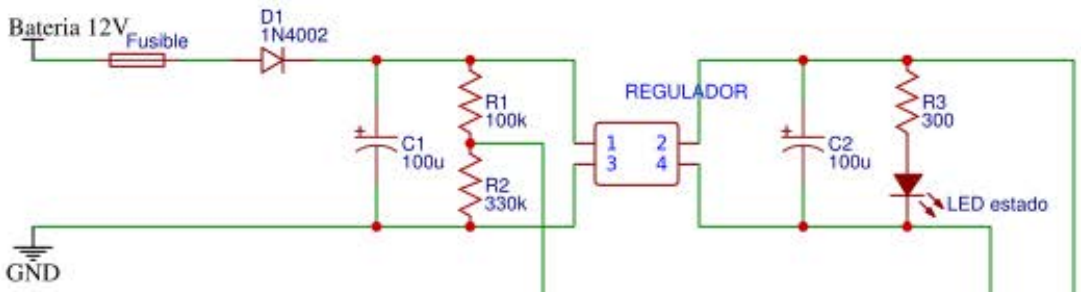


Figure 27. Circuito regulador de voltaje de la PCB Shield de Arduino – Fuente: Propia

Por otra parte, el circuito realizará la importante función de servir como red de adaptación para los sensores. En el caso de los sensores de calidad de aire y GPS únicamente servirá de placa de soporte con simples conexionados, pero en el caso del sensor de temperatura y humedad, al comunicarse mediante el protocolo I2C, se incluirán las resistencias de Pull-Up tal y como se indica en el manual de (Ada, 2018).

3.3.2 Software Back-End

3.3.2.1 Integrado en Arduino

El software Arduino se ha implementado de forma la forma mas sencilla posible; en intervalos cortos de tiempo envía la información de todos los sensores en un documento JSON. Después, mediante el protocolo Serie, Raspberry accede a él a través de una conexión USB.

3.3.2.2 Integrado en Raspberry Pi

Para implementar la mayoría de las funcionalidades Back-End en la Raspberry Pi, se ha escogido Node.js. Gracias a los siguientes módulos de Node, se ha logrado una buena integración de la mayoría de funcionalidades en un único proyecto Node.js:

Con Ngrok, se ha conseguido establecer túneles que permiten exponer a Internet múltiples conexiones hacia distintos puertos del Localhost de la Raspberry Pi, todo esto sin limitaciones externas como Firewalls o Routers.

Se han establecido dos túneles, el primero, con conexión al puerto 22, permite acceder de forma remota al SSH de la Raspberry para poder realizar tareas administrativas de forma remota. El otro, con conexión al puerto 8001, permite acceder a varios servicios del nodo, todos ellos gestionados por el microservicio Express, explicado a continuación.

El paquete Express, permite responder a llamadas http. En el caso de este proyecto se gestionan dos llamadas distintas. La primera es /data, donde Express responde con un JSON con los datos de los sensores, el ID del nodo y un Timestamp para conocer el momento exacto de los datos. La segunda llamada referente a /Icecast, simplemente direcciona hacia el puerto interno 8000, donde se encuentra el servicio de Streaming.

Finalmente, gracias al Bot del Telegram, se ha programado un Bot que permite enviar la URL del servidor de Hosting con la ID de la célula como parámetro. De esta forma desde cualquier dispositivo con Telegram se puede acceder a la célula sin conocer su dirección, ya que ésta es dinámica.

3.3.3 Software Front-End

La aplicación Front-End se ha desarrollado en Angular 6. Su funcionamiento es bastante sencillo; Se parte de una URL básica: http://saiio.000webhostapp.com/#/cell/, a partir de aquí se va concatenando en la URL las distintas ID de las células, también llamado Deep Linking. Entonces, la aplicación lo interpreta y procesa dando como resultado una página donde se puede visualizar todas las células introducidas a través de una barra selectora.

En cuanto a su estructura, la aplicación está compuesta por las 3 partes del modelo MVC:

- Modelo: Se trata de la célula y se compone de todas sus mediciones disponibles
- Vista: Es la estructura de cómo se visualiza los datos de la célula obtenidos de la célula. La aplicación permite visualizar distintas células a pesar de que actualmente solo se puede ver una en esta prueba de concepto.
- Controlador: Esta parte se encarga de realizar una llamada a cada una de las células que se le han especificado por Deep Link. Realiza llamadas asíncronas, esto significa que la aplicación empieza a mostrar datos aunque no haya recibido los datos de todas las células.

En la figura 28 se puede observar como la parte de modelo corresponde con la carpeta “model” donde se encuentra la clase cell que alberga todas las propiedades relacionadas con la célula. La parte de vista corresponde con la carpeta “modules”, encargados de representar toda la información en pantalla. Finalmente, la parte controlador corresponde con la carpeta “controller”, que contiene la clase cell.service, encargada de la gestión de la célula, esto incluye llamadas y mapeo.

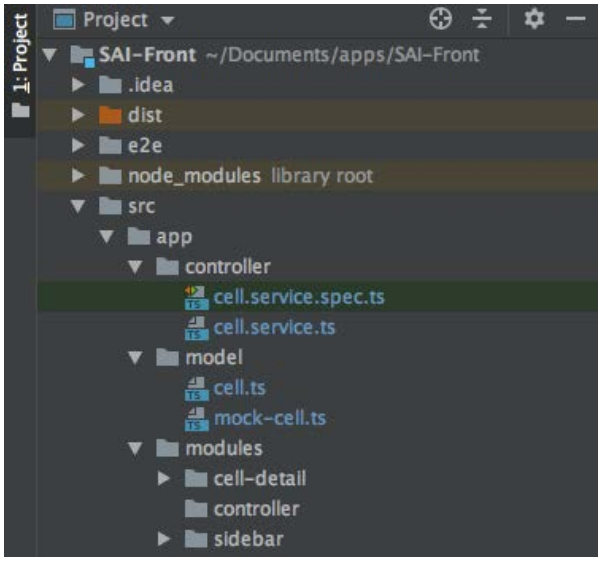


Figure 28. Estructura del proyecto según modelo MVC – Fuente: Propia

4 Desarrollo posterior a la PoC

Después de la ejecución de la prueba de concepto, se ha visto de forma bastante amplia las posibilidades que ofrece el proyecto basado en Fog Computing. En este apartado, teniendo en cuenta el alcance de este trabajo, se presentará una propuesta de diseño del prototipo final.

Se empezará detallando los cambios en el hardware, a continuación se detallaran los que pertenecen al software, tanto a nivel de Back-End como de Front-End. Además, se listará una serie de desarrollos adicionales que involucran a todos los niveles.

Finalmente, se concluirá el apartado con una descripción detallada de cómo han sido desarrollados estos cambios en el sistema.

4.1 Propuesta de diseño

4.1.1 Hardware

En cuanto a arquitectura de hardware de los nodos, será igual a la del prototipo de la prueba de concepto, se contempla alguna variación pero a nivel de detalle. En esencia contendrá los mismos sensores y microcontroladores.

4.1.2 Software Back-End

En cuanto al software de Arduino, éste será funcionalmente el mismo, ya que se ha comprobado que es totalmente estable. Respecto al software de la Raspberry Pi, éste será ampliado y reestructurado, pero manteniendo la arquitectura de microservicios basada en Node.js.

En primer lugar, se creará una base de datos con histórico de los valores de los sensores. Posteriormente, se procederá a hacer un procesado de los valores de voltaje de la batería. De esta manera se podrá calcular parámetros adicionales como determinar si durante la noche ha habido carga. Finalmente, se instalará un servicio de audio para distribuir los sonidos captados por el micrófono.

4.1.3 Software Front-End

El software Front-End esta completamente adaptado a funcionar simultáneamente con múltiples nodos, por lo que únicamente necesitará una adaptación a nivel de la capa vista para poder representar adecuadamente los valores provenientes de los nodos.

4.1.4 Otros desarrollos adicionales

Seguridad E2E: Como las células tienen información valiosa del estado de las cámaras de videovigilancia, es muy importante que no se haga un uso indebido de datos ya que pueden revelar una posible falla de estos sistemas.

Replicación de células: Una vez comprobado que el funcionamiento es el esperado para una célula, se crean mas células y se adapta todo el sistema para poder integrar varias células.

Despliegue: Proveer de células como mínimo en 3 puntos de los 6 que están instalados. Estos 3 son accesibles desde el suelo mientras que los otros 3 restantes solo son accesibles mediante grúa.

4.2 Desarrollos posteriores a la prueba de concepto

Este apartado detalla como han sido implementado los desarrollos que han sido propuestos en el apartado anterior. A continuación, se mostrará como ha sido diseñado el servicio de Streaming de voz, después las encriptación E2E y finalmente, el tratamiento de valores de voltaje mediante bases de datos.

4.2.1 Servicio de Streaming

Un Streaming es una difusión en continuada de algún contenido multimedia, es decir, el usuario final está consumiendo un contenido a la vez que se lo descarga. Los contenidos de un Streaming pueden ser o bien ya existentes, como el caso de Netflix donde el usuario está consumiendo un contenido ya grabado pero a la vez que se lo va descargando, o bien capturados a tiempo real, como la señal de un corresponsal de un telenoticias que va enviando el contenido a la vez que va siendo grabado.

En el caso de este proyecto, se necesita implementar un servicio de Streaming a partir de señales de voz digital capturada en tiempo real, por lo que se han necesitado dos componentes. Esto es debido porque, previamente a ser difundida por la red, es necesario registrar y codificar la señal de voz.

Para plataformas Linux, se ha considerado usar un sistema bastante liviano y muy recomendado para microcontroladores como Raspberry Pi llamado DarkIce. Este software es capaz de registrar la señal digital de una interfaz de audio, como un micrófono USB, y codificarla a señal de audio digital para poder ser difundida posteriormente.

A continuación, Icecast toma el papel de servidor de contenidos Streaming, es capaz de difundir tanto señales de audio como video, pero en este caso solamente se usará para señales de audio.

Para la implementación, se ha configurada un solo canal a 44100Hz de muestreo, 16 bits de cuantificación y el dispositivo de entrada de audio. Posteriormente, se ha configurado Linux para que el servicio de inicie automáticamente al iniciarse la Raspberry Pi. Finalmente, se ha configurado la forma de acceder al Streaming, estableciendo el servidor Icecast al puerto 8000 para no crear conflicto con el microservicio Express que esta establecido al puerto 8001.

Entonces, se ha tenido que modificar el código Node.js para que Express mapee las consultas localhost:8001/icecast a localhost:8000. De esta forma no es necesario mapear nuevos puertos, consecuentemente no será necesario tampoco abrir un nuevo túnel Ngrok.

Una vez implementado el servicio de Streaming, se puede probar su funcionamiento con el mismo navegador. Simplemente introduciendo la dirección del túnel Ngrok seguido de /icecast/mic, por ejemplo: http://c412291b.ngrok.io/icecast/mic, se obtiene el Streaming de la señal del micrófono USB. Dando como resultado la figura 29.

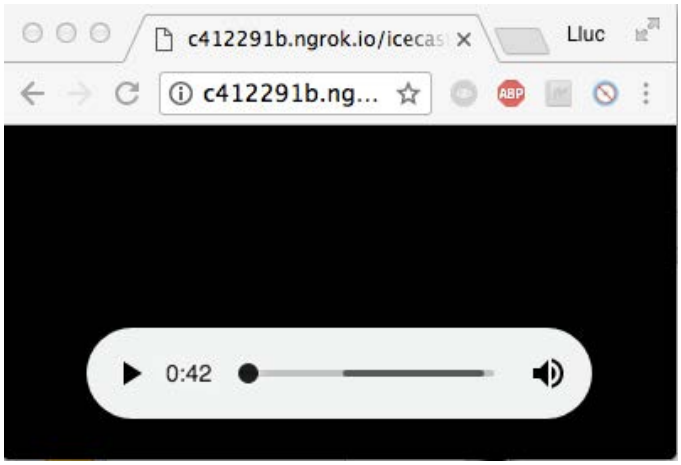


Figure 29. Streaming de la señal del micro accediendo desde Google Chrome – Fuente: Propia

4.2.2 Cifrado E2E

En el caso de este proyecto, se tratan datos completamente relacionados con la autonomía de la batería que alimenta las cámaras de videovigilancia, por lo que exponer datos acerca de cuando va a fallar el sistema de videovigilancia municipal puede suponer una gran vulnerabilidad para la seguridad de un municipio. En este sentido es de vital importancia un cifrado E2E.

Siguiendo la línea del estudio realizado en el apartado Tecnologías Actuales Aplicables, se ha decidido en la parte de Back-End cifrar el JSON completo mediante un modulo de Node.js llamado node-cryptojs-aes. Entonces, si se intenta acceder a los datos del JSON publicado a internet se obtendrá un documento totalmente inteligible sin ningún tipo de estructura tal y como se muestra en la figura 30.

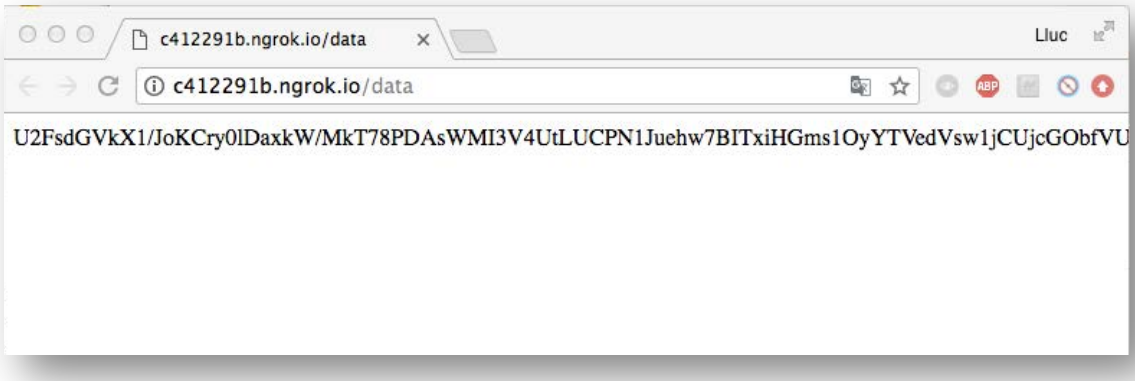


Figure 30. JSON totalmente codificado tratando de acceder desde Google Chrome – Fuente: Propia

Finalmente, en la parte Front-End, se ha implementado un descifrador que, conociendo la clave, será capaz de volver a recuperar el JSON original y a continuación procesarlo y presentarlo exactamente de la misma forma que lo hacia antes de ser implementado este cifrado E2E.

4.2.3 Tratamiento de valores de voltaje mediante bases de datos

Para cumplir con el objetivo de predicción, detección e identificación de fallos en el sistema de alimentación, se ha considerado realizar un tratamiento de los valores de voltaje en vez de limitarse a representarlos en tiempo real igual que las otras magnitudes. Este tratamiento se divide en dos partes.

Primeramente, se desea crear un histórico de datos, de esta forma se podrá determinar directamente si durante la noche ha habido carga, así será fácil identificar si el problema se localiza antes o después de la batería. Adicionalmente, también se podrá observar si la batería funciona en ciclos de carga profunda (Deep-Cycle), es decir, por debajo del 50% del SOC hecho que podría disminuir considerablemente su vida útil.

Consecuentemente, la parte de Front-End se deberá adaptar para poder mostrar los valores de este histórico. La forma más sencilla e intuitiva es mediante un gráfico que relacione SOC-temporal. Para su implementación se ha considerado la popular dependencia HighCharts, muy utilizada y con compatibilidad para la mayoría de frameworks orientados a aplicaciones web.

En segundo lugar, se procederá a implementar un sistema de avisos. Se utilizará el módulo del bot de Telegram, ya que resulta un sistema de broadcast muy eficaz y fácil de implementar en el usuario final, sin tener que configurar costosas aplicaciones.

Finalmente, se adecuarán estas nuevas implementaciones siguiendo la línea anterior, esto es, con un cifrado E2E que impida que los datos terminen a manos de terceras personas, hecho que podría comprometer la seguridad del sistema.

4.2.4 Histórico de valores del estado de carga

Para la implementación de la base de datos de un histórico de valores del SOC, se ha utilizado SQLite. Tal y como se ha explicado en el apartado de Tecnologías Actuales Aplicables, se ha elegido SQLite frente otras tecnologías por ser una base de datos ligera y fácil de poner en marcha. Además, como se verá más adelante, su implementación ha sido bastante sencilla y no compromete en absoluto la estabilidad del sistema. La implementación, integrada en su totalidad en el programa Node.js, ha consistido en tres partes. A continuación de analizará detalladamente como ha sido llevada a cabo.

Primeramente, se ha construido un método que, a partir del voltaje de la batería, devuelve el estado de carga. Según (EnergyMatters, 2018), la tabla de la figura 31 es una buena aproximación del SOC en función del voltaje para tres tipos de baterías basadas en la tecnología plomo-ácido.

Estado de carga	Batería de plomo-acido estándar	Batería de Gel	Batería AGM
100%	12.7V	12.85V	12.8V
75%	12.4V	12.65V	12.6V
50%	12.2V	12.35V	12.3V

25%	12V	12V	12V
0%	11.8V	11.8V	11.8V

Figure 31. Tabla con los valores de voltaje típicos para varios estados de carga de distintas tecnologías de batería

Como se puede apreciar en la figura 32, el SOC varia ligeramente en función del tipo de batería. Además, la ecuación resultante es una de primer grado definida a trozos.

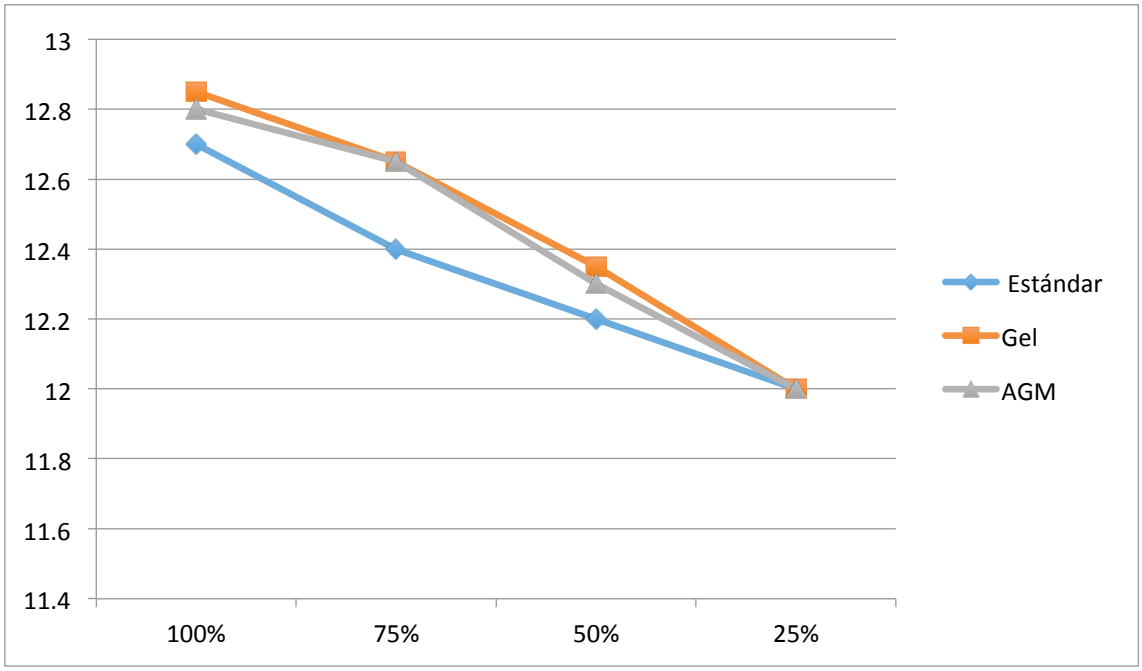


Figure 32. Representación gráfica de la tabla anterior

En segundo lugar, una vez ya obtenido el estado de carga, se modifica el JSON con los datos de los sensores para incluir esta nueva variable y a continuación se almacena en la base de datos. Este almacenamiento no se realiza a medida que se va capturando nuevos valores, sino que se lleva a cabo de forma periódica para no sobrecargar la base de datos con datos irrelevantes. Inicialmente se ha considerado un período de cinco minuto, que dará una buena resolución en el gráfico.

A continuación, para poder hacer disponible sus datos a través de Internet, se tomará la misma metodología que para publicar los datos de los sensores: se creará un JSON con valores de base de datos comprendidos desde el momento que se ha especificado en la petición hasta la última muestra. Este documento, también se codificará mediante la encriptación AES y, concretamente, con la misma contraseña.

Finalmente, se procede a adaptar la aplicación Angular para incluir un gráfico de los últimos 30 días del estado de carga. En primer lugar, se ha modificado el anterior servicio que realizaba únicamente una llamada asíncrona para obtener el JSON con los datos de los sensores, ahora hace dos llamadas secuenciales pero conjuntamente asíncronas: primeramente se recogen los datos de los sensores y, una vez obtenidos, ya se encuentran disponibles mientras se están obteniendo los datos históricos, ya que se trata de una llamada un poco más lenta.

Una vez obtenidos los datos, se actualizará el gráfico HighChart con los nuevos datos, él mismo incorpora funciones de zoom, scrolling, leyenda, y visión en detalle de las distintas muestras. Además, igual que el resto de la aplicación, es completamente adaptable a distintos tipos de resolución de pantalla, hecho que los hace multiplataforma desde dispositivos móviles hasta dashboards de gran resolución.

4.2.5 Sistema de avisos

En este caso se tendrán dos consideraciones; En primer lugar, se procederá a avisar mediante un mensaje Telegram en caso de nivel bajo de batería. Se ha establecido un 30% como umbral. Adicionalmente, para tratar de alargar la vida útil de las baterías, se avisará también en caso de entrar en ciclo profundo, es decir, al bajar del 50% de carga.

Al tratarse de un bot de Telegram, se puede incluir en cualquier conversación de grupo como cualquier persona física, entonces el bot, que previamente tendrá configurado su emplazamiento, notificará a todos los usuarios de la conversación de porcentajes bajos de batería indicando cual es el la localización.

5 Diseño del sistema

En este apartado, se presenta el diseño del sistema final una vez realizados los desarrollos posteriores a la prueba de concepto. En primer lugar se analizarán los diagramas del sistema para verlo en todo detalle y desde distintas perspectivas. Finalmente, se realizará una prueba conceptual de estrés para determinar cuales son lo límites del servicio, de esta forma se podrá demostrar que el sistema funcionará bajo las condiciones de trabajo.

5.1 Diagramas del sistema

A continuación se presentan cuatro tipos de diagramas para representar todo el sistema con distintos enfoques. En primer lugar se representará el diagrama físico, en segundo lugar se comentará el diagrama lógico, siendo éste una versión más simplificada del anterior y mas orientada al punto de vista de Internet. A continuación se analizará el diagrama funcional, donde se muestra el funcionamiento del sistema para los distintos tipos de usuarios que acceden al sistema. Finalmente se concluirá el apartado analizando el diagrama tecnológico, donde se verán reflejadas todas las tecnologías usadas en cada uno de los elementos.

5.1.1 Diagrama Físico

La representación física se encuentra dividida en dos partes; En primer lugar, se mostrará el nodo de forma individual para poder ser representado en mas detalle en la figura 33. Finalmente, se representará el sistema completo en la figura 34.

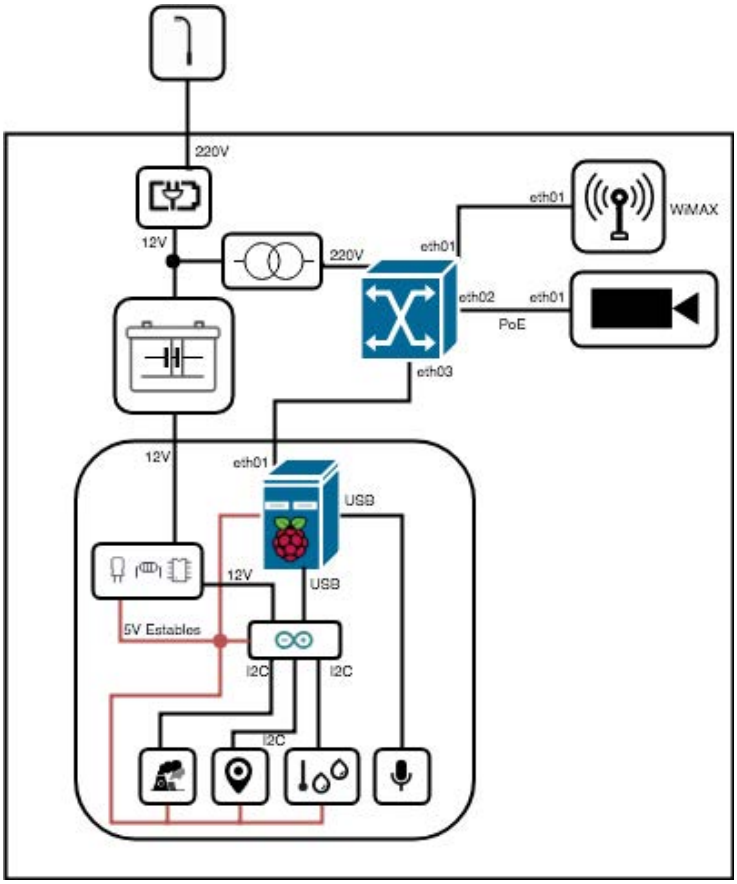


Figure 33. Diagrama físico en detalle del sistema – Fuente: Propia

El nodo está dividido básicamente en dos partes. Primeramente se dispone del sistema del que se disponía previamente, compuesto por el cargador de batería, la batería y el inversor, que componen el sistema de alimentación ininterrumpida. Este sistema da suministro a un Switch PoE que alimenta a la antena WiMAX y a las cámaras de videovigilancia.

En segundo lugar, se encuentra la unidad de procesamiento, que es la que se ha implementado en este proyecto. Esta unidad está compuesta primeramente por un circuito adaptador conectado directamente a la batería del sistema de alimentación. La función más importante de este circuito de adaptación es proteger a todos los subsistemas de subidas de tensión, corrientes parásitas y cambios de polaridad. Además, también tiene la misión de subministrar 5V estables a todos los subsistemas. Como último punto, también se encarga de bajar el voltaje de la batería para poder ser medido por Arduino.

Después del circuito de adaptación, se encuentra la Raspberry Pi, que a su vez está también conectada con el Switch PoE mediante conexión Ethernet y con el Arduino y el dispositivo de captación de sonido mediante conexión USB. Finalmente, el Arduino se encuentra conectado con los sensores a través de conexiones I2C.

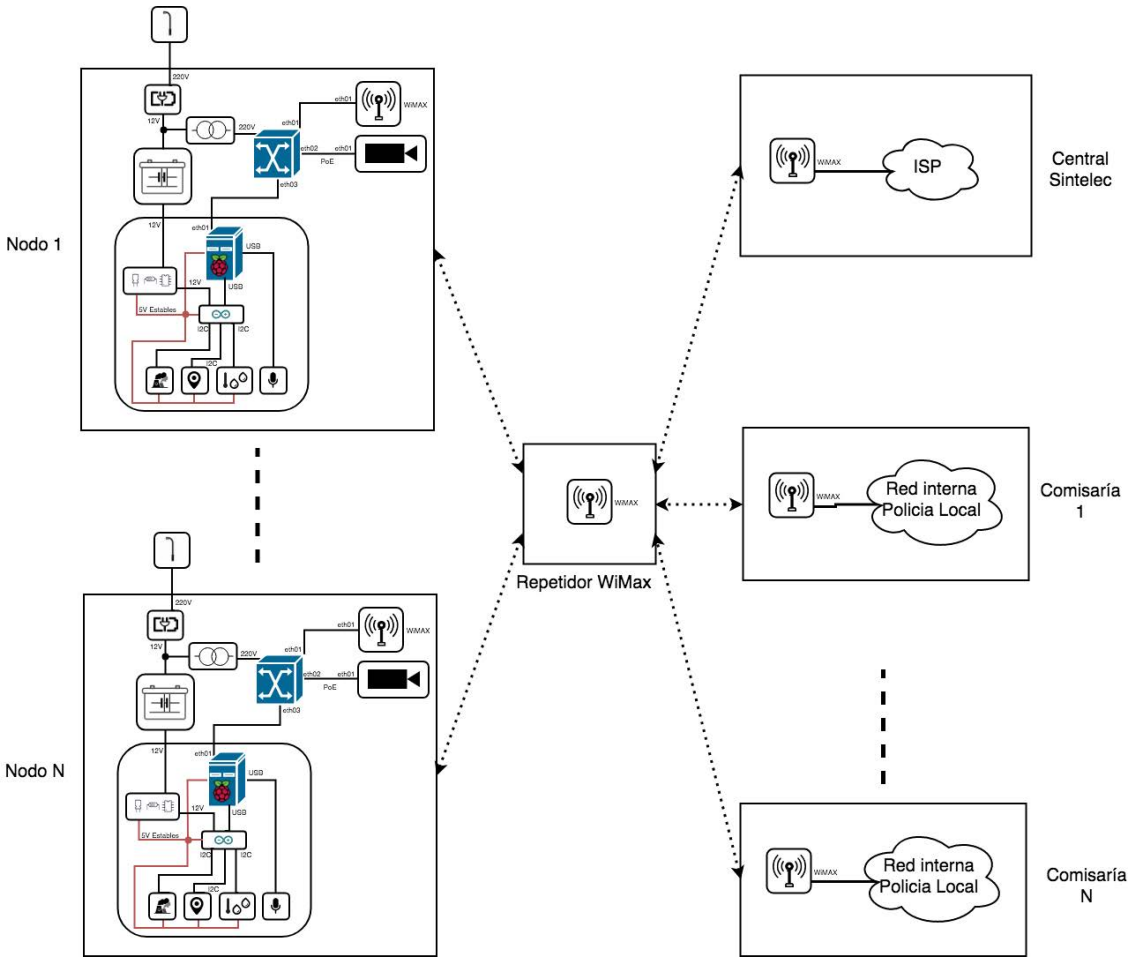


Figure 34. Diagrama físico completo del sistema – Fuente: Propia

En referencia a el diagrama completo, es importante mencionar que todos los nodos, las comisarias y el punto de suministro de internet se encuentran interconectados mediante

conexiones WiMAX reservada para Sintelec. Cada uno de los radioenlaces es de 30 Mbps y se reparten las IP dinámicamente mediante el protocolo DHCP.

5.1.2 Diagrama Lógico

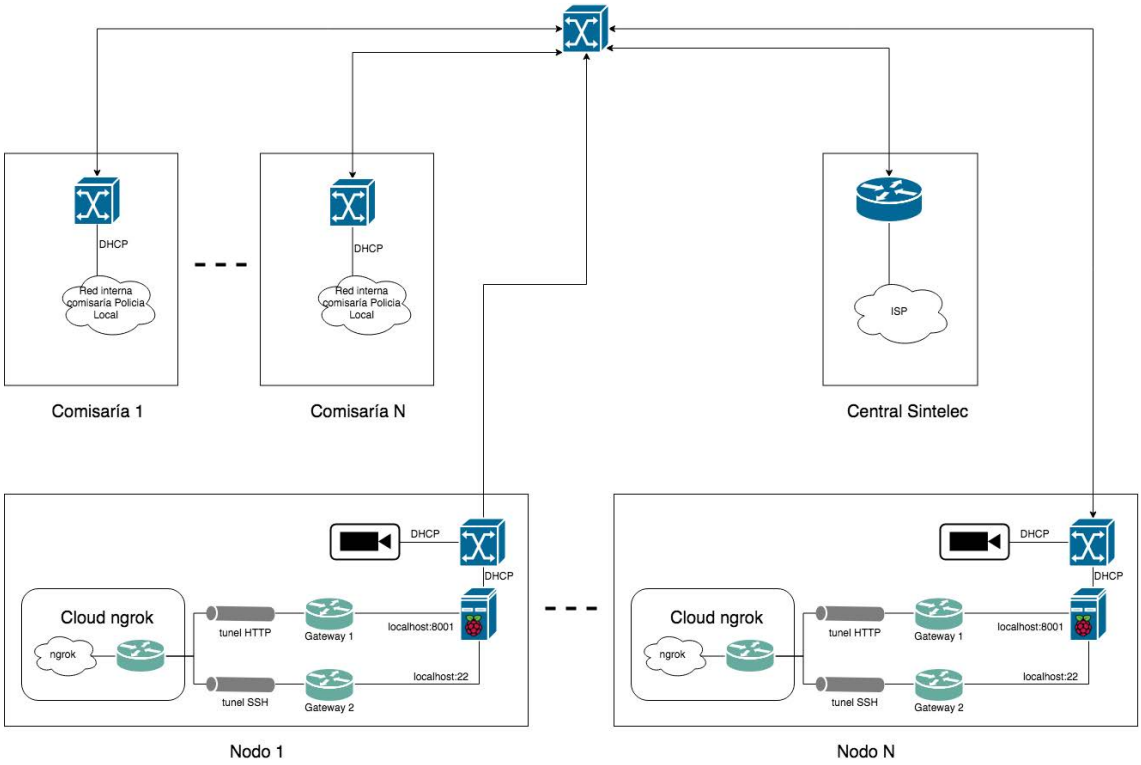


Figure 35. Diagrama lógico completo del sistema – Fuente: Propia

En la figura 35 se muestra el diagrama lógico de la red interna sobre la que se encuentran los nodos, está compuesta por los distintos nodos, comisarías y la central de Sintelec.

En relación a los nodos, se puede ver que cada Raspberry Pi está conectado a un Switch que da conectividad al exterior y a el servicio de videovigilancia. Además, dispone de dos VPN de Ngrok, una para el puerto 8001 destinada al acceso de microservicios y otra pera el puerto 22 para tareas de mantenimiento remoto.

Por otra parte, las comisarías usan la misma red tanto para acceder a internet como para conectarse a el servicio de videovigilancia, por lo que no necesitan salir fuera de la red para acceder a la información de los nodos.

Finalmente, la propia central de Sintelec, que se encuentra directamente conectada con la red interna proporciona conexión a internet mediante un ISP.

5.1.3 Diagrama Funcional

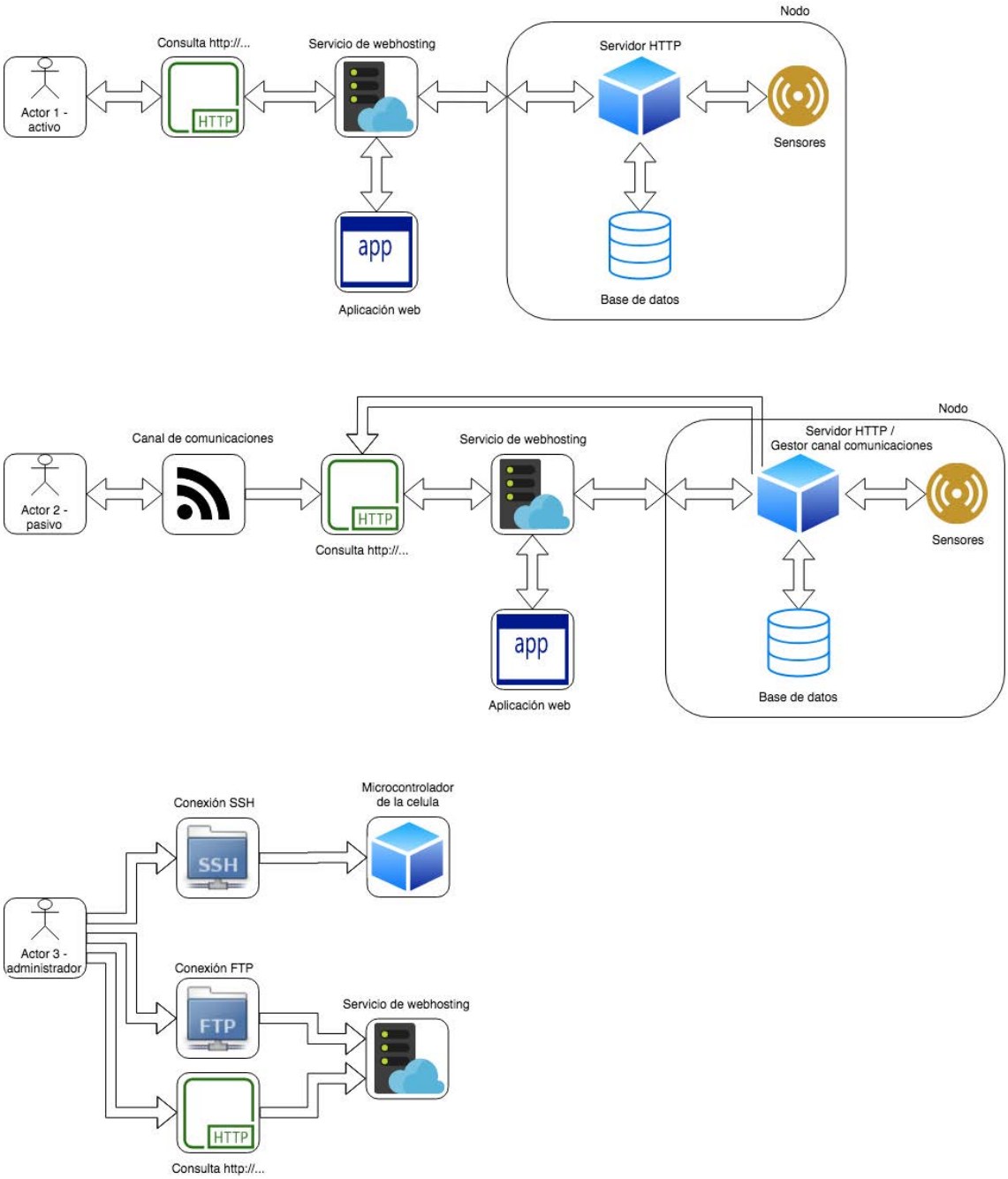


Figure 36.Diagrama funcional completo del sistema – Fuente: Propia

Para representar el diagrama funcional del sistema de la figura 36, se ha considerado tres actores que representan las distintas casuísticas. Los dos primeros son usuarios del servicio pero en distintas situaciones mientras que el último actor representa a un administrador que desea realizar tareas de remotas.

El primer actor se trata de un usuario activo, es decir, alguien que desee consultar por ejemplo el estado de la batería de uno de los nodos. Se parte de un usuario que conoce el ID del nodo que, en primer lugar, realiza una llamada http mediante el navegador de su dispositivo. Después, el servicio de Hosting responderá con los datos de la aplicación web que alberga. Esta aplicación se cargará en el navegador del dispositivo del usuario y, posteriormente, realizará

una llamada al nodo. El nodo, que dispondrá de la información que previamente ha capturado y almacenado en su base de datos, responderá con un procesado de esta información. Finalmente, la misma aplicación web del dispositivo del usuario interpretará estos datos y serán representados de una forma adecuada.

Por otra parte, el segundo actor representa a un usuario pasivo. Esto es, por ejemplo el sistema detecta que el nivel de batería es bajo y notifica al usuario mediante su canal de comunicaciones. El usuario, al ver el aviso decide consultar el estado del nodo y accede el vinculo del canal de comunicaciones. A partir de este momento toma el mismo papel que el primer actor (activo) y el flujo es el mismo.

Finalmente, existe un tercer actor que no representa a un usuario normal del servicio, representa un administrador del sistema que, por motivos de localización respecto al nodo accederá de forma remota. Este usuario dependiendo de sus labores se podrá conectar mediante tres tipos distintos de conexión. La primera es una conexión segura SSH directamente a la Raspberry Pi del nodo en cuestión. En segundo lugar, se puede conectar al servicio de Hosting mediante una llamada http para realizar tareas de mantenimiento del sitio web. Finalmente, también se puede conectar al servidor de Hosting mediante el protocolo FTP, con el fin de publicar la aplicación web.

5.1.4 Diagrama Tecnológico

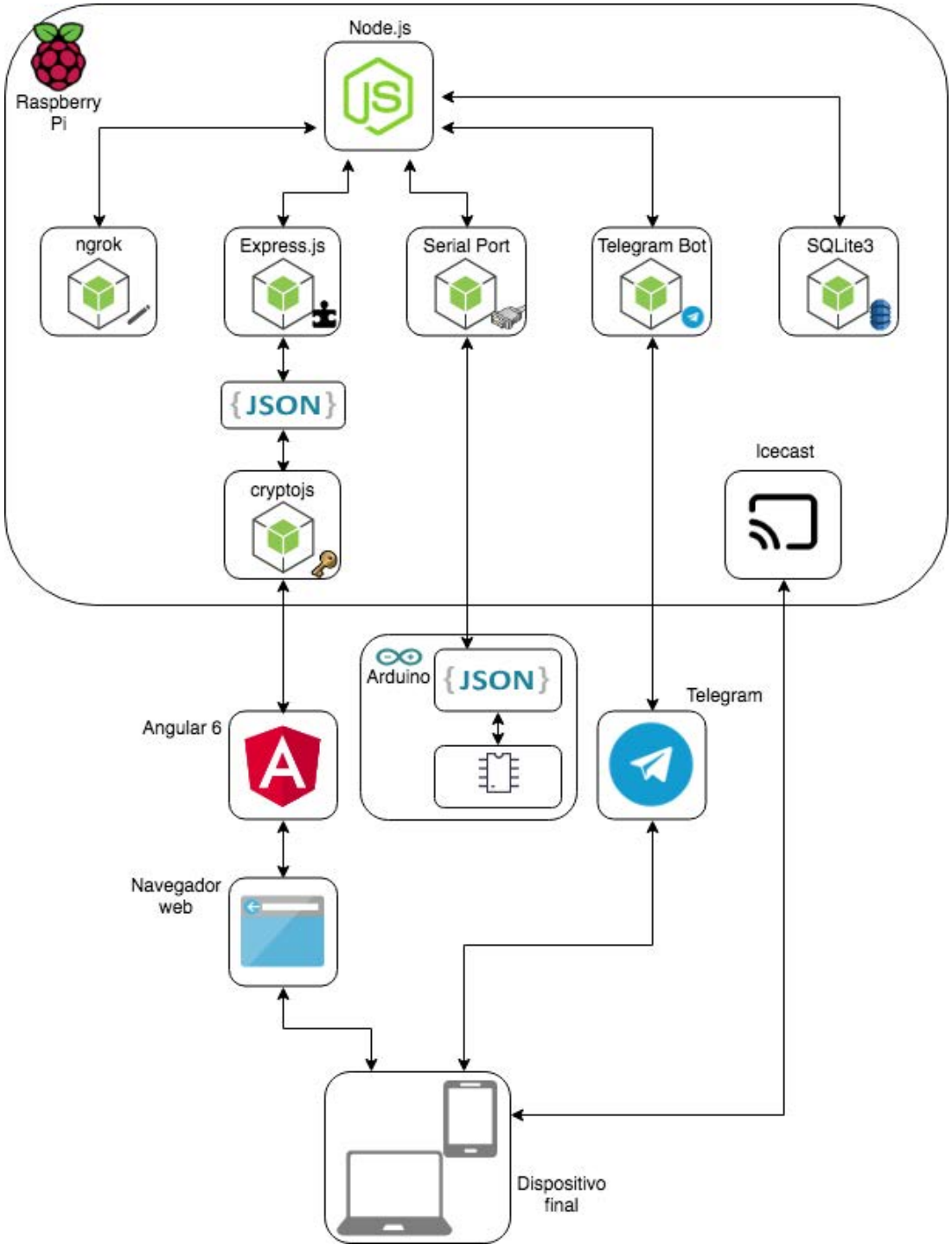


Figure 37. Diagrama tecnológico completo del sistema – Fuente: Propia

En este último diagrama de la figura 37, se pretende relacionar todas las tecnologías que han sido elegidas para la implementación del sistema.

En primer lugar, se cuenta con un microcontrolador basado en Raspberry Pi. Este computador alberga dos aplicaciones; Icecast para difundir un Streaming de la señal de audio del micrófono y Node.js para realizar múltiples funciones ya que cuenta con varios módulos. El primer módulo de Node.js es Ngrok, permitiendo establecer las dos VPN que han sido mostradas en el

diagrama lógico, además, existe el módulo express.js, que permite acceder a los datos de los sensores a través de un fichero JSON.

Para poder comunicarse con Arduino se cuenta también con el módulo Serial Port, que permite acceder al JSON generado a partir de los datos de los sensores. Por otra parte, se dispone del módulo del Bot del Telegram, que permite enviar mensajes al canal de comunicaciones Telegram. Adicionalmente, también se dispone del módulo SQLite3, que sirve como base de datos para disponer de un histórico de datos de determinados sensores.

Finalmente, para añadir una capa de seguridad a los datos de los sensores, se hace uso de un último módulo nombrado Cryptojs-aes que básicamente cifra todo el documento JSON con tecnología de Encriptación AES. Entonces, gracias a una aplicación Angular, se puede descifrar y visualizar de forma ordenada y clara los contenidos proporcionados por cada uno de los nodos, asegurando una encriptación E2E desde cada uno de los nodos hasta el usuario final.

5.2 Prueba de estrés

Las pruebas de rendimiento se realizan para determinar la estabilidad de toda la infraestructura ante situaciones límite, como podría ser ante un número alto de usuarios que quieran acceder a él, o ante un crecimiento de nodos. Los resultados de esta prueba permitirán conocer la capacidad del sistema en cuanto a escalabilidad.

Dentro del concepto de escalabilidad, existen dos tipos que son convenientes definir; La escalabilidad vertical consiste en añadir nuevas características a los nodos del sistema de tal forma que éste mejora. En cambio, la escalabilidad horizontal consiste en añadir más nodos, de forma que la infraestructura crece.

Para realizar la prueba de estrés del proyecto, se ha realizado un estudio teórico a partir de las especificaciones reales de los distintos sistemas que han sido involucrados en la prueba de concepto. Este estudio está dividido en: la prueba de estrés de los nodos y la prueba de estrés de la aplicación web.

5.2.1 Nodos

Todos los puntos de la infraestructura están dotados de conexiones WiMAX con una tasa máxima de transferencia de 30 Mbps, teniendo 5 Mbps ocupados por cada uno de los Streamings de videovigilancia. Los puntos están dotados con entre una y dos cámaras de videovigilancia, por lo que se dispone como mínimo de 20 Mbps para el tráfico de los nodos.

Los nodos están basados en Raspberry Pi, esto implica una limitación de rendimiento en cuanto a hardware a pesar de estar implementados con soluciones muy optimizadas. Para probar su rendimiento se ha procedido a obtener el tiempo de acceso de los dos documentos JSON: el relacionado con los datos de los sensores, y el que contiene el histórico de 30 días con datos de la batería. Esto se ha probado en el caso más desfavorable, es decir, con una situación de máximo uso de sus servicios, esto es, con el sistema de Streaming de voz activado.

La medición ha sido efectuada con un servicio externo (https://apitester.com/), dando como resultado 145ms para el JSON de datos de los sensores y 889ms para el histórico, es decir un

total de 1.034 s asumiendo que las llamadas se realizan en paralelo. A continuación se muestran los resultados de los test, representados en las figuras 38 y 49.

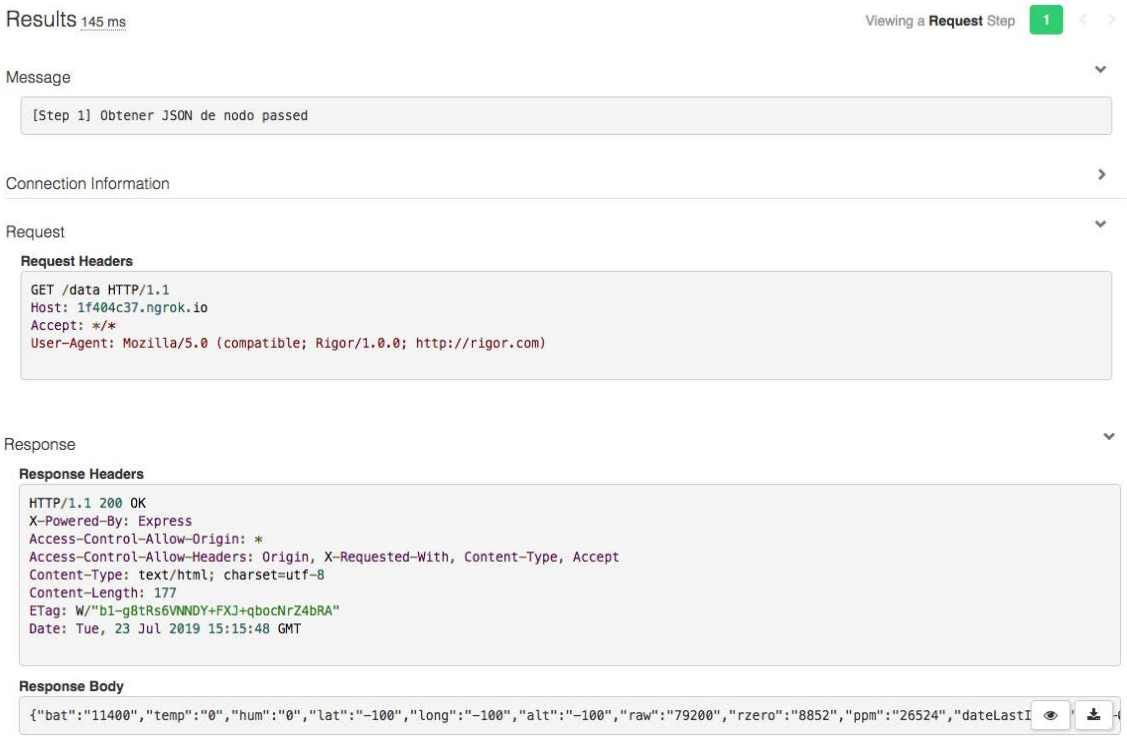


Figure 38. Prueba de estrés de la llamada a los datos de los sensores – Fuente: Propia

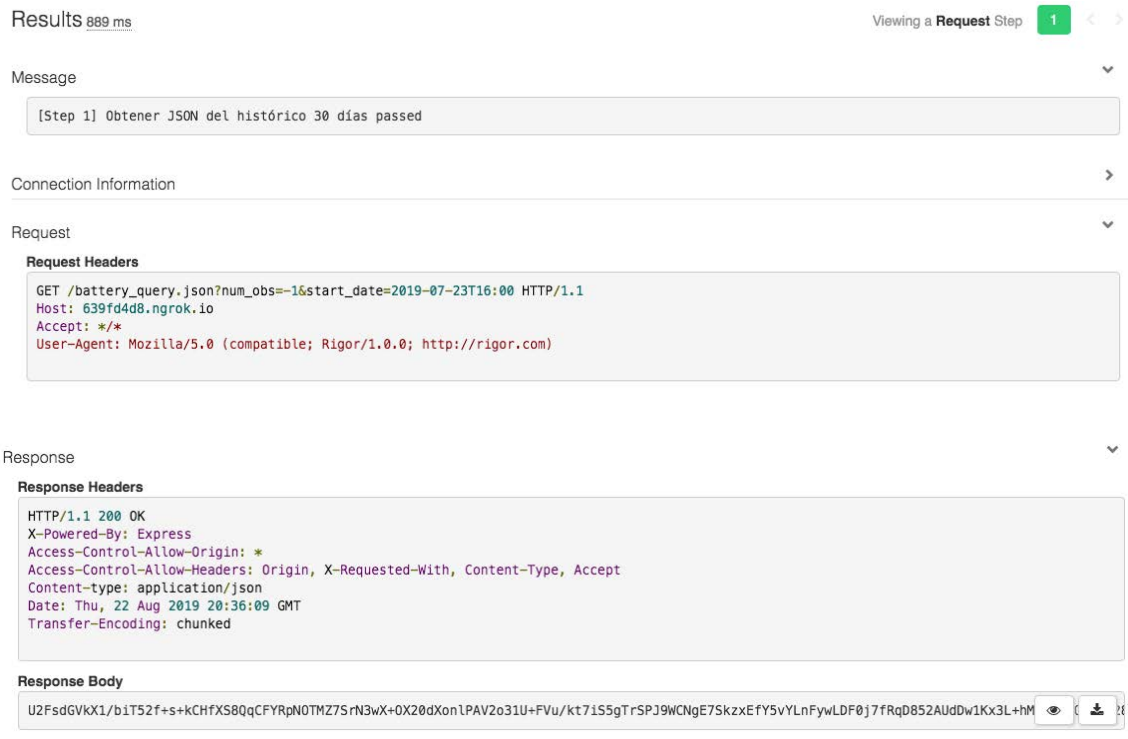


Figure 39. Prueba de estrés de la llamada a los datos históricos – Fuente: Propia

A partir del tiempo de acceso de los JSON se puede calcular el nombre máximo de peticiones por segundo, que es 0.97, equivalente a 58 peticiones por minuto. Por otra parte, se ha obtenido la cifra de 469 bytes para el documento JSON de los sensores y 160KB para los datos históricos de la batería y la tasa de 92 Kbps para el Streaming de audio, cifras que se puede considerar despreciable con los aproximadamente 20 Mbps que ofrece la conexión.

Estas cifras calculadas anteriormente no llegan a la limitación que presenta el servicio gratuito de Ngrok, que cuenta con un máximo de 40 conexiones por minuto. Por lo tanto las capacidades de Ngrok son las que limitan el rendimiento máximo de las células.

Finalmente, suponiendo que una célula realiza una notificación difundida de batería baja mediante el canal de comunicaciones, ésta podría atender un máximo de 40 usuarios simultáneamente.

5.2.2 Aplicación web

Al acceder a el servicio web, el usuario primeramente se tiene que conectar y descargar la aplicación desde el servicio de Hosting y a continuación ésta correrá completamente desde el dispositivo final, haciendo uso de sus recursos y su conexión.

Por otra parte, el servicio de Hosting cuanta con sus limitaciones de rendimiento. Al tratarse de la modalidad de servicio totalmente gratuita, las capacidades son de 10000 MB al mes para el ancho de banda y de 1000 MB para el almacenamiento de los ficheros de la aplicación web. Teniendo en cuenta que la aplicación de la prueba de concepto pesa 567 KB, los 1000 MB no van a ser una limitación para este proyecto.

Por otra parte, la carga de la página sin caché en el dispositivo final son aproximadamente 200 KB, entonces considerando que el servicio tiene un limite de 10000 MB mensuales, implica un máximo de 50000 consultas mensuales, o lo que es lo mismo, 1666 consultas diarias, una cifra muy alta que raramente será alcanzada teniendo en cuenta que su uso se encuentra orientado a los administradores.

6 Despliegue

En este apartado, se detalla todo el proceso de montaje del sistema sobre los puntos donde se encuentran los existentes sistemas de alimentación ininterrumpida. Se empezará por la prueba unitaria de un único nodo, se continuará con la replicación de los nodos y, finalmente, se concluirá con el despliegue total del sistema. En todo momento se analizará la problemática que ha ido surgiendo así como la forma en la que han sido resueltos todos los incidentes.

6.1 Prueba unitaria de estabilidad del sistema

Antes de desplegar un sistema, es muy importante realizar un proceso de QA (Quality Assurance). Este tipo de pruebas son indispensables para minimizar lo máximo posible la probabilidad de fallos de diseño, tanto en Software como en Hardware. En este punto se procede a dejar en continuo funcionamiento el dispositivo prototipo para observar como se comporta.

En la figura 40, se puede observar el prototipo en pruebas de estabilidad, está alimentado con una fuente de corriente continua de 12V para simular la alimentación mediante una batería y, por otra parte, está conectado a internet mediante conexión Ethernet a un Router con el firewall activado. De esta forma se ha probado el sistema en un entorno de conexionado similar al que se encontrará en el emplazamiento final.

Figure 40. Célula en entorno de pruebas – Fuente: Propia

El sistema ha estado montado en estas condiciones durante 60 días y al principio presentó un pequeño problema de inestabilidad por parte del hardware; Resulta que en voltajes inferiores a 11V, el sensor de temperatura y humedad AM2320 dejaba de funcionar, y en voltajes inferiores a 7V la Raspberry también dejaba de funcionar. El problema no debería suponer ningún problema ya que la batería tiene un SOC del 0% por valores inferiores a 11.8V. Pero de todas formas se resolvió el problema colocando resistencias de Pull-Up en las señales digitales del sensor AM2320 y, además, se colocó un condensador de 1000 uF a la salida del regulador de voltaje, de esta forma, tanto el sensor como Arduino funcionaban a partir de un voltaje de entrada al sistema de 5V.

6.2 Instalación en ubicación final

Una vez concluido el desarrollo hardware y software y asegurando estabilidad del sistema, se procede a instalar una célula en el emplazamiento final. Esta primera parte del despliegue ha consistido en instalar primeramente un nodo, comprobar que es estable y que el hardware cumpla completamente con los requerimientos para posteriormente replicar los nodos y desplegarlos. Es de vital importancia asegurar estos dos puntos, ya que cualquier actualización que no sea de software implicará un desplazamiento.

6.2.1 Instalación de un punto

La primera instalación se ha realizado en un punto de Sant Vicenç de Montalt, como se puede ver en la siguiente figura, la instalación se ha realizado de la forma menos intrusiva posible. Únicamente se ha tenido que pinchar el conexionado de la batería e instalar la célula en un lugar estable mediante bridas, finalmente se ha procedido a conectar la célula a la red mediante un cable Ethernet. Por último, se comprueba in situ que el dispositivo se haya conectado a Ngrok y se da la instalación por concluida.

En la figura 41 se puede observar cual es el resultado del montaje de la célula dentro del armario, quedando perfectamente estable e integrado en el conjunto.

Figure 41. Célula instalada dentro del armario – Fuente: Propia

Analizando los resultados, todas las mediciones funcionan a excepción del GPS; Al tratarse de un armario de obra, se bloquea la señal GPS en el interior y, en consecuencia, no se puede obtener un posicionamiento.

Por otra parte, se ha observado que el voltaje a lo lago del día va desde los 12.78V a primera hora de la mañana, hasta los 12.5V a finales del día. Esto es superior a los 12.4V que se considera el 100% del SOC y, consecuentemente, el estado de carga a lo largo del día resulta ser del 100%. Si bien se probó el funcionamiento en un entorno controlado con una batería estándar de plomo-ácido que pertenecía a un vehículo, ahora se ha observado este comportamiento que sale de lo previsto. Esto puede ser debido a dos opciones; o bien la tabla que relaciona voltaje con SOC para baterías plomo-ácido no aplica a esta batería al tratarse de una de ciclo profundo, o bien la capacidad de la batería esta sobredimensionada y no se descarga lo suficiente para ser percibido en 12 horas.

En la figura 42, se puede ver que desde el 3 de septiembre de 2019 (día del montaje) hasta el momento de visualización de el nodo, es decir, el 10 de septiembre de 2019, no ha habido ninguna interrupción. Además, como se ha comentado anteriormente, el valor de la batería es del 100% constante. Por otra parte, el resto de magnitudes se puede comprobar que son correctos a excepción del GPS, que como se indicó no recibe señal.

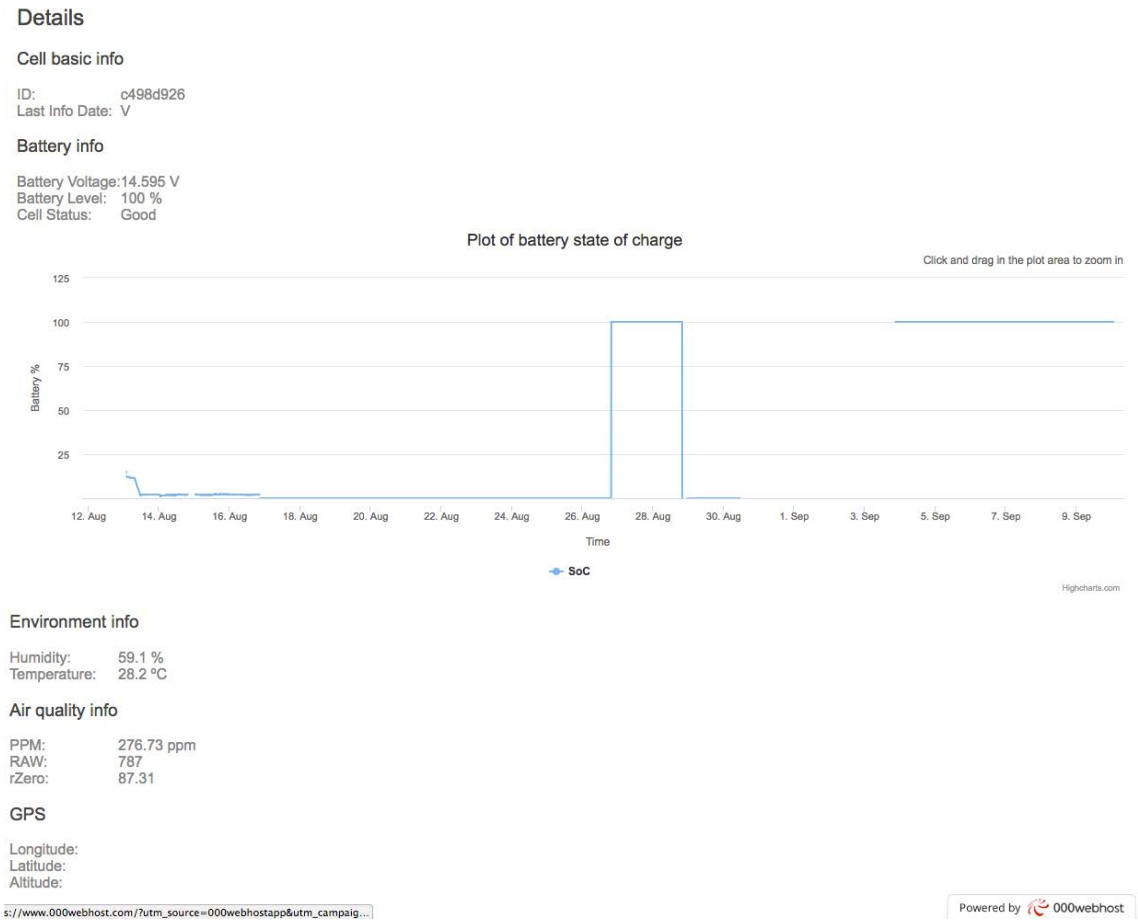


Figure 42. Página web Front-End del nodo en emplazamiento final

En referencia al sensor de temperatura, siempre marca voltajes por encima de la temperatura exterior, pudiendo llegar a indicar temperaturas de hasta 40ºC. Esto es debido a que se encuentra en un armario sin ventilación y totalmente expuesto al sol, por lo que las temperaturas quedan falseadas. De todas formas, aunque no se pueda usar para datos meteorológicos, si que puede ser de ayuda para conocer la reducción de vida útil debido a la temperatura.

Finalmente, se ha valorado muy positivamente la estabilidad; Llevando una semana, no se ha tenido que realizar ninguna tarea administrativa desde su instalación. Esto es gracias al proceso de QA, que ha permitido la instalación del nodo con una garantía de estabilidad.

6.2.2 Montaje de los demás nodos

Se montaron dos nodos adicionales: uno en Cabrera de Mar y el otro en Sant Vicenç de Montalt. La construcción de las células fue exactamente idéntica a la primera, pero como se trataba de otros nodos, se crearon otros bots de Telegram, otros túneles VPN de Ngrok, y el código fue adaptado en consecuencia.

La instalación en los demás puntos de suministro fue idéntica que en la primera: buscando un emplazamiento estable y conectando alimentación y Ethernet. Como se puede ver en la figura 43, en este caso el armario es metálico, hecho que también provoca que no reciba señal GPS.

Figure 43. Punto de alimentación situado en la playa de Sant Vicenç de Montalt – Fuente: Propia

Una vez instalado, se observaron otros aspectos destacables; En primer lugar, no existe una buena integración de todos los bots de Telegram: en situaciones como pedir la URL de la aplicación Front-End, todos hablan a la vez mandando una URL para cada nodo, en vez de aprovechar la característica programada de concatenar distintas ID de nodo en la misma URL. Este aspecto, al tratarse de software Back-End, puede ser corregido mediante una actualización remota sin ningún tipo de intervención presencial, por lo que no implica ningún problema de despliegue.

Por otra parte, destacar también que no existe ningún nodo donde su nivel de batería baje del 50%, por lo que no se puede probar la funcionalidad de avisos en tiempo real en el emplazamiento final. De todas formas, como también es cuestión de software Back-End, un mal funcionamiento puede ser corregido remotamente.

Finalmente, se destaca la estabilidad del sistema; de la misma forma que con el primer nodo, no se ha observado ninguna interrupción o fallo. Tanto el hardware como el software es completamente estable en todo momento del día, sin presentar ninguna anomalía.

7 Presupuesto

El presupuesto está basado en la estimación de recursos humanos, el material usado y, finalmente, el software usado para el desarrollo del sistema.

7.1 Recursos humanos

Este primer apartado hace referencia al personal involucrado en el proyecto; en el caso de este trabajo se incluye el estudiante y el supervisor.

Para calcular el coste del trabajo del estudiante, se ha teniendo en cuenta que el proyecto ha tenido una extensión de 4 meses y medio a razón de 4 horas diarias, resulta un total de 360 horas.

Por otro lado, para calcular el coste del supervisor del trabajo, se tiene en cuenta el gasto temporal, es de media 4 horas semanales y, teniendo en cuenta la duración del trabajo, resultan 80 horas totales.

En la figura 44 se muestran los gastos totales de los recursos humanos.

Rol	**Horas de trabajo**
Ingeniero Junior	360
Supervisor de la universidad	80
Total	440

Figure 44. Coste en horas del proyecto – Fuente: Propia

7.2 Material

En este apartado, se calcula el coste del material usado en la realización del proyecto. Como solamente se ha instalado tres nodos, se calcularán los costes de estos tres prototipos que han sido construidos e instalados.

A continuación, en la figura 45, se detallan los costes de los materiales:

Material	Proveedor	Precio
Caja de empalmes	Leroy Merlin	3.49
Fusible + porta fusible x10	Aliexpress	3.88

Conector potencia DC	Aliexpress	1.76
Cable USB tipo B	Aliexpress	2.23
Cable tipo micro USB	Aliexpress	2.87
Cable Ethernet	Amazon	3.33
Arduino UNO	Amazon	23.4
Arduino UNO PCB shield	Aliexpress	4.45
Sensor de aire MQ135	Aliexpress	5.56
Módulo GPS NEO6MV2	Aliexpress	6.65
Módulo temperatura humedad AM2320B	Aliexpress	2.92
Resistencia 100Kohm	Miliwatts Mataró	0.05
Resistencia 330Kohm	Miliwatts Mataró	0.05
Resistencia 300 ohm	Miliwatts Mataró	0.05
Diodo 1N4002	Miliwatts Mataró	0.1
Condensador electrolítico 1000uF 25V	Miliwatts Mataró	2
Regulador de voltaje 5V 2A	Aliexpress	3.39
LED verde	Miliwatts Mataró	2
Raspberry Pi 3 Model B	Amazon	34.5
Micrófono USB	Aliexpress	2.55
Tarjeta de memoria micro SD 32 GB	Amazon	15.19

Total		115.08€

Figure 45. Coste detallado de una célula – Fuente: Propia

7.3 Software

En referencia al software, se ha hecho uso de varios IDE para programar los distintos componentes del software del proyecto. Pero en cuanto a programas que requieran de una licencia de pago para poder ser usados, únicamente se ha hecho uso de un programa a lo largo del trabajo. Este software es IntelliJ IDEA, de la compañía JetBrains. Este software ha sido utilizado para programar toda la parte Front-End en Angular 6.

A continuación, en la figura 46, se muestra el coste de la licencia, como el producto sólo dispone de licencia anual, para calcular el coste se ha tenido en cuenta la parte proporcional que corresponde a la duración del trabajo.

Producto	Desarrollador(es)	Licencia	Total
IntelliJ IDEA	JetBrains	499 €/año	166.33€

Figure 46. Coste del Software de desarrollo – Fuente: Propia

8 Conclusiones y trabajo futuro

Este proyecto trata sobre la implementación de un sistema de prevención, detección e identificación de posibles fallas de suministro eléctrico para sistemas alimentados con batería, con el objetivo de minimizar al máximo la probabilidad de corte eléctrico en ciertas situaciones donde se necesita flujo eléctrico constante.

El sistema parte de una ya existente red de videovigilancia municipal alimentado con sistemas de alimentación ininterrumpida que se cargan por las noches a través de la red de alumbrado público. Entonces, se ha diseñado un completo sistema con el propósito de cumplir con el principal requerimiento, esto es, prevenir fallos de suministro, y que además obtiene un conjunto de informaciones adicionales que pueden dar un valor añadido al servicio.

En este apartado, en primer lugar se sacarán las conclusiones del trabajo en función de los objetivos que fueron planteados al principio. Finalmente, se concluirá indicando futuras líneas a seguir.

8.1 Conclusiones

En relación con el primer objetivo planteado, el sistema contribuye positivamente a la empresa responsable del mantenimiento de los sistemas de alimentación, ya que, mediante notificaciones, anticipa fallos de suministro eléctrico que en la mayoría de casos se pueden solucionar antes de que ocurran. Además, proporciona información sobre la batería, como el voltaje instantáneo y un gráfico con valores históricos de carga, por lo que se puede determinar el estado de salud de la batería de forma completamente remota. De esta forma la empresa no solamente puede detectar y prevenir el fallo de suministro, sino que puede identificar su origen de forma totalmente remota.

En referencia con el objetivo planteado sobre el uso de últimas tendencias tecnológicas, se ha procurado usar Software y Hardware de fuente abierta de forma mayoritaria; En cuanto a hardware, se ha usado Arduino y Raspberry Pi, dos plataformas abiertas, en cuanto a software se ha usado Linux como sistema operativo y principalmente Node.js y Angular como frameworks de desarrollo, de código abierto también. Por otra parte se ha usado GIT tanto para el proyecto Back-End como para Front-End para el control de versiones. Finalmente también se puede concluir este punto de forma satisfactoria con el uso de frameworks de ultima generación como Angular 6 y Node.js, siendo de los mas recientes por el momento.

El sistema se ha instalado directamente en varios puntos de alimentación. En este sentido, se ha podido comprobar que el sistema presenta estabilidad y robustez frente a condiciones adversas: el sistema se comporta bien para gran diversidad de voltajes de alimentación y para temperaturas desfavorables de trabajo. Además, en caso de fallo de red, de suministro eléctrico, o de su propio software, el sistema es capaz de reconectarse y seguir funcionando sin ningún tipo de intervención humana.

Por otra parte también se ha cumplido con el objetivo de seguridad; sin conocer la clave es totalmente imposible obtener datos de los sensores, únicamente se puede acceder al servicio de Streaming de audio, pero en ningún momento esto compromete la seguridad ya que se trata de sonidos de espacios públicos como la calle. Además, como los microservicios y bases

de datos están gestionados por un Linux completamente actualizado, no presenta ningún tipo de vulnerabilidad frente ataques remotos. Por otra parte el mismo Linux cifra los datos de la tarjeta micro SD de la Raspberry, por lo que aunque existiera una intrusión física seria imposible obtener los datos del servidor.

En relación con la escalabilidad, se puede asegurar una muy buena escalabilidad tanto vertical como horizontal. Vertical porque las células disponen de una Raspberry Pi 3 que permite gran cantidad de desarrollos adicionales con procesamientos relativamente pesados y de forma completamente remota, es decir, sin necesidad de interacción humana. Finalmente, en referencia con la escalabilidad horizontal, el sistema permite añadir gran cantidad de nodos y usuarios; En la prueba de estrés del sistema, se demostró que la infraestructura tiene unos limites muy por encima de los limites que ofrecen los actuales servicios de licencia gratuita, por lo que únicamente seria necesario cambiar el tipo de servicio de Hosting y de Ngrok.

Por otra parte, también se han cumplido los objetivos académicos, ya que ha podido desarrollar un sistema completo E2E de forma totalmente autónoma. Esta implementación empieza desde la parte hardware de mas bajo nivel que es el circuito ubicado en la Shield PCB de Arduino hasta un Front-End totalmente diseñado para el usuario genérico, pasando por un Back-End basado en arquitectura de microservicios.

Además, tal y como se planteó en otro objetivo, se han podido consolidar conceptos impartidos en el MET. Se han visto temáticas relacionadas con Arduino desde otros puntos de vista y, además, se han ampliado conceptos de Front-End pero enfocados de forma totalmente distinta; En vez de desarrollar una aplicación Android, se ha desarrollando una aplicación Web con un framework de ultima generación y multiplataforma.

Como ultimo punto referente a los objetivos académicos, se ha podido adquirir un nuevo concepto que no se ha tocado en el máster, ya que es un concepto mas propio de másteres en Ingeniería de Software: se trata de la arquitectura de microservicios. Esto es un tipo de aplicación Back-End orientado a servidores. Su funcionamiento se basa en la interacción de múltiples servicios con Internet, comunicándose entre si de forma totalmente autónoma. Esto tiene gran relevancia en los proyectos de IoT, ya que determina la forma de la tipología y el comportamiento de los nodos.

Finalmente, para concluir este apartado, es importante mencionar el ultimo objetivo de crear un servicio aplicable a otras empresas con las misma problemática no ha podido ser cumplido. Como se conocían los requerimientos por parte de Sintelec, se ha creado un servicio que cumpliera con sus demandas, pero tal vez otras empresas requieran de avisos mas personalizables, o mediante otro tipo de canal. Este incumplimiento de este objetivo es debido a que el trabajo ha tenido un tiempo y alcance limitados, entonces se han priorizado los otros objetivos frente a este.

8.2 Desarrollo futuro

Como la infraestructura IT del trabajo ha partido de cero, la arquitectura Back-End y Front-End la ha sido diseñada y implementada desde cero, por lo que no se ha podido profundizar en ciertos puntos que se han considerado que tienen especial interés. En primer lugar, se propone seguir con el ultimo objetivo que se planteó y no ha sido alcanzado por falta de tiempo, que

como se comentó en el apartado anterior se trata de adaptar el servicio a otras empresas con la misma problemática.

Por otra parte, también se propone otra línea de desarrollo: un procesamiento en profundidad de los datos relacionados con la batería. De esta forma, se podría obtener parámetros acerca la disponibilidad y la vida útil de la batería, como los que se han estudiado en el apartado de Tecnologías Actuales Aplicables.

Finalmente, y en base a las observaciones posteriores a el montaje de las células en emplazamiento final, se propone investigar sobre el problema del estado de carga del 100% a lo largo del día.

Bibliografía

Ada, L. (2018). Adafruit AM2320 Sensor. *adafruit learning system* .

Bhatt, J. M. (2013). Effect of Temperature on Battery Life and Performance in Electric Vehicle. *Institute of Engineering and Technology, Ahmadabad (Gujarat), India.*

Chang, W.-Y. (2013). The State of Charge Estimating Methods for Battery: A Review. *Hindawi* .

González, C. P. (2016). Deteccion y seguimiento de objetos por colores en una plataforma Raspberry Pi. *Universidad Politécnica de Madrid* .

Martin, S. (2019). Angular vs React vs Vue: Which is the Best Choice for 2019? *Hackernoon* .

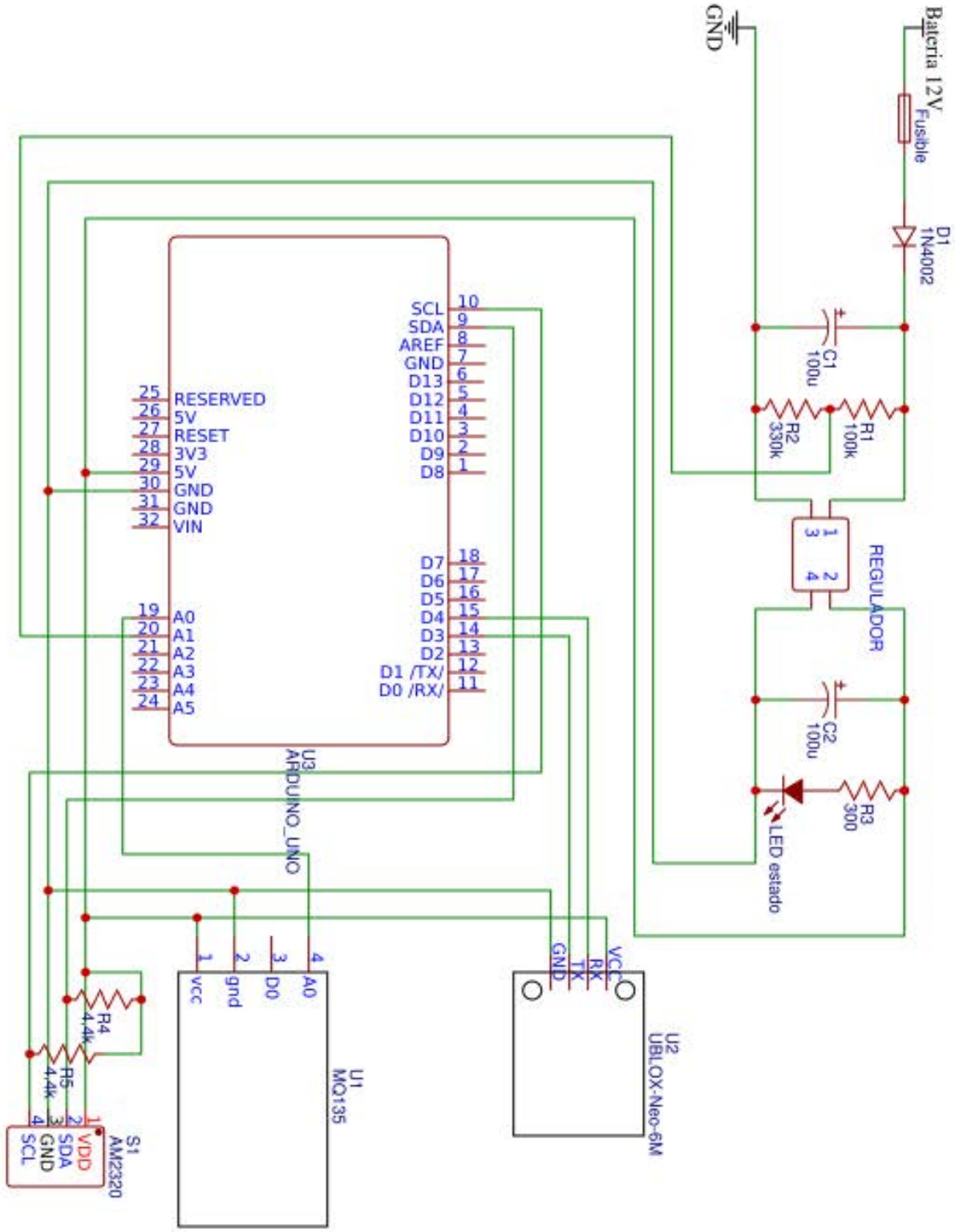
Pousa, A. (2011). ALGORITMO DE CIFRADO SIMÉTRICO AES. *Universidad Nacional de La Plata* .

Tovar, L. C. (2010). IMPLEMENTACIÓN DE ALGORITMOS DE ENCRIPTACIÓN: UNA ESTRATEGIA DE SEGURIDAD PARA LA PROTECCIÓN DE LA INFORMACIÓN. *Revista Colombiana de Tecnologías de Avanzada* .

Valenzuela, P. P. (2018). Utilización de SQLite para almacenamiento de información en puertos I/O de Raspberry Pi con Raspbian. *medium* .

Anexo

Esquemático del circuito de la Shield de Arduino



Código Arduino para la obtención de datos de los sensores

arduinogetdata.ino

```
#include <math.h>
#include "MQ135.h"
#include "AM2320.h"
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include <Wire.h>

unsigned long UpdateDelay = 1000UL * 20 * 1;  //Update frequency (5 minutes)
const byte NbSamples = 8; //Averaging

//Battery
const int batPin = A1;
long batVoltage = -1;

//AM2320 (temp and hum)
AM2320 am;
const int AIR_PIN = A0;
float temp = -1;
float hum = -1;

//GPS
char dato=' ';
MQ135 gasSensor = MQ135(A6);
SoftwareSerial serialgps(4,3);
TinyGPSPlus gps;
int year;
byte month, day, hour, minute, second, hundredths;
unsigned long chars;
unsigned short sentences, failed_checksum;
float latitude = -1;
float longitude = -1;
float alt = -1;

//Gas Sensor
float rawAir = -1;
float rZero = -1;
float ppm = -1;


void setup()
{
  delay(1000);
  Serial.begin(9600); //Start serial port
  Wire.begin();
  serialgps.begin(9600);
}

void loop(){
  updateValues();
  sendString();
  //delay(UpdateDelay);
}
```

```
void updateValues(){
  batVoltage = analogRead(batPin);
  am.Read();
  temp = am.t;
  hum = am.h;
  updateGPS();
  rawAir = analogRead(AIR_PIN);
  rZero=gasSensor.getRZero();
  ppm= gasSensor.getPPM();
}

void sendString(){
  Serial.println("{\"bat\":\"" + String((long)round(100.0 * batVoltage)) +
      "\", \"temp\":\"" + String((long)round(100.0 * temp)) +
      "\", \"hum\":\"" + String((long)round(100.0 * hum)) +
      "\", \"lat\":\"" + String((long)round(100.0 * latitude)) +
      "\", \"long\":\"" + String((long)round(100.0 * longitude)) +
      "\", \"alt\":\"" + String((long)round(100.0 * alt)) +
      "\", \"raw\":\"" + String((long)round(100.0 * rawAir)) +
      "\", \"rzero\":\"" + String((long)round(100.0 * rZero )) +
      "\", \"ppm\":\"" + String((long)round(100.0 * ppm)) +
      "\"}");
}

void updateGPS(){
  int readCycles = 100;
  while(readCycles>-1){
    if(serialgps.available() > 0){
      gps.encode(serialgps.read());
      if(gps.location.isUpdated()) {
        latitude = gps.location.lat();
        longitude= gps.location.lng();
        alt = gps.altitude.meters();
        readCycles=0;
      }
    }
    readCycles--;
  }
}
```

Código Node.js para el servicio Back-End en Raspberry Pi

saiBackService.js

```
/*jslint es6 */
"use strict";

//Dependencies
const express = require("express");
const ngrok = require("ngrok");
const http = require("http");
const nodestatic = require("node-static");
const TelegramBot = require("node-telegram-bot-api");
const proxy = require("express-http-proxy");
const CryptoJS = require("node-cryptojs-aes").CryptoJS;
const sqlite3 = require("sqlite3");
const sys = require("sys");
const SerialPort = require("serialport");

//Config
const key = "d6F3Efeq";
const isEncrypt = true;
const arduinoSerialPort = "/dev/ttyACM0";
const token = "799864144:AAFYArmIUiBmZaUCqxaolRA5Taj3Gnnogds";
const DODalert = 50;
const lowBatAlert = 30;
const location= "Sant Vicenç de Montalt, C/ Sol Naixent";

//Variables
const bot = new TelegramBot(token, { polling: true });
var app = express();
var db = new sqlite3.Database("./piBat.db");
const Readline = SerialPort.parsers.Readline;
const port = new SerialPort(arduinoSerialPort);
const parser = new Readline();
port.pipe(parser);
var staticServer = new nodestatic.Server(".");
var restUrl;
var restID;
var sshUrl;
var cell = "NaN";
var batVolt = NaN;
var batPercent = NaN;
var date = NaN;
var chatId = NaN;
var dodFlag = false;
var lowFlag = false;

//Constants

// Matches "/echo [whatever]"
bot.onText(/\/echo (.+)/, function (msg, match) {
    // 'msg' is the received Message from Telegram
    // 'match' is the result of executing the regexp above on the text content
    // of the message

    chatId = msg.chat.id;
```

```
    const resp = match[1]; // the captured "whatever"


    const baseURL = "http://saiio.000webhostapp.com/#/cell/";

    // send back the matched "whatever" to the chat
    bot.sendMessage(chatId, baseURL + restID);
});

//Connect ngrok to rest API
(async function () {
    restUrl = await ngrok.connect({
        proto: "http",
        addr: 8001,
        authtoken: "22Ywb26viDtAwYjnouhBp_3CmqyRhsJ1tWC3qxzRbwq"
    });
    console.log("Ngrok connected [localhost:8001] at url: [" + restUrl + "]");
    restID = restUrl.split("/")[2].split(".")[0];
})();

//Connect ngrok to SSH
(async function () {
    sshUrl = await ngrok.connect({
        proto: "tcp",
        addr: 22,
        authtoken: "22Ywb26viDtAwYjnouhBp_3CmqyRhsJ1tWC3qxzRbwq"
    });
    console.log("Ngrok connected [localhost:22] at url: [" + sshUrl + "]");
})();

app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
    next();
});

app.get("/data", (req, res) => {
    res.send(cell);
});

app.get("/:data", function (request, response) {
    // Grab the URL requested by the client and parse any query options
    var url = require("url").parse(request.url, true);
    var pathfile = url.pathname;
    var query = url.query;

    // Test to see if it's a database query
    if (pathfile == "/battery_query.json") {
        // Test to see if number of observations was specified as url query
        if (query.num_obs) {
            var num_obs = parseInt(query.num_obs);
        }
        else {
            // If not specified default to 20. Note use -1 in query string to get all.
            var num_obs = -1;
        }
        if (query.start_date) {
            var start_date = query.start_date;
```

```
        }
        else {
            var start_date = "1970-01-01T00:00";
        }
        // Send a message to console log
        console.log("Database query request from " + request.connection.remoteAddress + " for " + num_obs + "
records from " + start_date + ".");
        // call selectBat function to get data from database
        selectBat(num_obs, start_date, function (data) {
            response.writeHead(200, { "Content-type": "application/json" });
            response.end(encrypt(JSON.stringify(data)), "ascii");
        });
        return;
    }

    // Test to see if it's a request for current battery
    if (pathfile == "/battery_now.json") {
        readBat(function (data) {
            response.writeHead(200, { "Content-type": "application/json" });
            response.end(encrypt(JSON.stringify(data)), "ascii");
        });
        return;
    }

    // Handler for favicon.ico requests
    if (pathfile == "/favicon.ico") {
        response.writeHead(200, { "Content-Type": "image/x-icon" });
        response.end();

        // Optionally log favicon requests.
        //console.log('favicon requested');
        return;
    } else {
        // Print requested file to terminal
        console.log("Request from " + request.connection.remoteAddress + " for: " + pathfile);

        // Serve file using node-static
        staticServer.serve(request, response, function (err, result) {
            if (err) {
                // Log the error
                sys.error("Error serving " + request.url + " - " + err.message);

                // Respond to the client
                response.writeHead(err.status, err.headers);
                response.end("Error 404 - file not found");
                return;
            }
            return;
        })
    }
});

app.use("/icecast", proxy("http://localhost:8000"));

http.createServer(app).listen(8001, () => {
    console.log("Server started at http://localhost:8001");
});
```

```
parser.on("data", function (data) {//When a new line of text is received from Arduino over USB
    try {
        var myJsonObject = JSON.parse(data); //change to obj
        batVolt = parseInt(myJsonObject.bat * 1142 / 5070, 10);
        batPercent = batVoltToPercent(batVolt / 1000);
        manageTelegramBatPercent(batPercent);
        date = new Date();
        myJsonObject.dateLastInfo = date;
        myJsonObject.id = restID;
        myJsonObject.batVolt = batVolt.toString();
        myJsonObject.batPercent = batPercent.toString();
        data = JSON.stringify(myJsonObject);
        cell = encrypt(data);
    }
    catch (ex) {
        console.warn(ex);
    }
});

const encrypt = (text) => {
    if (isEncrypt) {
        try {
            return CryptoJS.AES.encrypt(text, key).toString();
        } catch (ex) {
            console.log(ex);
        }
    } else {
        return text;
    }

}

function insertBat(data) {
    // data is a javascript object
    var statement = db.prepare("INSERT INTO battery_records VALUES (?, ?)");
    // Insert values into prepared statement
    statement.run(data.battery_record[0].unix_time, data.battery_record[0].charge);
    // Execute the statement
    statement.finalize();
}

function readBat(callback) {
    // Add date/time to battery
    var data = {
        battery_record: [{
            unix_time: Date.now(),
            charge: batPercent
        }]
    };
    // Execute call back with data
    callback(data);
};

function logBat(interval) {
    // Call the readBat function with the insertBat function as output to get initial reading
    readBat(insertBat);
    // Set the repeat interval (milliseconds). Third argument is passed as callback function to first (i.e.
readBat(insertBat)).
```

```
    setInterval(readBat, interval, insertBat);
};

function selectBat(num_records, start_date, callback) {
    // - Num records is an SQL filter from latest record back trough time series,
    // - start_date is the first date in the time-series required,
    // - callback is the output function
    var current_bat = db.all("SELECT * FROM (SELECT * FROM battery_records WHERE unix_time >
(strftime('%s',?)*1000) ORDER BY unix_time DESC LIMIT ?) ORDER BY unix_time;", start_date, num_records,
        function (err, rows) {
            if (err) {
                response.writeHead(500, { "Content-type": "text/html" });
                response.end(err + "\n");
                console.log("Error serving querying database. " + err);
                return;
            }
            var data = { battery_record: [rows] }
            callback(data);
        });
};

// Start battery logging (every 5 min).
var msecs = (60 * 5) * 1000; // log interval duration in milliseconds
logBat(msecs);
// Send a message to console
console.log("Server is logging to database at " + msecs + "ms intervals");

function batVoltToPercent(voltage) {
    if(voltage<11.8){
        return 0;
    }else if (voltage >= 11.8 && voltage <12){
        let M = (25+11.8)/12;
        let N = -11.8 * M;
        return Math.round(M*voltage + N);
    }else if (voltage >= 12 && voltage <12.2){
        let M = (55+12)/12.2;
        let N = -12 * M;
        return Math.round(M*voltage + N);
    }else if (voltage >= 12.2 && voltage <12.4){
        let M = (75+12.2)/12.4;
        let N = -12.2 * M;
        return Math.round(M*voltage + N);
    }else {
        return 100;
    }
}

function manageTelegramBatPercent(batPerc){
    if (batPerc < DODalert && !dodFlag) {
        dodFlag = true;
        if (chatId){
            bot.sendMessage(chatId, "[Atenció] El nivell de bateria de "+location+" ha baixat del "+batPerc+"%, mes
informació: " + baseURL + restID);
        }
    } else {
        dodFlag= false;
    }
    if (batPerc < lowBatAlert && !lowFlag) {
```

```
        lowFlag = true;
        if (chatId){
            bot.sendMessage(chatId, "[Alerta] El nivell de bateria de "+location+" ha baixat del "+batPerc+"%, mes
informació: " + baseURL + restID);
        }
    }else{
        lowFlag = false;
    }
}
```

Código Angular para la aplicación web

(Partes más relevantes del proyecto Angular)

Modelo:

cell.ts

```
export class Cell {
  id: string;
  location: string;
  batteryLevel: number;
  batteryVoltage: number;
  inverterVoltage: number;
  cellStatus: string;
  position: {lat: number, long: number, alt: number };
  dateLastInfo: string;
  humidity: number;
  temperature: number;
  ppm: number;
  raw: number;
  rzero: number;
  plotData: any[] = [];
}
```

Vista:

cell-detail.component.ts

```
import { Component, Input, OnInit, SimpleChanges, HostListener } from '@angular/core';
import { Cell } from '../../model/cell';
import * as Highcharts from 'highcharts';
import { HttpClient } from '@angular/common/http';
import { CellService } from 'src/app/controller/cell.service';

@Component({
  selector: 'app-cell-detail',
  templateUrl: './cell-detail.component.html',
  styleUrls: ['./cell-detail.component.css']
})
export class CellDetailComponent implements OnInit {
  @Input() cell: Cell;

  @Input() plotData;

  options: Highcharts.Options = {
    chart: {
      renderTo: 'container',
      // type: 'spline',
      zoomType: 'x',
      reflow: true
    },
    series: [{
      id: 'series',
      name: 'SoC',
      type: 'line',
      data: [1, 2, 3]
    }],
    title: {
      text: 'Plot of battery state of charge'
    },
    subtitle: {
```

```
    text: 'Click and drag in the plot area to zoom in',
    align: 'right',
  },
  responsive: {
    rules: [{
      condition: {
        maxWidth: 500
      },
      chartOptions: {
        legend: {
          align: 'center',
          verticalAlign: 'bottom',
          layout: 'horizontal'
        },
        yAxis: {
          min: 0, max: 100,
          labels: {
            align: 'left',
            x: 0,
            y: -5
          },
          title: {
            text: null
          }
        },
        subtitle: {
          text: null
        }
      }
    }]
  },
  xAxis: {
    type: 'datetime',
    tickPixelInterval: 150,
    maxZoom: 20 * 1000,
    title: {
      text: 'Time',
      margin: 15
    }
  },
  yAxis: {
    minPadding: 0.2,
    maxPadding: 0.2,
    showFirstLabel: false,
    title: {
      text: 'Battery %',
      margin: 15
    }
  },
  plotOptions: {
    area: {
      fillColor: {
        linearGradient: { x1: 0, y1: 0, x2: 0, y2: 1 },
        stops: [
          [0, Highcharts.getOptions().colors[0]],
          [1, 'rgba(2,0,0,0)'],
        ]
      },
      lineWidth: 1,
      marker: {
        enabled: false,
        states: {
          hover: {
            enabled: true,
            radius: 5
          }
        }
```

```
      },
      shadow: false,
      states: {
        hover: {
          lineWidth: 1
        }
      },
      threshold: null
    },
  },
};

Highcharts: typeof Highcharts = Highcharts; // required
chartOptions: Highcharts.Options = this.options; // required

ngOnChanges(changes: SimpleChanges) {
  this.updateChart(changes.plotData.currentValue);
}

updateChart(data) {
  this.options.series[0].data = data;
}

constructor(private httpClient: HttpClient, private cellService: CellService) { }

ngOnInit() {
}

@HostListener('window:resize', ['$event'])
onResize(event) {
  this.updateChartSize();
}

updateChartSize() {
  const x = window.innerWidth;
  if (x !== NaN && x <= 500) {
    Highcharts.charts[0].setSize(x - 55, 400, false);
  } else {
    Highcharts.charts[0].setSize(500, 400, false);
  }
}
}
```

cell-detail.component.html

```
<div *ngIf="cell">
  <h2>{{cell.location | uppercase}} {{'cell.details' | translate}}</h2>
  <div>
    <h3>{{'dataType.basicInfo' | translate}}</h3>
    <tr>
      <td>{{'cell.id' | translate}}:</td>
      <td>{{cell?.id}}</td>
    </tr>
    <tr>
      <td>{{'cell.dateLastInfo' | translate}}:</td>
      <td>{{cell?.dateLastInfo}} V</td>
    </tr>
    <h3>{{'dataType.batteryInfo' | translate}}</h3>
    <tr>
      <td>{{'cell.batteryVoltage' | translate}}:</td>
      <td>{{cell?.batteryVoltage | number : '1.2'}} V</td>
    </tr>
    <tr>
      <td>{{'cell.batteryLevel' | translate}}:</td>
      <td>{{cell?.batteryLevel}} %</td>
    </tr>
    <tr>
```

```
      <td>{{'cell.cellStatus.title' | translate}}:</td>
      <td>{{'cell.cellStatus.status.' + cell.cellStatus | translate}}</td>
    </tr>
  </div>
  <highcharts-chart
    [Highcharts]="Highcharts"
    [options]="chartOptions"
    style="width: 100%; height: 400px; display: block;"
  ></highcharts-chart>
  <div>
    <h3>{{'dataType.environmentInfo' | translate}}</h3>
    <tr>
      <td>{{'cell.humidity' | translate}}:</td>
      <td>{{cell?.humidity | number}} %</td>
    </tr>
    <tr>
      <td>{{'cell.temperature' | translate}}:</td>
      <td>{{cell?.temperature}} ºC</td>
    </tr>
    <h3>{{'dataType.airQuality' | translate}}</h3>
    <tr>
      <td>{{'cell.ppm' | translate}}:</td>
      <td>{{cell?.ppm | number : '1.2'}} ppm</td>
    </tr>
    <tr>
      <td>{{'cell.raw' | translate}}:</td>
      <td>{{cell?.raw}}</td>
    </tr>
    <tr>
      <td>{{'cell.rzero' | translate}}:</td>
      <td>{{cell?.rzero | number : '1.2'}}</td>
    </tr>
    <h3>{{'dataType.gps' | translate}}</h3>
    <tr>
      <td>{{'cell.longitude' | translate}}:</td>
      <td>{{cell?.position?.long}}</td>
    </tr>
    <tr>
      <td>{{'cell.latitude' | translate}}:</td>
      <td>{{cell?.position?.lat}}</td>
    </tr>
    <tr>
      <td>{{'cell.altitude' | translate}}:</td>
      <td>{{cell?.position?.alt | number : '1.2'}}</td>
    </tr>
  </div>

</div>
```

Controlador:

cell.service.ts

```
import { Injectable } from '@angular/core';
import {Cell} from '../model/cell';
import { HttpClient } from '@angular/common/http';
import {Observable, of, forkJoin} from 'rxjs';
import {map} from 'rxjs/operators';
import * as CryptoJS from 'crypto-js';
import {CELL1} from '../model/mock-cell';
import {formatDate} from '@angular/common';

@Injectable({
  providedIn: 'root'
})
export class CellService {
```

```
protected cell: Cell;
private protocol = 'http://';
private domain = '.ngrok.io';
private encryptSecretKey = 'd6F3Efeq';
private plotDaysBefore = 30;

constructor(private httpClient: HttpClient) { }

getCells(id: string[]): Observable<Cell>[] {
  let cells: Observable<Cell>[];
  cells = [];
  id.forEach( (uId) => {
    cells.push(this.getDataFromCell(uId));
  });
  return cells;
}

getDataFromCell(url: string): Observable<Cell> {
  return forkJoin(this.getSensorDataFromCell(url),
  this.getPlotDataFromCell(url)
  ).pipe(map(([res1, res2]) => {
      res1.plotData = res2;
      return res1;
    }));
}

mapCell(obj: any): Cell {
  console.log(obj);
  if (obj) {
    let cell: Cell;
    cell = new Cell();
    const obj1 = obj;
    if (obj1.id) {
      cell.id = obj1.id;
    }
    if (obj1.batVolt) {
      cell.batteryVoltage = this.parseBattery(obj1.batVolt);
      if (cell.batteryLevel < 20) {
        cell.cellStatus = 'low';
      } else if (cell.batteryLevel >= 20 && cell.batteryLevel < 60) {
        cell.cellStatus = 'normal';
      } else {
        cell.cellStatus = 'good';
      }
    }
    if (obj1.batPercent) {
      console.log(obj1.batPercent);
      cell.batteryLevel = parseFloat(obj1.batPercent);
    }
    if (obj1.hum) {
      cell.humidity = this.parseFloatFromCell(obj1.hum);
    }
    if (obj1.temp) {
      cell.temperature = this.parseFloatFromCell(obj1.temp);
    }
    if (obj1.lat && obj1.long && obj1.alt) {
      cell.position = this.parseGPSData(obj1.lat, obj1.long, obj1.alt);
    }
    if (obj1.ppm) {
      cell.ppm = this.parseFloatFromCell(obj1.ppm);
    }
    if (obj1.raw) {
      cell.raw = this.parseFloatFromCell(obj1.raw);
    }
    if (obj1.rzero) {
      cell.rzero = this.parseFloatFromCell(obj1.rzero);
    }
```

```
      return cell;
    }
    return null;
  }

  parseFloatFromCell(param: string): number {
    if (param !== '-100') {
      return parseFloat(param) / 100;
    } else {
      return null;
    }
  }

  parseBattery(bat: string): number {
    return this.parseFloatFromCell(bat) / 10;
  }

  parseGPSData(lat: string, long: string, alt: string): {lat: number, long: number, alt: number } {
    let position: {lat: number, long: number, alt: number };
    if (lat !== '-100' && long !== '-100' && alt !== '-100') {
      position = {lat: parseFloat(lat) / 10, long: parseFloat(long) / 10, alt: parseFloat(alt) / 10};
    }
    return position;
  }

  decrypt(data) {
    try {
      const bytes = CryptoJS.AES.decrypt(data, this.encryptSecretKey);
      if (bytes.toString()) {
        return JSON.parse(bytes.toString(CryptoJS.enc.Utf8));
      }
      return data;
    } catch (e) {
      console.log(e);
    }
  }

  getSensorDataFromCell(url: string): Observable<Cell> {
    return this.httpClient.get(this.protocol + url + this.domain + '/data', {responseType: 'text'})
    .pipe(map(res => this.decrypt(res)), map(res => this.mapCell(this.decrypt(res))));
  }

  getPlotDataFromCell(url: string): Observable<any[]> {
  return this.httpClient.get(this.protocol + url + this.domain + '/battery_query.json?num_obs=-1&start_date=' +
formatDate(
    new Date().setDate(new Date().getDate() - this.plotDaysBefore), 'yyyy-MM-dd', 'en') + 'T16:00', {responseType:
'text'})
    .pipe(map(res => this.decrypt(res)), map((res: any) => {
      const data: any[] = [];
      let i = 0;
      // Iterate JSON data series and add to plot
      while (res.battery_record[0][i]) {
        data.push([res.battery_record[0][i].unix_time, res.battery_record[0][i].charge]);
        i++;
      }
      return data;
    }));
  }
}
```