Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de red de sensores IoT de bajo coste y bajo consumo

Development of low cost and low consumption IoT sensor network

Autor

Francisco Bas Jalón

Director

Bonifacio Martín del Brío

Departamento de Ingeniería Electrónica y Comunicaciones
Escuela de Ingeniería y Arquitectura de Zaragoza

Año académico: 2018 – 2019

Resumen

El objetivo de este trabajo es desarrollar una red de nodos de sensores de temperatura IoT (*Internet of Things*) alimentados mediante batería. Los nodos sensores deben ser pequeños y de muy bajo consumo. Asimismo, tienen que conectarse a Internet y almacenar las temperaturas registradas por todos los sensores de la red. Posteriormente el usuario tiene que poder visualizar las medidas tomadas.

Para abordar este problema se ha optado por diseñar un nodo central que se comunica mediante Wi-Fi, tanto con el rúter de la red local, como con el resto de nodos sensores de la red. Para almacenar las temperaturas se ha empleado tanto la propia memoria del módulo Wi-Fi del nodo central (ESP-12E), como un servidor proporcionado por la plataforma thinger.io, empresa dedicada a proveer servicios del IoT.

Las temperaturas almacenadas en la nube, se pueden visualizar usando las herramientas que la plataforma thinger.io proporciona. Para visualizar las temperaturas almacenadas en el nodo central es necesario emplear un navegador conectado a la misma red local que la red de sensores.

Para conseguir esto ha sido necesario desarrollar el firmware de los módulos Wi-Fi. Este firmware gestiona las conexiones Wi-Fi, la toma, transmisión y recepción de las temperaturas, la gestión de los ficheros y los archivos de texto, la obtención de la fecha y la hora a través de internet y la conexión con el servidor de thinger.io. También ha sido necesario desarrollar una página web que permita visualizar desde el navegador las temperaturas almacenadas en el nodo central.

En cuanto al hardware, se ha diseñado el circuito sensor de temperatura, basado en un termistor NTC, así como los circuitos de carga de la batería y alimentación del sensor y del módulo Wi-Fi. A partir de dichos diseños se han fabricado tres prototipos, el nodo central, alimentado directamente desde la red, y dos sensores más, alimentados con baterías. Se ha comprobado que estos prototipos funcionan correctamente.

A la hora de hacer el diseño del hardware se ha procurado que el precio y el consumo sean lo más bajos posibles. Aunque en los prototipos se han fabricado con componentes convencionales, también se ha realizado el diseño de una PCB con componentes SMD para intentar reducir el tamaño final del diseño.

Índice

Lista de figuras

Lista de ecuaciones

Anexos

1 Introducción

1.1 Objetivos

El principal objetivo de este proyecto es desarrollar un prototipo funcional de una red de nodos de sensores de temperatura IoT ("*Internet of Things*", Internet de las cosas). Es decir, que los sensores tienen la capacidad para conectarse a internet y enviar datos autónomamente, sin intervención humana.

Estos sensores son adecuados para su empleo en un ámbito domótico, no industrial. Es decir, no estarán sometidos a temperaturas extremas ni requerirán de una gran exactitud.

A parte de su correcta funcionalidad, se ha prestado especial atención al consumo de energía, el precio de los componentes y al tamaño final de la PCB y del sensor. Cada uno de los sensores es alimentado mediante una batería de iones de litio de una celda. Se pretende obtener una autonomía lo mayor posible para un precio y un tamaño ajustados, lo que redunda en un consumo mínimo de energía.

1.2 Procedimiento

A la hora de realizar el proyecto, la primera idea planteada es desarrollar un sensor de temperatura conectado a un módulo Wi-Fi con posibilidad de ser duplicado n veces. De esta forma cada uno de los módulos estaría conectado a la red Wi-Fi y se encargaría de medir y enviar las temperaturas tomadas a un servidor específico para aplicaciones de IoT. Dichos servidores son proporcionados por múltiples empresas en la actualidad.

Esta solución sería correcta y totalmente funcional. Sin embargo, pensando desde un punto de vista económico, sería poco eficiente. Esto es así porque las empresas que proporcionan los servidores para aplicaciones de IoT cobran por su uso. Normalmente cobran por dispositivo conectado y por mes (entorno al medio euro por dispositivo y mes). Esto quiere decir que los costes de mantenimiento podrían superar a los costes de fabricación en menos de un año.

Además, el hecho de tener que conectar cada uno de los sensores a la red Wi-Fi podría resultar tedioso para el usuario final.

La solución planteada, que consigue solucionar ambos problemas, es desarrollar un nodo central que se encarga de recibir las temperaturas tomadas por el resto de los sensores y almacenarlas en ficheros de texto junto con la fecha y la hora. Este nodo cumple una doble funcionalidad, por un lado, actúa como servidor en una red local, y por otro, sube las temperaturas a la nube.

Al ser solo un dispositivo el encargado de subir todas las temperaturas, no es necesario pagar por cada uno de los dispositivos mensualmente sino sólo por uno. Incluso hay plataformas como la empleada en este proyecto que permiten un número de dispositivos gratis. Además, únicamente es necesario conectar el nodo central a la red Wi-Fi, conexión que se realiza de forma sencilla mediante el estándar WPS.

La siguiente figura muestra un pequeño esquema de las conexiones:

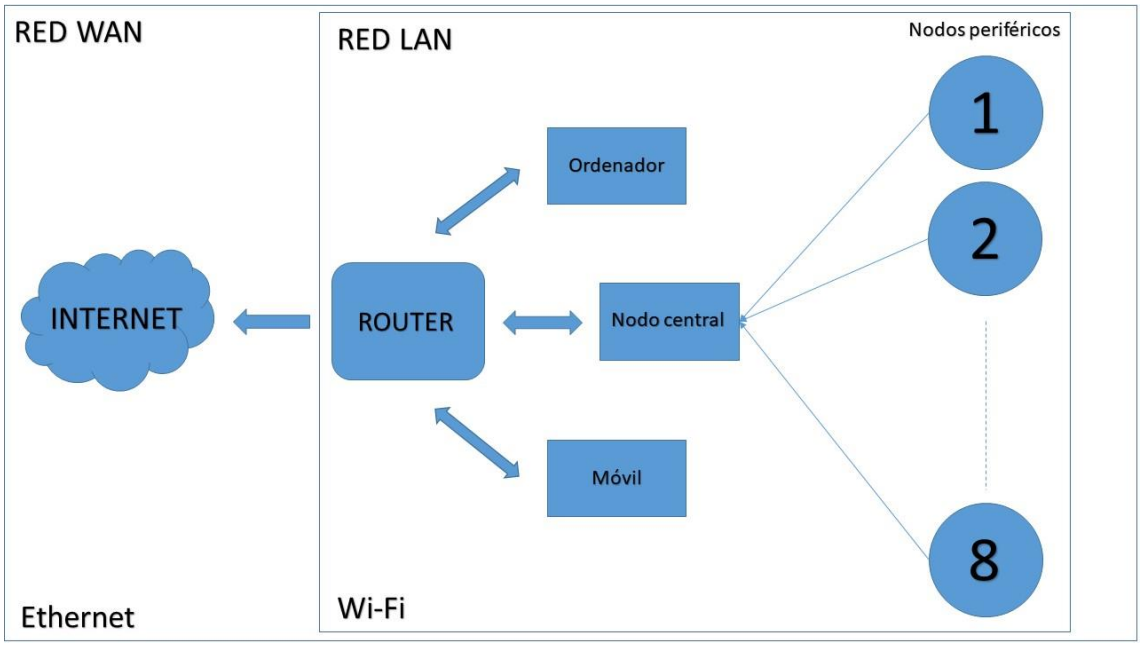


Figura 1. Esquema de la red y de las conexiones

Empleando esta solución para desarrollar nuestro proyecto tenemos dos formas diferentes de visualizar los datos. En primer lugar, usando un navegador en un dispositivo (teléfono móvil, ordenador) conectado a la misma red Wi-Fi que el módulo Wi-Fi del nodo central. En segundo lugar, desde la plataforma que nos brinda la misma empresa que nos proporciona el servicio IoT. Estas plataformas suelen tener distintas opciones para la gestión y visualizado de los datos que varían dependiendo de la empresa.

El nodo central debe estar constantemente activo y preparado para recibir información del resto de los módulos Wi-Fi o responder a una petición del navegador. Esto hace inviable alimentarlo mediante una batería durante un periodo largo de tiempo. Por lo tanto, este módulo es alimentado directamente por la red eléctrica. Para no desaprovechar recursos, el nodo central también cuenta con un sensor de temperatura.

Por el contrario, los nodos periféricos están “dormidos” la mayor parte del tiempo, únicamente están activos mientras miden y envían la temperatura (aproximadamente entre 0,5 y 5 segundos cada 300). Esto hace que sea perfectamente posible que la fuente de alimentación sea una batería de iones de litio.

Cada uno de los sensores está basado en un termistor NTC como sensor de temperatura, solución elegida por su bajo precio y simplicidad. La tensión del sensor se lee con un módulo Wi-Fi ESP-12E basado en el SoC ESP8266, uno de los módulos Wi-Fi más populares y más económicos en el mercado con un precio de 1,21 €. Dicho módulo posee 4 MB de memoria flash, aunque únicamente es posible emplear 3 para alojar los ficheros.

Puesto que la memoria del nodo central es limitada, es necesario ir eliminando datos para dejar espacio a otros nuevos una vez la memoria está llena. Dependiendo de la cantidad de

sensores conectados, el sensor es capaz de almacenar temperaturas por periodos de tiempo más o menos largos, variando aproximadamente desde 140 hasta 700 días.

El SoC ESP8266 permite ser configurado como punto de acceso y estación al mismo tiempo, esto quiere decir que además de conectarse a una red Wi-Fi, se encarga de generar otra a la que pueden conectarse hasta 8 dispositivos. Por lo tanto, cada vez que uno de los nodos periféricos sale del modo de bajo consumo, se conecta a la red Wi-Fi generada por el nodo central y envía la temperatura medida.

En este proyecto se han desarrollado los prototipos del nodo central y de dos nodos periféricos. Aun así, el desarrollo del firmware del nodo central se ha hecho pensando que puede haber en cualquier momento de 0 a 8 módulos Wi-Fi conectados.

Como ya se ha mencionado, el sensor está formado por un termistor NTC en serie con una resistencia. Se escoge un termistor NTC como transductor puesto que es barato y cumple los requisitos de diseño necesarios.

Además de todo lo mencionado, es necesario desarrollar una página web capaz de representar los datos almacenados en los ficheros de texto alojados en el nodo central.

1.3 Planificación

Para desarrollar la red de sensores IoT y cumplir con los objetivos inicialmente propuestos, se establecen las siguientes fases en el desarrollo del proyecto:

La primera fase del proyecto consistió en **elegir un módulo Wi-Fi** que permita enviar las temperaturas medidas a un nodo central o servidor. El criterio primario a la hora de realizar esta elección es el precio.

La siguiente fase fue el **estudio del módulo elegido (ESP-12E)** para conocer y valorar las posibilidades que este ofrece. Conociendo las distintas opciones que el módulo brinda, se realiza **el primer boceto completo del proyecto**.

Con el proyecto ya definido, el siguiente paso es **desarrollar el firmware del módulo Wi-Fi y el software de la página web**. Para el desarrollo del firmware se emplea la placa de desarrollo NodeMCU con el módulo ESP-12E integrado.

La siguiente fase es **el diseño del sensor**. De entre los muchos sensores de temperatura posibles, se busca diseñar uno barato y que consuma poca corriente, sin necesidad de que tenga un rango o una precisión especialmente grandes.

Con el firmware, la página web y el sensor diseñados, se **hace una prueba en una placa blanca** para comprobar que todo funciona según lo planeado y solucionar los errores que puedan aparecer. Para esto se emplea la placa de desarrollo NodeMCU que actúa como nodo central junto con un módulo ESP-12E que es el encargado medir y enviar las temperaturas.

El siguiente paso es **medir la corriente consumida por el nodo periférico**. Una vez medida la corriente consumida, se puede elegir la batería sabiendo cuál será su autonomía aproximada.

La fase siguiente consiste en **diseñar el circuito final y elegir el resto de componentes** (integrado de carga de la batería, reguladores de tensión, condensadores, conectores, etc.).

Con el diseño del circuito cerrado, **se diseña la PCB** prestando especial atención a minimizar el ruido en todo el circuito y especialmente en la señal correspondiente a la temperatura.

La última fase es **fabricar, montar y probar la PCB** depurando los posibles errores que puedan aparecer.

1.4 Organización de la memoria

El resto de la memoria se ha dividido en los siguientes capítulos que resumimos aquí brevemente:

- **Capítulo 2: Estado del arte. Elección del módulo y del sensor.** En este capítulo se describe brevemente el concepto de Internet de las cosas, se hace un repaso por los módulos de Internet de las cosas disponibles en el mercado y sus características, y se justifica la elección del módulo ESP-12E. También se hace un repaso de los tipos de sensores de temperatura empleados en el mercado y se explica la elección del termistor NTC.
- **Capítulo 3: Diseño del hardware.** En este capítulo se tratan todos los temas del proyecto referentes al hardware: hardware y características del módulo ESP-12E y de la placa de desarrollo NodeMCU, hardware y características del sensor de temperatura, consumo y autonomía del nodo periférico y diseño de los esquemáticos y de las placas de circuito impreso de ambos nodos.
- **Capítulo 4: Diseño del firmware y del software.** En este capítulo se trata todo lo referente al firmware y al software. Se analiza y explica cómo y dónde se almacenan las temperaturas tomadas. Se explica cómo se realizan las diferentes comunicaciones y cómo el nodo central actúa como un pequeño servidor. Por último, se profundiza en cómo se visualizan las temperaturas y se explica resumidamente el funcionamiento de la página web.
- **Capítulo 5: Resultados.** En este capítulo se muestran los prototipos desarrollados y se analiza brevemente su funcionamiento.
- **Capítulo 6: Conclusiones.** En este capítulo se analiza el trabajo realizado durante el proyecto. Y se valora si se han cumplido los objetivos inicialmente marcados. También se hace una valoración a nivel personal de que ha supuesto realizar este proyecto.

2 Estado del arte. Elección de módulo y sensor

2.1 Internet de las cosas

Desde el nacimiento de Internet nuestras vidas han cambiado mucho. La creación de una red mundial nos ha permitido comunicarnos rápida y fácilmente. Internet nos ha permitido adquirir cualquier tipo de información instantáneamente.

Hasta hace relativamente poco, solo ordenadores y móviles tenían conexión a Internet, pero ahora la tendencia está cambiando. Cada vez más, todo tipo de aparatos electrónicos incorporan conexión a Internet, desde electrodomésticos hasta sensores, vehículos o pulseras inteligentes. Cualquier aparato con capacidad de conectarse a Internet y enviar y recibir datos sin interferencia humana pertenece al Internet de las cosas.

Aunque cada día vemos más y más aparatos conectados a Internet, el concepto de Internet de las cosas es muy moderno y está aún en vías de desarrollo. Con el Internet de las cosas se pretende crear una gran red de información de tal manera que se generen automáticamente grandes cantidades de datos que puedan ser compartidos y analizados desde cualquier parte del mundo y así llegar a conclusiones y proceder de la manera más adecuada posible.

El concepto de Internet de las cosas es un concepto versátil que se puede aplicar en muchos y diferentes campos. Un sensor con conexión a Internet puede ser usado, por ejemplo, en una fábrica en un proceso de fabricación o como es nuestro caso en un entorno domótico. [1] [2]

2.2 Elección del módulo Wi-Fi

Para la elección del módulo Wi-Fi se valora fundamentalmente el precio, por lo que se busca un módulo Wi-Fi pequeño; que consuma poca energía; con un puerto de entrada analógico; que sea completamente autónomo, es decir, que no necesite de ningún otro componente electrónico para funcionar y que sea lo más barato posible.

Existen en el mercado una gran variedad de módulos Wi-Fi con diferentes características y precios. Para saber cuál es el más adecuado para el proyecto, se hace una revisión de los principales fabricantes y sus módulos, de sus precios y de sus características.

- **BeagleBoard:**

  Este fabricante ofrece la placa de desarrollo “BeagleBone Green Wireless”. A pesar de que esta placa es válida para el proyecto, puesto que cuenta con conexión Wi-Fi, sus características se asimilan más a las de una Raspberry (un mini ordenador). Por lo tanto, su precio es algo elevado.

- **Electronic Imp:**

  Esta empresa vende varios módulos destinados al Internet de las cosas con su propio software, llamado ImpOS, ya preinstalado. Este software se encarga de simplificar su programación y ofrecer al usuario una solución sencilla. El módulo más barato es algo caro, cuesta 21 dólares.

- **Mc-Things:**

Esta empresa ofrece también módulos para el Internet de las cosas. Aunque en su página no especifica precios, sus módulos incluyen varios tipos de sensores e incluso procesadores como el ARM4. Este tipo de módulos no son adecuados para el proyecto puesto que sus prestaciones superan las necesidades del proyecto.

- **Mediatek labs:**

Esta empresa tiene dos de placas de desarrollo para el Internet de las cosas que resultan interesantes: la "LinkIt Smart 7688" y la "LinkIt 7697". La primera, aunque encajaría en el diseño es algo más sofisticada de lo necesario, acepta programación en Python, Node.js o C. La segunda, está dotada de Wi-Fi y bluetooth, basada en el SoC MediaTek MT7697. Es el módulo más básico de mediatek labs y con unas especificaciones un poco más avanzadas que las de la placa de desarrollo empleada. Aún con todo, el precio sigue siendo algo alto para el diseño, 14,90 dólares.

- **Onion:**

Onion cuenta con dos módulos de los cuales uno puede resultar interesante:

Figura 2. Módulo Omega2+

Este módulo, el Omega2+ [3] está basado en Linux y se puede programar en diversos leguajes de programación C, C++, Python, PHP, Perl… A pesar de que sus especificaciones son bastante avanzadas: CPU MIPS de 580MHz, 128MB de memoria o 18 GPIOs. Su precio es de 13 dólares. Muy adecuado para las especificaciones que ofrece, pero un poco alto para el proyecto.

- **Particle:**

El producto más barato que oferta esta empresa para soluciones de IoT es el kit de desarrollo Xenon. Este módulo tiene conectividad Wi-Fi y Bluetooth, 20 GPIOs, leds RGB de estado y varias características que sobrepasan las necesidades del diseño. Además, tiene un sistema de archivos de 2Mb, el cual se puede quedar algo corto si se desean almacenar temperaturas durante largos periodos de tiempo. Su precio es de 25 dólares.

- **Pycom:**

El kit de desarrollo más barato que se puede obtener en Pycom es el Wipy. Este módulo tiene integrado el SoC Espressif ESP32, encargado de dotar al módulo con conectividad Wi-Fi y Bluetooth. Además, cuenta con un segundo procesador y con características relativamente avanzadas como 8 canales analógicos de 12 bits, 24 puertos digitales, temporizadores, ranura para tarjetas SD, 4 MB de memoria RAM y 8 MB de memoria flash. Por el precio de 20€ es un buen módulo. Sin embargo, sigue siendo excesivo en prestaciones y en precio para el diseño.

- **AI-Thinker:**

Este fabricante chino vende hasta 15 módulos diferentes basados en el SoC ESP8266. Todos estos módulos tienen características bastante similares, fundamentalmente difieren en número de pines, tamaño, cantidad de memoria flash y precio. Todos ellos tienen un precio extremadamente bajo en comparación al resto de módulos mencionados hasta ahora. Además, debido a su bajo precio, sus buenas prestaciones y su gran versatilidad, son módulos muy populares y que cuentan con un gran número de usuarios. Esto facilita la tarea de investigación y hace que resulte más fácil solucionar problemas y encontrar información y ejemplos.

Figura 3. Módulo ESP-12E

En concreto, el empleado en el diseño, es el módulo ESP-12E. Este módulo se puede encontrar por poco más de un euro, 1,21€ (el más barato de todos con entrada analógica). Este módulo encaja perfectamente en el diseño, es barato, pequeño y cuenta con los recursos necesarios para desarrollar la red de sensores sin ningún tipo de problema [4]. Además, todos los módulos de AI-Thinker se pueden programar en el entorno de desarrollo de Arduino y cuentan con gran cantidad de librerías que facilitan mucho el trabajo. Más adelante, en el siguiente capítulo se discutirán en profundidad las especificaciones, las posibilidades y el hardware de estos módulos.

- **SparkFun:**

SparkFun también ofrece módulos basados en el SoC ESP8266:

Figura 4. Placa SparkFun ESP8266 Thing

El SparkFun ESP8266 Thing [5] es prácticamente igual al ESP-01, pero prácticamente quintuplica su precio. Además, no cuenta con ninguna entrada analógica. SpakFun también ofrece una placa de desarrollo por 16,95€, la cual tampoco puede competir con el económico precio de la placa NodeMCU.

- **Adafruit:**

El fabricante Adafruit ofrece una placa de desarrollo con un módulo Wi-Fi basado en el SoC ESP8266. Sin embargo, su precio es algo elevado:

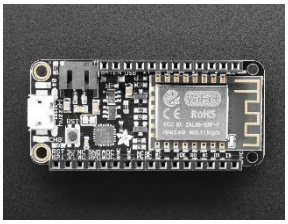


Figura 5. Placa Feather Hluzzah ESP8266

La placa de desarrollo Adafruit Feather HUZZAH ESP8266 [6] cuenta con un módulo esp-12E integrado y con unas características muy similares a la placa de desarrollo NodeMCU usada en el proyecto. Su precio es bastante elevado en comparación con el precio del módulo elegido, 18,56€.

- **NodeMCU:**

La placa NodeMCU [7] es una placa de desarrollo de hardware abierto y por lo tanto no existe un único fabricante. Esta placa se puede encontrar hasta por 2€, un precio muchísimo menor que cualquiera de los vistos anteriormente en las placas de desarrollo. Existen 3 versiones de esta placa de desarrollo, todas prácticamente iguales. En el proyecto se emplea la tercera versión, es decir, la más moderna.

Figura 6. Placa NodeMCU

Esta placa de desarrollo incluye un módulo ESP-12E y toda la circuitería necesaria para programarla directamente desde el entorno de desarrollo de Arduino mediante un cable USB. Esto simplifica enormemente el proceso de programación del módulo Wi-Fi y permite hacer cambios y pruebas en el firmware con mucha más agilidad.

- **Wemos:**

Wemos ofrece también 3 placas de desarrollo basadas en el SoC ESP8266 a un precio muy económico, pero sí algo superior al de la NodeMCU. Entre estas placas destaca la Wemos mini D1 pro [8]:

Figura 7. Palca Wemos mini D1 pro

Esta placa de desarrollo es una de las que dispone de más memoria RAM de todas las basadas en un SoC ESP8266. Sus 16MB de flash la hacen perfecta para almacenar grandes cantidades de datos. Si fuese este un requisito del proyecto, haciendo pequeños cambios, se podría incluir este módulo en el nodo central. Su precio ronda los 3€ por lo que se adapta al presupuesto del proyecto.

Como se ha mencionado, por su económico precio (1,21 €) y por contar con unas características adecuadas para el diseño, el módulo elegido es el módulo Wi-Fi ESP-12E.

2.3 Elección del sensor de temperatura

En esta sección se repasan los diferentes tipos de sensores de temperatura existentes en el mercado y se justifica la elección del termistor NTC como transductor. [9]

- **RTD o detectores de temperatura resistivos:** Los RTD son conductores cuya resistencia aumenta con la temperatura. Las más empleadas están hechas de platino. La variación de la resistencia con la temperatura es lineal pero bastante pequeña, por lo que su sensibilidad es bastante baja.

Entre las ventajas de los RTD se encuentra su alta repetibiliad, estabilidad y exactitud. El rango de temperaturas que puede medir es amplio, de cientos de grados. Entre sus desventajas se encuentra su poca sensibilidad y su precio algo elevado, que puede ascender hasta las decenas de euros.

- **Termopares:** Los termopares son sensores generadores, es decir, que a partir de una magnitud física genera una magnitud eléctrica sin requerir alimentación eléctrica. El termopar consiste en dos conductores diferentes unidos en los extremos. Estos conductores generan una diferencia de tensión entre sus extremos proporcional a la diferencia de temperatura entre ambos conductores.

  Son sensores muy empleados en la industria. Son baratos, robustos y con un amplio margen de medida (-270 °C a 3000 °C). Sin embargo, tienen muy baja sensibilidad, del orden de 50 µV por °C y muy baja exactitud, de entre 0,5 y 2 °C.

- **Sensores integrados de silicio:** Los sensores de silicio se valen de la variación de la corriente que circula por las uniones p-n directamente polarizada de los semiconductores de silicio. Esta corriente varía inversamente y de forma cuasi lineal con la temperatura.

  Estos integrados no requieren circuitos externos de linealización o amplificación puesto que está todo incluido en el mismo integrado. La sensibilidad típica de estos integrados es de 10 mV por °C, que es relativamente alta, aunque puede variar dependiendo del integrado. El margen es reducido en comparación a las opciones antes descritas, este puede ser de entre unos -50 °C hasta los 150 °C. La precisión y el precio varía mucho entre distintos integrados.

- **Termistores NTC y PTC:** Son resistores cuya resistencia varía negativamente (*Negative Temperature Coeficient*) o positivamente (*Positive Temperature Coeficient*) con la temperatura, pero de forma no lineal.

  Puesto que la variación de la resistencia con la temperatura no es lineal, es necesario linealizarla añadiendo una resistencia en serie. Puesto que la zona lineal es normalmente bastante limitada (a unas pocas decenas de grados), su margen es el más pequeño, si bien resulta suficiente para realizar medidas de temperatura ambiente en entornos domésticos. Además, su precisión no es alta. Por el otro lado, son muy baratos y muy sensibles.

Para elegir una de entre todas las opciones expuestas anteriormente, se atiende principalmente al criterio económico. Los termistores NTC y PTC son los más baratos. Además, por ser los más sensibles, no es necesario añadir una etapa de acondicionamiento. El hecho de que su precisión sea pequeña y el rango sea limitado, no es un problema teniendo en cuenta los objetivos del diseño.

Por lo tanto, se ha seleccionado un termistor NTC para este proyecto.

3 Diseño del hardware

En este apartado se va a analizar el hardware y las características del módulo y de la placa de desarrollo elegidos, así como todo lo relativo al diseño del hardware de ambos nodos: el sensor, el circuito de carga de baterías, el diseño de las placas de circuito impreso, etc.

3.1 Módulo ESP-12E y placa de desarrollo NodeMCU. Hardware y prestaciones

Como ya se ha mencionado antes, la placa de desarrollo es una placa de hardware libre y por lo tanto hay varias empresas que la fabrican. En la placa podemos distinguir 3 niveles [10]:

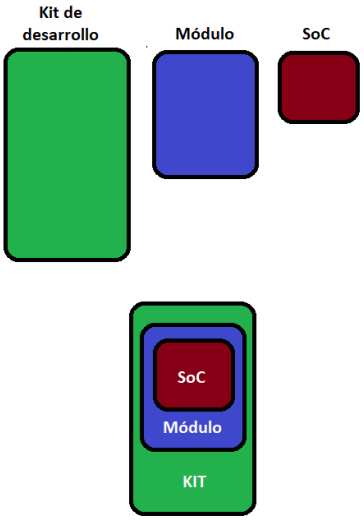


Figura 8. Esquema hardware placa NodeMCU

El primer nivel es el SoC ESP8266. Las siglas SoC vienen del inglés *System On a Chip*, es decir, sistema en chip si lo traducimos al español. SoC describe la tendencia cada vez más común de integrar todos los módulos de un sistema en un solo chip, de forma que se reducen costes y espacio. En realidad, la denominación SoC es una denominación muy genérica que podría dársele a gran cantidad de chips hoy en día.

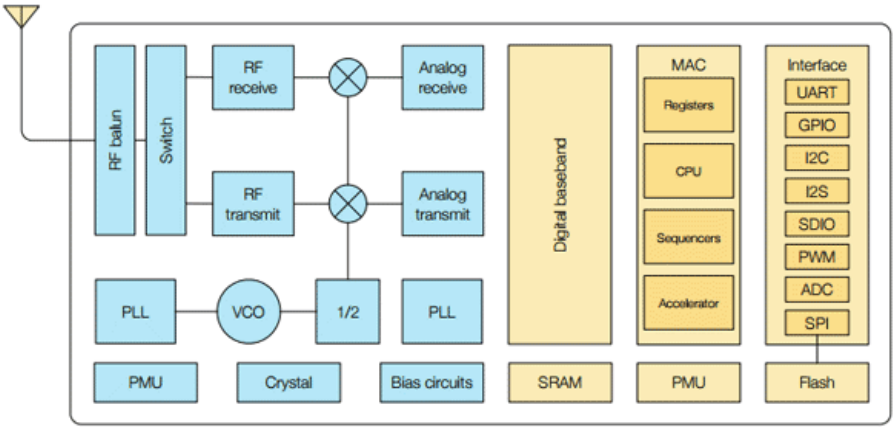


Figura 9. Esquema de bloques SoC ESP8266

Este es el diagrama de bloques del SoC ESP8266 extraído directamente de su hoja de características. En el lado derecho, pintados de color amarillo, se encuentran todos los bloques relacionados con el MCU (microcontroller unit) o microconctrolador en español.

El ESP8266 integra un procesador **Tensilica L106 de 32 bits** y de bajo consumo que puede funcionar hasta a **160 MHz** (en el diseño funciona a 80 MHz). Sólo un 20% de la capacidad de computo del procesador se destina a tareas relacionadas con la gestión de las comunicaciones Wi-Fi, por lo que la capacidad restante es más que suficiente para ejecutar el firmware del proyecto.

El ESP8266 cuenta con **50 KB de memoria RAM**, nuevamente más que suficiente para los requisitos del proyecto. Por otro lado, no cuenta con memoria flash, por lo que se debe conectar una externa mediante el puerto SPI.

De entre todos los módulos que se encuentran en la interfaz, se utilizan los siguientes: se hace uso del **UART,** para programar el módulo y depurar errores; de dos **GPIOs** (entradas salidas de propósito general) en el nodo central, para controlar el encendido de dos leds y del **ADC,** para transformar la tensión del sensor de temperatura.

En la parte izquierda del diagrama de bloques se ven los bloques encargados de gestionar la comunicación Wi-Fi. Estos bloques son transparentes a la hora de realizar el diseño.

En el caso del **consumo**, el dato más relevante para nuestro proyecto es el consumo de corriente cuando el ESP8266 se encuentra en el modo *deep-sleep*. Este modo es en el que el ESP8266 de los nodos periféricos, alimentados con baterías, permanece la mayor parte del tiempo. En el caso del nodo central, este está conectado a la red, por lo tanto, el consumo no es tan relevante. En la siguiente imagen sacada de la hoja de características del SoC ESP8266, se pueden ver los diferentes consumos en los diferentes modos.

Power Mode	Description	Power Consumption
Active (RF working)	Wi-Fi TX packet	Please refer to 5-2.
	Wi-Fi RX packet	
Modem-sleep①	CPU is working	15 mA
Light-sleep②	-	0.9 mA
Deep-sleep③	Only RTC is working	20 uA
Shut down	-	0.5 uA

Figura 10. Consumo del SoC ESP8266

Cuándo el ESP8266 se encuentra en ***deep-sleep***, este solo consume **20 uA**, un consumo adecuado para los objetivos del proyecto. Más adelante se profundizará en el tema del consumo.

Comprobamos que las prestaciones que brinda el SoC ESP8266 son más que suficientes para desarrollar la red de sensores, tanto a nivel de potencia de procesador, como de periféricos, como de consumo.

El segundo nivel es el módulo ESP-12E. Como ya se mencionó antes, existen infinidad de módulos basados en el SoC ESP8266. Estos módulos básicamente se encargan de integrar el SoC ESP8266 con una antena y memoria flash externas. También, dependiendo del módulo, se tiene acceso a más o menos pines del SoC.

Por ejemplo, el módulo ESP-01 de AI-Thinker da acceso únicamente a 8 pines, contando entre ellos los pines de alimentación y "enable", y dejando solo dos pines de propósito general a disposición del usuario. Por el contrario, en el módulo ESP-12E, es posible acceder hasta a 22 pines del ESP8266.

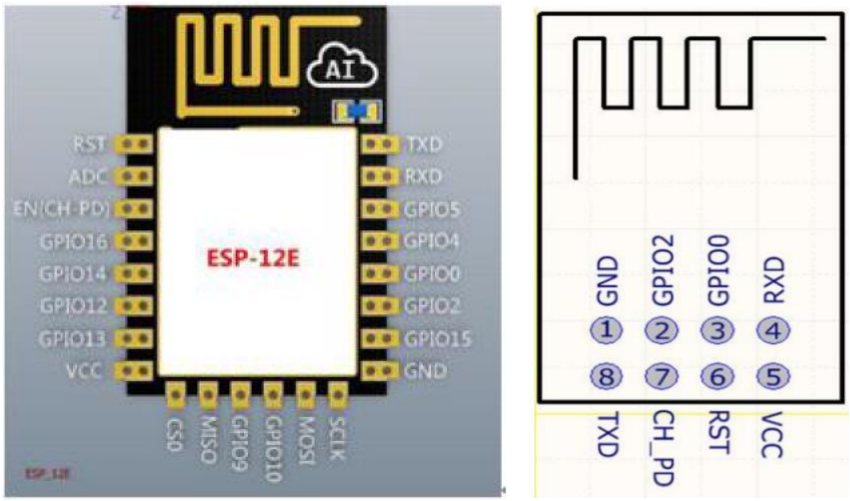


Figura 11. Esquema de pines del módulo ESP-12E y del módulo ESP-01

Entre todos los módulos basados en el SoC ESP8266, el módulo ESP-12E es el que mejor se ajusta al diseño por varias razones. En primer lugar, es uno de los más económicos si no el más. Por otro lado, cuenta con acceso al conversor analógico digital del ESP8266 y, por último, es el módulo con más memoria flash (4 MB), necesaria para almacenar el programa, la página web y los ficheros con las temperaturas medidas.

El tercer y último nivel es el kit de desarrollo. El objetivo de los kits de desarrollo es el de facilitar el prototipado y desarrollo de proyectos. Entre otras cosas, la placa de desarrollo proporciona fácil acceso a los pines, conversor serie-USB para poder programar y alimentar el módulo, pines de alimentación, leds de estado o botón de reset.

3.2 Diseño del sensor de temperatura

Como se ha mencionado antes, el transductor elegido es un termistor NTC (componente electrónico cuya resistencia varía con la temperatura). En esta sección no se van a explicar los fundamentos del funcionamiento de los NTC que se puede encontrar en [11]. Por el contrario, en esta sección se justifica la elección de los componentes del sensor y se calcula la precisión y el error del sistema.

A la hora de diseñar el sensor, se deben tener en cuenta varios factores: el consumo de corriente, el precio de los componentes empleados, el rango del sensor y su precisión. Puesto que el uso del sensor es doméstico, no es necesario que el rango del sensor sea muy amplio. En cuanto a la precisión, el sensor tampoco debe ser excesivamente preciso, pero sin ser su error máximo mayor que un grado.

La fórmula que determina el valor de la resistencia del termistor NTC es la siguiente:

$$R = R_0 e^{\beta(\frac{1}{T}-\frac{1}{T_0})}$$

Ecuación 1. Resistencia del termistor NTC

Donde: R_0 es el valor de la resistencia del termistor NTC a temperatura ambiente (298 K o lo que es lo mismo 25 °C); T es la temperatura del termistor NTC expresada en kelvin; T_0 es la temperatura ambiente expresada en kelvin y β es una constante que depende del propio termistor.

Para minimizar el consumo del sensor, tanto el termistor NTC, como la resistencia que se coloque en serie, deben ser relativamente grandes. Se elige un termistor NTC con una resistencia de 100 KΩ a 25 °C y una tolerancia del 1% y una resistencia de 75KΩ con una tolerancia del 1%. Con estos componentes, la corriente consumida por el sensor a 25 °C es 8,6 µA.

Cuanto más grande es el parámetro β del NTC, mayor es su sensibilidad y menor la zona en la que la tensión varía linealmente con la temperatura. Nos interesa que el sensor sea lineal entre los 15 y 35 °C aproximadamente. Por lo tanto, se escoge un termistor NTC con el parámetro β de 4450 K con una variación máxima del 1%.

El conversor ADC del ESP-12E mide tensiones entre 0 y 1 V. Si se quisiera obtener una resolución elevada, sería conveniente acondicionar la señal del sensor para ajustarlo exactamente a este intervalo. Puesto que la precisión no es excesivamente importante y el resultado debe ser lo más barato posible, se prescinde de la etapa de acondicionamiento.

Ya que el módulo ESP-12E está alimentado a 3.3V, y es necesario que la señal de tensión del sensor en la zona lineal se encuentre entre 0 y 1 V, se añade una referencia de tensión de 1,5V con una precisión del 0,5%.

En la Fig. 12 se muestra el esquemático del circuito del sensor de temperatura. Aparte de los elementos ya citados, aparecen varios condensadores. Estos condensadores tienen como función conseguir una señal sin ruido, lo más limpia posible.

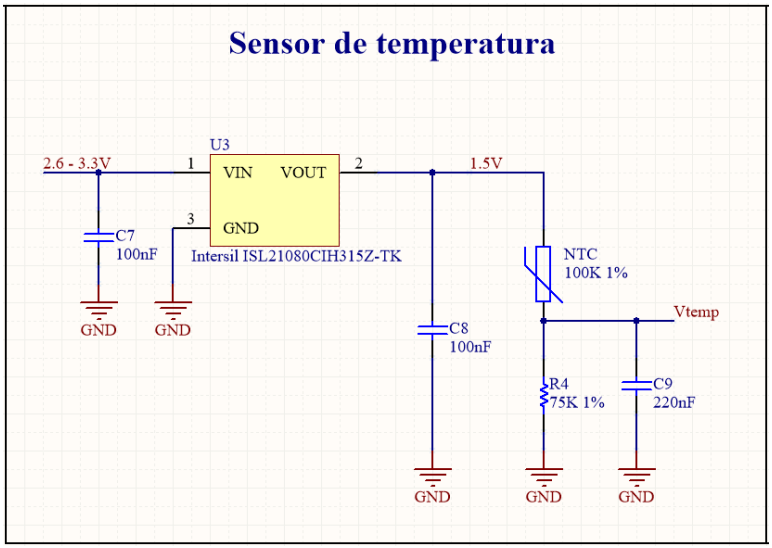


Figura 12. Esquemático del circuito del sensor de temperatura

La variación de la tensión de salida del sensor con la temperatura va a depender de la variación antes descrita de la resistencia del termistor NTC. De esta forma, la ecuación que determina el valor de la tensión será:

$$V_o = V_i \frac{R}{R_{NTC}(T) + R}$$

Ecuación 2. Variación de la tensión de salida del sensor

Donde V_i es la tensión de entrada del sensor, que en el diseño es 1,5V y R es el valor de la resistencia en serie, que en el diseño es de 75 KΩ. R_{NTC} es la resistencia del termistor que es función de la temperatura.

Introduciendo estas ecuaciones en un script de Matlab, se calcula la gráfica que se puede observar en la Fig. 13. En ella está representada la relación existente entre la tensión del sensor y la temperatura (línea roja) y la relación que existiría entre estas dos magnitudes si el sensor fuera lineal en todo su rango (línea azul). El script de Matlab se puede ver en el anexo I. Podemos apreciar que este sencillo circuito NTC y resistencia serie, linealiza bien su comportamiento entorno a temperatura ambiente.

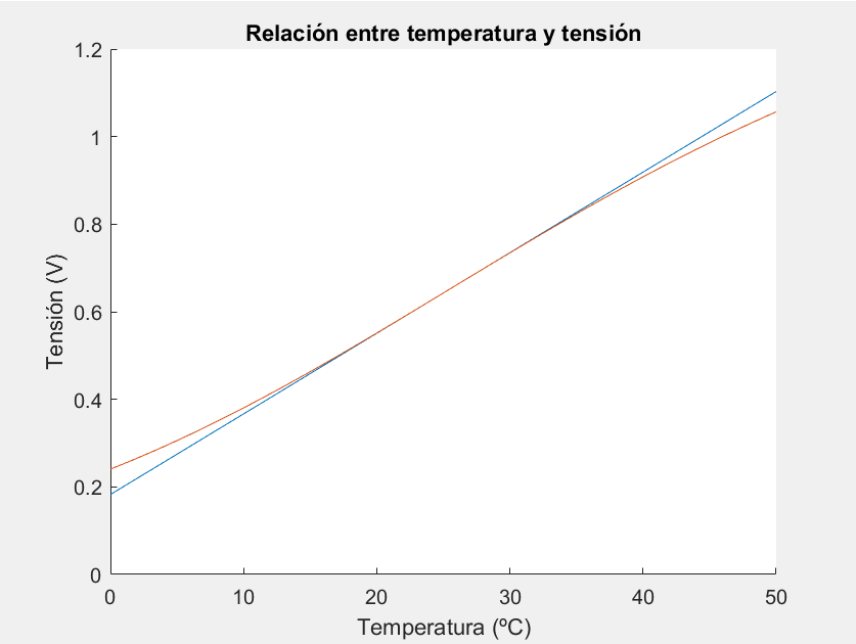


Figura 13. comportamiento del sensor

La ecuación de la recta azul, es decir, la ecuación que se implementa en el firmware para calcular la temperatura a partir de la tensión medida en el ADC es:

$$T = \frac{V - 0{,}1827}{0{,}018408}$$

Ecuación 3. Temperatura en función de la tensión del sensor

La respuesta del sensor es lineal entre los 15 y los 35 °C aproximadamente. La sensibilidad del sensor es de 1 °C por cada 0,018408 V. Con todo, este es el modelo ideal y es necesario calcular los errores en los que puede incurrir el sensor en sus mediciones.

Existen varias fuentes de error en el sistema: la tensión de referencia puede variar un 0,5% (también pierde estabilidad con el tiempo y la temperatura, pero la pérdida de precisión es tan pequeña que se puede descartar); tanto la resistencia colocada en serie como el valor de la resistencia del NTC a 25 °C tienen un 1% de tolerancia; el parámetro β del NTC también puede variar en un 1% y la referencia de tensión con el condensador de filtrado escogido introduce 150 µV de ruido según su hoja de características.

Teniendo en cuenta todas estas posibles fuentes de error, se calcula y se representa usando Matlab el máximo error posible para cada una de las temperaturas del sensor en su zona de respuesta lineal. El script de Matlab se puede ver en el anexo II.

Para realizar estos cálculos, es necesario obtener cuál es la mayor y la menor resistencia posible del termistor NTC para cada temperatura, teniendo en cuenta las variaciones del ±1% del parámetro β y de la resistencia a temperatura ambiente. Además, hay que tener en cuenta

que un error positivo del parámetro β producirá un aumento de la resistencia del termistor a menos de 25 °C y un decremento de la resistencia a temperaturas mayores de 25 °C y viceversa.

Una vez realizados los cálculos anteriores, se calcula cuál es la tensión máxima y mínima que el sensor puede dar para una misma temperatura. Para esto, se calcula la tensión de salida del divisor resistivo cuando todos los errores contribuyen a dar una tensión más baja y cuando todos los errores contribuyen a dar una tensión más alta. Una vez obtenido el error en voltios, se pasa a grados centígrados dividiendo dicho error por 0,0184 V/°C.

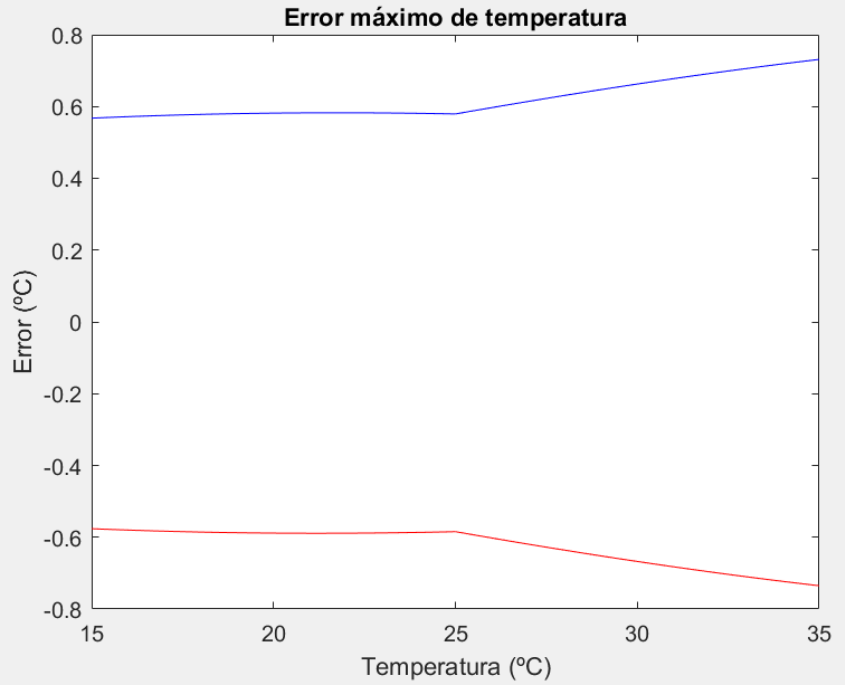


Figura 14. Error del sensor de temperatura

En la Fig. 14 están representados los errores máximos tanto positivos, como negativos que el sensor puede cometer en la medición a lo largo de su rango. Por lo tanto, de acuerdo con esta gráfica, el error de precisión máximo estimado a 15 °C es de ±0,576°C. Mientras que, en el fondo de escala, a 35 °C, el error de precisión máximo es de ±0,735 °C. A 25 °C, el error máximo es de ±0,584 °C.

El autocalentamiento del sensor es completamente despreciable. Supongamos una temperatura de 25 °C, la resistencia total del divisor resistivo sería de 175 KΩ y la potencia disipada por el termistor NTC sería de $(1,5\ V/175\ K\Omega)^2$ x 100 KΩ = 7,35 nW. El factor de disipación del termistor escogido es de aproximadamente 4 mW/K por lo que el autocalentamiento que el termistor experimentaría sería de 7,35 nW/ 4 mW/K = 1,84e-6 °C.

Otra posible fuente de errores podría ser el ruido provocado por los efectos no ideales de los circuitos (impedancias compartidas, capacidades e inductancias parásitas, ruido electromagnético…). Experimentalmente (en un entorno doméstico) se comprueba que, en el caso de los nodos periféricos, entre varias medidas consecutivas, el resultado varía a lo sumo un bit de los diez del conversor analógico digital.

El nodo central es ligeramente más sensible al ruido (probablemente porque se alimenta mediante una fuente de alimentación en vez de batería y por estar el termistor NTC fuera de la placa), por lo que, puesto que no hay ningún tipo de restricción energética, se realizan varias medidas para hacer un filtrado por software.

Probando el circuito del nodo central en una placa de pruebas se descubre que el conversor analógico digital del módulo ESP-12E no es preciso. Se detectan fallos de hasta 30 unidades sobre 1024.

Tras hacerse varias pruebas con distintos módulos, distintas alimentaciones y distintas tensiones de entrada, se llega a la conclusión de que la referencia interna del ESP-12E ofrece poca precisión. La tensión de esta referencia de tensión debería ser de un voltio, aunque en realidad en todos los casos se comprueba que es menor.

Para corregir este error, se realiza una calibración: se determina experimentalmente el valor de tensión que equivale a 1024 unidades. Ese valor es el verdadero fondo de escala del ADC que se debe sustituir por el de un voltio en la ecuación: V = ADC/1024 * fondo de escala ADC.

3.3 Consumo del nodo periférico

Para elegir una batería y poder estimar la autonomía del sensor, es necesario saber cuánta corriente consume el sistema. Para saber el consumo total, es necesario conocer tanto el consumo estático como el dinámico.

El **consumo estático** es aquel que no varía con el tiempo. En el sistema el consumo estático viene determinado por el consumo del sistema cuando el módulo ESP8266 se encuentra en modo *deep-sleep*.

Este consumo es la suma del consumo del módulo ESP8266 en *deep-sleep* (15 µA), la corriente por el divisor resistivo del sensor y diversas corrientes estáticas que consumen los distintos integrados solo por el hecho de estar activos. Este consumo es prácticamente invariable (varía ligeramente con la temperatura) y se puede medir fácilmente con el polímetro. El valor de este consumo es de 88 µA a unos 23 grados.

En segundo lugar, se necesita conocer cuál es el **consumo dinámico**, es decir, cuál es el consumo del módulo ESP8266 mientras está activo (midiendo y enviando la temperatura). Este consumo varía fuertemente con el tiempo y por lo tanto se calcula el consumo medio durante el periodo de actividad.

Como se explica en el capítulo 4, Diseño del firmware y del software en el apartado Comunicación entre nodos, hay dos posibilidades para realizar la comunicación entre ambos módulos. Esta se puede realizar o bien por protocolo UDP o por protocolo HTTP. Puesto que con el protocolo HTTP, se comprueba que los paquetes han llegado correctamente, el tiempo que el módulo Wi-Fi permanece activo es mayor y el consumo también lo es.

Además de depender del protocolo empleado, el consumo dinámico depende fuertemente del tiempo que el nodo periférico tarda en conectarse a la red Wi-Fi creada por el nodo central. Este tiempo varía dependiendo de la intensidad de la señal de Wi-Fi. Si la intensidad de la señal es baja se pueden perder paquetes y el tiempo de conexión puede llegar

a ser de varios segundos. Si la intensidad es buena, la conexión y el envío se realizan en menos de 1 segundo.

Para estimar los distintos consumos se hace uso del osciloscopio. Con el osciloscopio se mide la tensión en una resistencia de shunt de 3,3 Ω conectada entre la masa del ESP-12E y la masa de la alimentación. Para hacer esta medición, se emplean tres resistencias de 10 Ω con el 1% de tolerancia puestas en paralelo.

Es necesario comprobar que los cambios de tensión en la entrada del ESP-12E no afectan a su correcto funcionamiento. Viendo que el funcionamiento es el adecuado, se realizan las medidas. En las siguientes imágenes podemos ver las medidas tomadas por el osciloscopio de los distintos casos

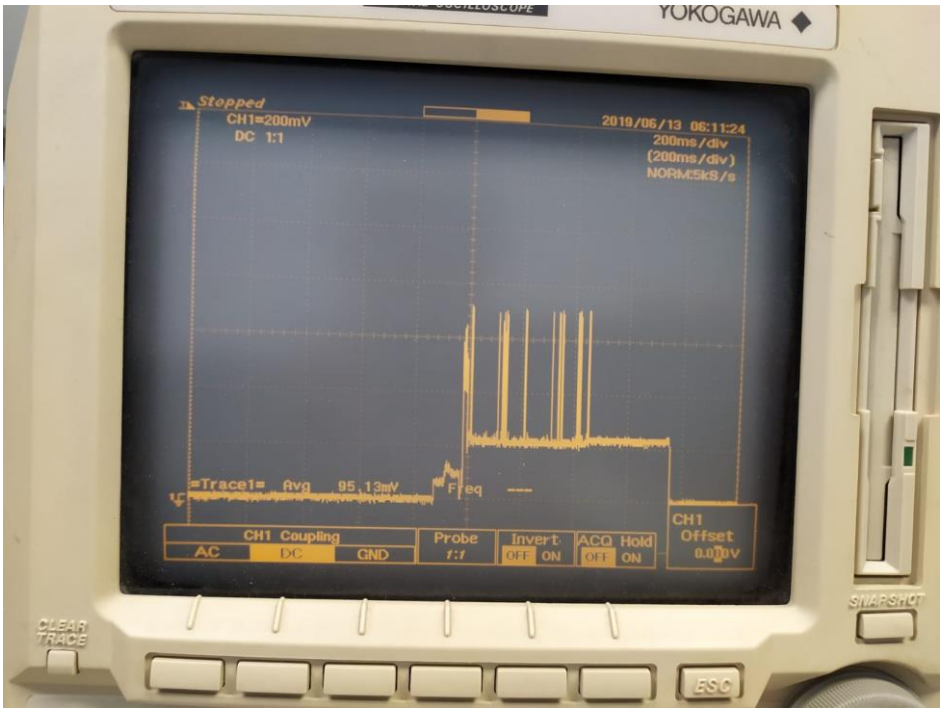


Figura 15. Consumo con los módulos próximos y usando el protocolo HTTP

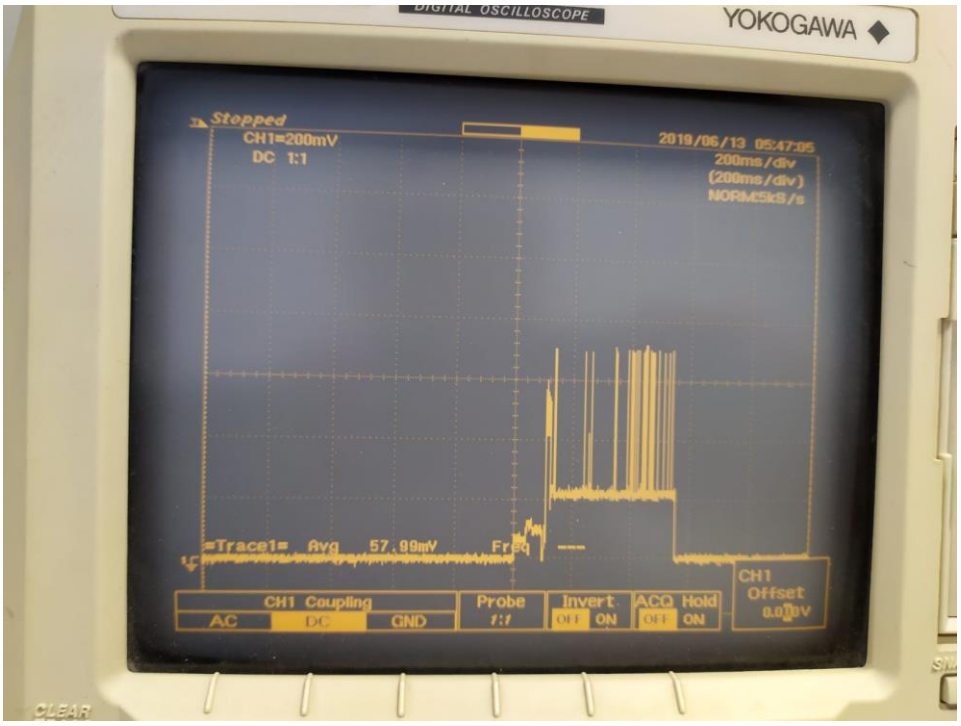


Figura 16. Consumo con los módulos próximos y usando el protocolo UDP

Figura 17. Consumo cuando el módulo no consigue establecer conexión

La tensión media varía entre las distintas mediciones incluso para unas mismas condiciones, aunque esta variación es únicamente de unos poco mV. Tomando como ejemplo

el primer caso, el de la Fig. 15, en el que ambos módulos están próximos y la comunicación se realiza mediante el protocolo HTTP, los cálculos son los siguientes:

Teniendo en cuenta que la media de tensión que muestra el osciloscopio es de 95 mV. 95 mV / 3,3 Ω = 28,79 mA. Si nos fijamos en la imagen del osciloscopio, podemos observar que cada división horizontal se corresponde a 200 ms. Por lo tanto, el ESP-12E permanece activo durante aproximadamente 0,9 segundos.

La tensión y por lo tanto la corriente media están calculadas para un intervalo de 2 segundos (10 divisiones) y, por lo tanto, podemos concluir que el consumo dinámico es equivalente a un consumo estático de 28,79 mA durante 2 segundos cada 300 segundos. A su vez, este consumo es equivalente a un consumo estático de 28,79 mA x 2 s/300 s = 191,9 µA.

Realizando el mismo cálculo para el caso de la Fig. 16, módulos próximos y realizando la comunicación mediante el protocolo UDP, el resultado es de 117 µA de consumo estático.

Cuanto más alejados estén ambos módulos, más probable es que la conexión no sea inmediata. Como vemos en la Fig. 17, cada segundo extra que el módulo tarda en conectarse a la red Wi-Fi, este consume 235 mV/3,3 Ω = 71,2 mA de corriente media.

En un intento por ver si se puede reducir el consumo de corriente del ESP-12E, se mide la corriente con el módulo funcionando a 80 y a 160 MHz. Sin embargo, la corriente consumida no varía.

La conclusión es que para una máxima autonomía es necesario que la intensidad de la señal Wi-Fi del nodo central sea suficientemente buena como para que no haya pérdida de información durante la comunicación.

Puesto que hay infinidad de variables que actúan sobre la intensidad de la señal, es realmente complicado hacer una estimación general de la autonomía del nodo periférico.

Pensando en que la autonomía del nodo periférico sea de como mínimo 100 días, se escoge una batería de una celda de iones de litio de una capacidad de 2600 mAh. Esta batería no es demasiado cara, ni grande, ni pesada y a la vez puede dotar de dicha autonomía al nodo periférico.

Suponiendo una intensidad buena y constante, la autonomía del nodo periférico se calcula dividiendo la capacidad de la batería por el consumo total. El consumo total está conformado por el estático y el dinámico. 88 µA + 192 µA = 280 µA. La autonomía es aproximadamente de 2600 mAh/ 280 µA = 9285,7 horas, es decir, casi 387 días.

En el caso de emplear el protocolo UDP, la autonomía se reduce a 2600 mAh/ 205 µA = 12672,4 horas, es decir, 528 días.

Haciendo pruebas con un nodo central y otro periférico separados aproximadamente 9 metros, se observa que el nodo periférico tarda, aproximadamente 0,3 o 0,4 segundos más en conectarse que si los módulos estuvieran uno al lado del otro. Esto significa que el consumo medio aumenta en 0,4 s * 235 mA/ 300 s = 313 µA. Si rehacemos los cálculos de la autonomía se obtiene que esta es de 2600 mAh/ (280 µA + 313 µA) = 4384,5 horas, es decir, 182 días.

Por lo tanto, mientras la distancia entre ambos nodos no sea mayor de 10 metros, la batería va superar los 100 días de autonomía casi con toda seguridad.

Si se quisiese realizar, por ejemplo, un estudio de la temperatura de una casa en diferentes puntos durante una semana únicamente, esta batería se podría sustituir por una mucha más pequeña y barata. Una batería de 200 mAh que se puede comprar por 2-3 € sería más que suficiente. 200 mAh/ 593 µA = 337,3 horas, es decir, 14 días.

3.4 Diseño de los circuitos de ambos nodos

En este apartado se justificará la elección de los componentes empleados y se explicará brevemente el diseño de los circuitos de ambos nodos.

3.4.1 Circuito del nodo central

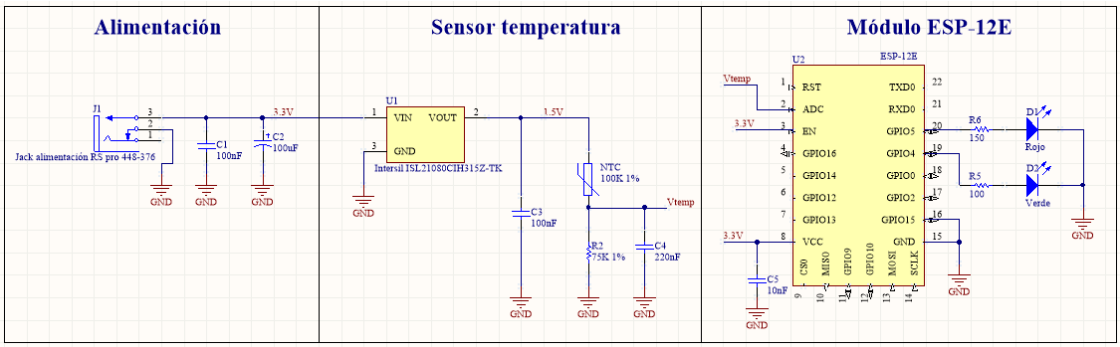


Figura 18. Esquemático del circuito del nodo central

En la Fig. 18. Se muestra el esquemático del circuito del **nodo central**, creado con el programa Circuitmaker.

Según la hoja de características del módulo ESP-12E, este puede alimentarse a entre 3 y 3,6 V, voltaje a partir del cual podría provocar daños irreversibles en el módulo. Experimentalmente se comprueba que la tensión mínima a la que el módulo puede funcionar correctamente es de 2,4 V. Además, la referencia de tensión funciona correctamente si esta está alimentada a una tensión de entre 2 y 5 V.

El circuito se debe alimentar con un adaptador de corriente continua que suministre una tensión de entre 2,4 y 3,6 V para que el módulo funcione, aunque lo más lógico es emplear un adaptador de 3,3 V. Además, el adaptador debe ser capaz de proporcionar como mínimo 250 mA, puesto que los picos de corriente son bastante pronunciados. Los dos condensadores de la entrada se emplean para filtrar el ruido proveniente de la red.

Como se ha explicado anteriormente, el sensor de temperatura está alimentado directamente por la tensión proveniente del adaptador. El condensador C3 filtra la mayor parte del ruido producido por la referencia de tensión, mientras que el C4 se encarga de filtrar el ruido justo antes de la entrada del ADC.

En cuanto al módulo ESP-12E, es alimentado directamente con la tensión de entrada. La entrada digital EN (enable) debe estar en alto para que el módulo funcione. El condensador en la entrada VCC sirve para filtrar el ruido y asegurar el correcto funcionamiento del módulo.

Como podemos ver en la Fig. 19, el *bootloader* del ESP-12E puede funcionar en 3 modos distintos. Puesto que no se va a emplear una SD en ningún momento, solo nos interesarán el primer modo y el segundo.

Mode	GPIO0	GPIO2	GPIO15
UART Download Mode (Programming)	0	1	0
Flash Startup (Normal)	1	1	0
SD-Card Boot	0	0	1

Figura 19. Modos del bootloader del SoC ESP8266

Si la configuración de los "GPIOs" de la tabla se corresponde con los del modo programación, el módulo no ejecuta el programa actual y escucha el puerto serie esperando una actualización del programa. Si los "GPIOs" se corresponden con el segundo modo, el módulo ESP-12E simplemente ejecuta el programa grabado en flash.

Tanto el pin RST como los pines GPIO0, GPIO2 y GPIO15 tienen una resistencia de pull-up interna. Por eso, para que nuestro ESP-12E ejecute el programa normalmente, lo único que hacemos es conectar el pin GPIO15 a masa. Además, de esta forma, si queremos reprogramar el módulo, únicamente es necesario conectar con un cable el pin GPIO0 a masa.

En los esquemáticos de las placas de desarrollo NodeMCU y Wemos d1 mini pro, hay situada una resistencia de 12 KΩ entre el GPIO15 y masa para reducir la corriente consumida por la resistencia de pull-up. Experimentalmente no se aprecia ningún cambio por lo que se decide no introducir esta resistencia en el diseño.

Al pin correspondiente al GPIO5 se conecta una resistencia de 100 Ω en serie con un led verde y al pin correspondiente al GPIO4 se conecta una resistencia de 150 Ω en serie con un led rojo. Las resistencias están elegidas de manera que circule suficiente corriente para que los leds luzcan con cierta intensidad, sin sobrepasar la corriente máxima que es capaz de suministrar una salida digital del ESP-12E, que es de 12 mA.

3.4.2 Circuito del nodo periférico

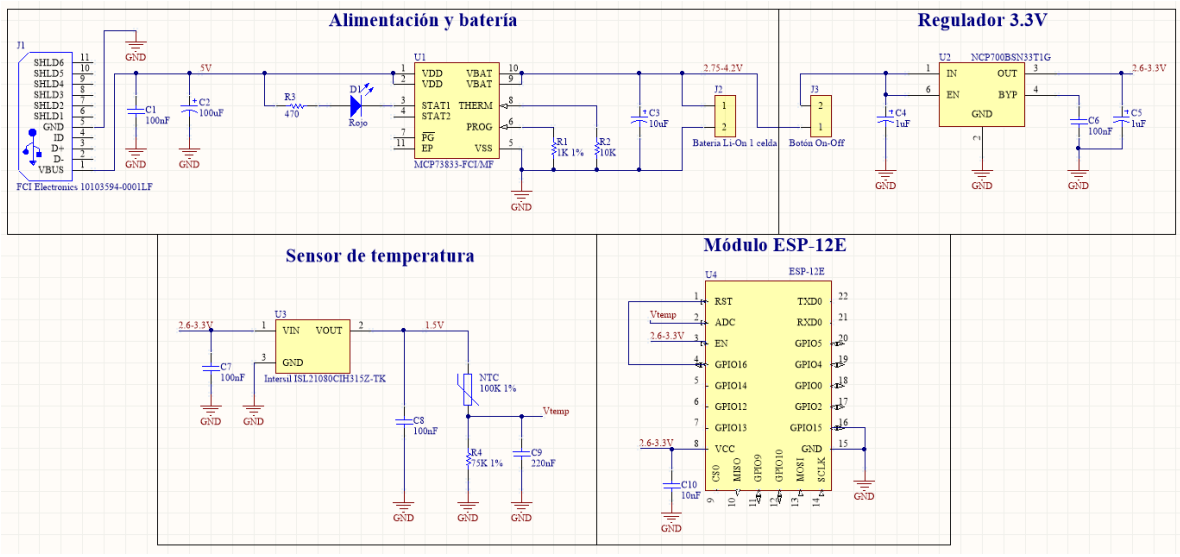


Figura 20. Esquemático del circuito del nodo periférico

En la Fig. 20 se muestra el esquemático del **nodo periférico**. El bloque del sensor de temperatura es idéntico al del nodo central. El bloque del módulo ESP-12E es igual a excepción de los LED y de la conexión entre el pin de reset (RST) y el GPIO14. Los leds se eliminan ya que implicarían un consumo excesivo. La conexión entre el pin de reset y el GPIO14 permite al ESP-12E despertar del modo *deep-sleep* cuando ha transcurrido el tiempo especificado.

La diferencia fundamental entre el módulo periférico y el central, en cuanto al esquemático se refiere, es la alimentación. Para alimentar el nodo periférico se emplea una batería de iones de litio de una celda con capacidad de 2600 mAh.

Las baterías de iones de litio deben cargarse adecuadamente para no acortar su vida. Cada ciclo de carga completo de una batería de iones de litio sigue tres fases. Podemos ver estas tres fases en la Fig. 21 sacada de la hoja de características del integrado de carga de baterías MCP73844-840I/MS.

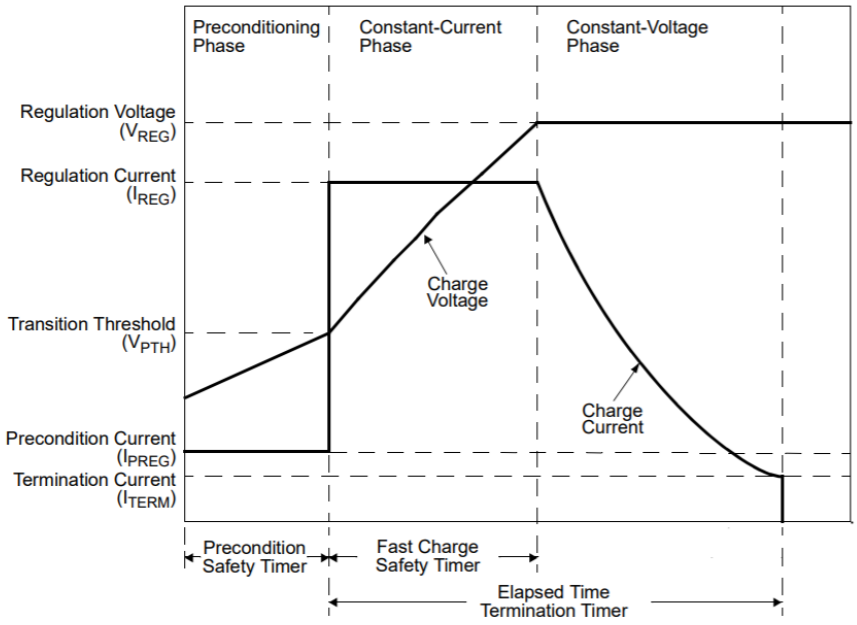


Figura 21. Ciclo de carga de una batería de iones de litio

Para asegurar que se cumplen las condiciones adecuadas en cada fase de carga, es necesario incluir en el diseño un integrado encargado de controlar la carga de la batería. El integrado que usamos en nuestro diseño es el MCP73833.

La primera fase de carga es **el preacondicionamiento**. La fase de preacondicionamiento se activa siempre que la batería está muy descargada y su tensión es inferior a la V_{PTH} o *precondition voltage threshold* (voltaje umbral de preacondicionamiento). En el caso del integrado empleado, el V_{PTH} es típicamente 2,8 V que equivale al 66,5% de V_{REG} o *regulation voltage*. V_{REG} es la tensión máxima que alcanza la batería, que en el caso de las baterías de iones de litio de una celda es 4,2V.

En la fase de preacondcionamiento, la batería se carga con corriente constante. El valor de esta corriente es una décima parte de la corriente I_{REG}, 0,1 A. La corriente I_{REG}, es la corriente con la que se carga la batería en la segunda fase o fase de corriente constante, que para este integrado es de 1 A. La segunda fase de la carga comienza cuando la tensión de la batería supera el V_{PTH}.

En la fase de **corriente constante**, la batería va aumentando su tensión mientras se carga con una la corriente constante I_{REG}. Esta fase llega a su fin cuando la tensión de la batería alcanza V_{REG} (4,2 V).

En la tercera fase, fase de **tensión constante**, la tensión se mantiene constante a valor V_{REG}, mientras que la corriente va disminuyendo hasta I_{TERM} o corriente de finalización. La corriente de finalización en el diseño es igual al 5% de I_{REG}, es decir, de 0,05 A.

El integrado de carga de baterías Li-On de 1 celda MCP73833 tiene ciertas características que lo hacen muy adecuado para el diseño. En primer lugar, es muy económico y no necesita de transistores ni de demasiados componentes externos, lo cual incrementaría el precio. Además, el integrado puede suministrar hasta 1 A de corriente, más de lo que pueden suministrar la

mayoría de integrados de un precio similar. La batería tarda en cargarse completamente aproximadamente dos horas y media.

Otra de las ventajas de este integrado es que se puede adquirir con un encapsulado DFN-10. Este encapsulado ofrece una muy baja resistencia térmica, fundamental para que el integrado de carga no alcance temperaturas elevadas. Este integrado tiene una protección contra sobrecalentamientos excesivos que podrían destruir el integrado. Está protección se encarga de limitar la corriente en función de la temperatura de la junta. En la Fig. 22, tomada de la hoja de características del integrado, se observa la relación entre la temperatura y la corriente suministrada.

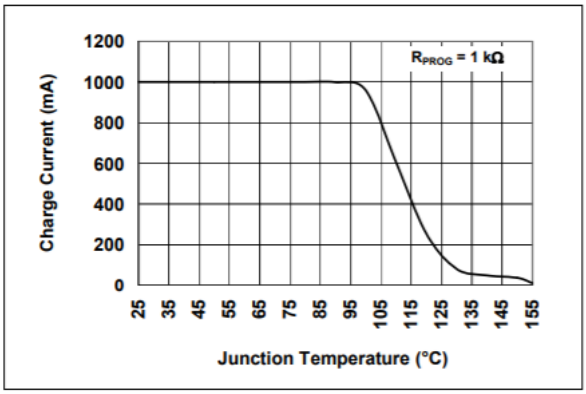


Figura 22. Variación de la corriente de carga con la temperatura

Aunque el integrado este protegido contra el sobrecalentamiento, una alta resistencia térmica produciría una ralentización de la carga de la batería por lo que es necesario asegurarse de que la temperatura del integrado no interrumpe la carga.

La potencia máxima disipada por el integrado es el producto de la caída de tensión máxima en el integrado por la corriente máxima. Esta condición se da justo en la transición entre la fase de preacondicionamiento y la fase de corriente constante:

$$P_{max} = (V_{DD} - V_{PTH})\, I_{REG}$$

Ecuación 4. Potencia máxima del integrado de carga

Siendo V_{DD} la tensión de alimentación (5 V), la potencia máxima será (5 V – 2,7 V) x 1 A = 2,3 W. La temperatura máxima que alcanza el integrado es:

$$T_{max} = T_{amb} + (\theta_{JA} * P_{max})$$

Ecuación 5. Temperatura máxima del integrado de carga

Siendo T_{amb} la temperatura ambiente que se supone de 35 °C y θ_{JA} la resistencia térmica entre la unión del integrado y el ambiente que es de 41 °C/W: La temperatura máxima es T_{max} = 35 °C + (41 °C/W * 2,3 W) = 129,3 °C.

Aunque la temperatura máxima calculada es de 129,3 °C, esto solo sería si se mantuviese la potencia máxima durante el tiempo suficiente como para alcanzar el equilibrio térmico. Puesto que la caída de tensión en el integrado va disminuyendo conforme avanza la carga, el integrado no alcanza temperaturas tan elevadas. Experimentalmente, comprobamos que ni la carga se detiene ni el integrado se calienta excesivamente durante la carga.

Volviendo al bloque de alimentación y carga del esquemático, se observa que el integrado de carga tiene 11 pines, de esos 11 pines el EP no es exactamente un pin sino la parte del integrado que se suelda al plano de masa para que disipe mejor el calor.

Los pines 1 y 2 (V_{DD}), van conectados a la alimentación que debe ser de al menos 0,3 V mayor que V_{REG}. El circuito cuenta con un conector micro USB tipo B al que se conecta un adaptador compatible con USB y que sea capaz de proporcionar al menos 1 A. Por lo tanto, la tensión de entrada del circuito es de 5 V e irá directamente conectada a los pines 1 y 2.

El integrado de carga de batería cuenta con una entrada (THERM) encargada de monitorizar la temperatura y pausar la carga si está excede ciertos límites. Para que esta característica funcione es necesario añadir dos resistencias y un termistor, y puesto que ya existe otra protección interna que limita la corriente y se ha comprobado que no existen problemas a la hora de disipar eficientemente el calor, se prescinde de ella. Siguiendo las pautas indicadas en la hoja de características, conectamos una resistencia de 10 KΩ entre el pin THERM y masa.

El pin V_{SS} va conectado a masa y entre el pin PROG y el pin V_{SS}, se debe colocar una resistencia que determina la corriente I_{REG} según la fórmula:

$$I_{REG} = \frac{1000\ V}{R_{PROG}}$$

Ecuación 6. Corriente de carga de la batería

Puesto que nos interesa que I_{REG} sea lo mayor posible, es decir, 1 A, la resistencia elegida es de 1 KΩ con el 1% de tolerancia. En la Fig. 23 se puede ver como la corriente que entra a la batería es efectivamente de casi un amperio.

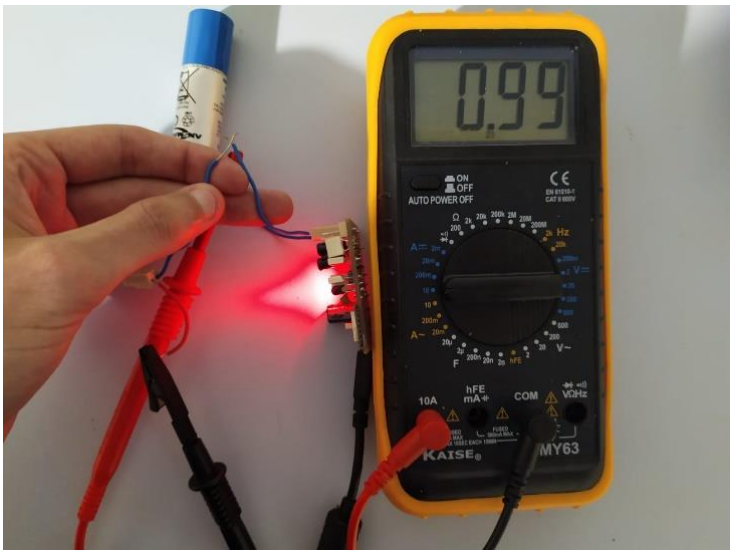


Figura 23. Medición de la corriente de carga

Los pines 9 y 10 (V_{BAT}) deben conectarse al terminal positivo de la batería que va en paralelo con un condensador electrolítico. En la hoja de características del integrado de carga se especifica que un condensador de 4,7 µF es suficiente para asegurar la estabilidad con una corriente de carga de 500 mA. Puesto que no especifica nada para corrientes superiores, elegimos un condensador con un valor algo superior, 10 µF.

Los pines STAT1, STAT2 y PG son salidas digitales. Si conectamos una resistencia en serie con un led como en STAT1, cuando la salida esté en bajo, el led se encenderá, proporcionando información acerca del estado de la batería. La salida STAT1, está en bajo siempre que la batería este cargando. En la hoja de características encontramos un diagrama de estados que nos indica también como varían estas salidas digitales (Fig. 24).

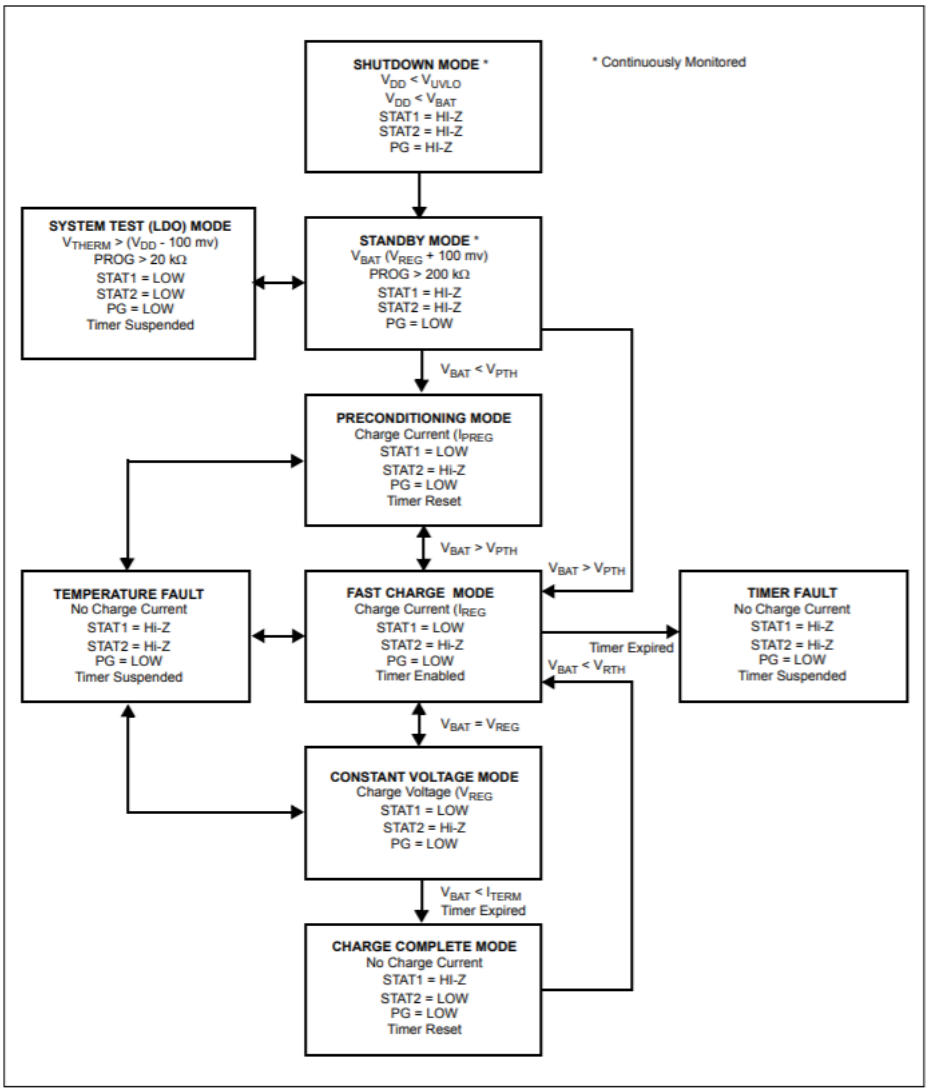


FIGURE 4-1: *Flow Chart.*

Figura 24. Diagrama de estados del integrado de carga MCP73833

En el diseño no es posible alcanzar el estado "TEMPERATURE FAULT" ni el "TIMER FAULT". El integrado proporciona un temporizador de carga que para activarlo es necesario aplicar lógica digital de nivel 1,8 V en el pin PG. Por este motivo, no se emplea esta característica.

Cuando no haya alimentación o esta sea menor que V_{UVLO} (3,4 V), el integrado entra en el modo "SHUTDOWN". Si el pin PROG está en circuito abierto o la resistencia R_{PROG} es mayor de 200 KΩ, entonces el integrado entra en modo "STANDBY". Si la batería ya cargada y conectada a la alimentación se descarga hasta ser su tensión inferior a V_{RTH} (3,95 V), entonces comienza la carga de nuevo.

El siguiente bloque en el esquemático es el **regulador lineal de 3,3 V**. Puesto que la tensión de la batería no es continua, sino que varía entre 2,75 y 4,2 V según su carga, es necesario adaptar esta tensión para que en ningún momento sea superior a 3,6 V. Para conseguirlo, empleamos un regulador lineal de bajo *dropout*. Además, en este bloque, y justo después de la batería, se sitúa el conector correspondiente al botón de encendido y apagado.

El regulador empleado, el NCP700BSN33T1G, se elige por su precio y por su bajo *dropout* (110 mV suministrando 200 mA). Puesto que experimentalmente se ha comprobado que el módulo ESP-12E funciona correctamente hasta a 2,4V, el circuito sigue funcionando, aunque la batería se descargue. Si la batería suministra 2,75 V y caen 110 mV en el regulador, la tensión de alimentación del ESP-12E seguirá siendo mayor que 2,4 V.

Siguiendo las directrices de la hoja de características, se colocan varios condensadores para filtrar el ruido antes y después del regulador lineal. De acuerdo con las siguientes gráficas tomadas de la hoja de características (Fig. 25), el condensador situado entre el pin BYP y masa, afecta tanto al tiempo de respuesta de encendido como al ruido que es capaz de filtrar.

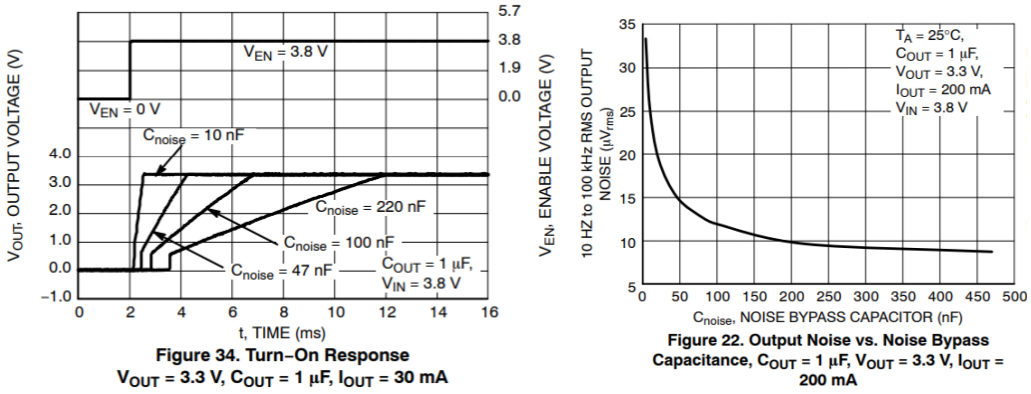


Figure 34. Turn−On Response
V_{OUT} = 3.3 V, C_{OUT} = 1 μF, I_{OUT} = 30 mA

Figure 22. Output Noise vs. Noise Bypass Capacitance, C_{OUT} = 1 μF, V_{OUT} = 3.3 V, I_{OUT} = 200 mA

Figura 25. Gráficas de la hoja de características del integrado NCP700BSN33T1G

Puesto que es más determinante en el diseño el ruido que el transitorio de encendido, usamos un condensador de 100 nF.

3.5 Diseño de las placas de circuito impreso

A la hora de plasmar los circuitos anteriores en una placa de circuito impreso, hay que tener en cuenta la incidencia de los efectos no ideales. Estos son entre otros: ruido electromagnético proveniente del exterior o del propio circuito; capacitancias, inductancias y resistencias parásitas; impedancias compartidas; etc.

Para limitar estos efectos no deseados, es necesario seguir unas normas. A la hora de diseñar las placas de circuito impreso, se han seguido las directrices y normas expresadas en [12], indicadas para la reducción de estos efectos no deseados.

A continuación, se muestran los diseños de las placas de circuito impreso realizados con Circuitmaker. En ambas placas se ha añadido plano de masa por las dos caras. El color azul se corresponde con el cobre situado en la cara inferior mientras que el color naranja se corresponde con el cobre situado en la cara superior.

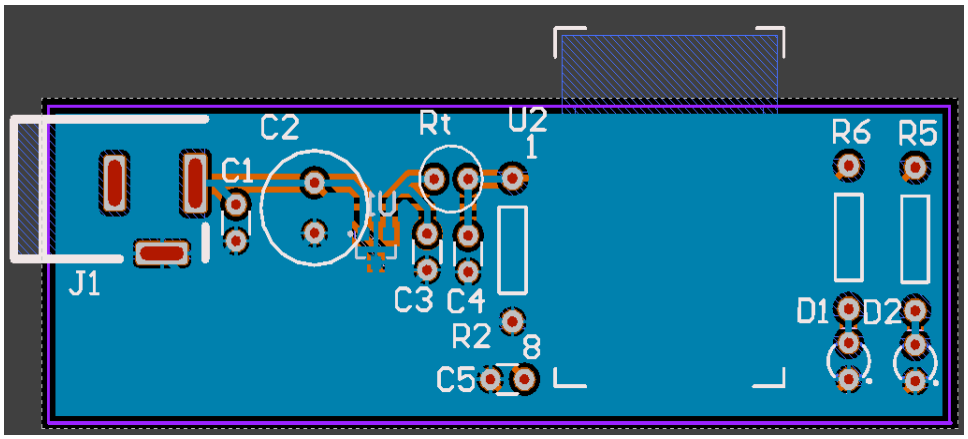


Figura 26. Cara inferior de la PCB del nodo central

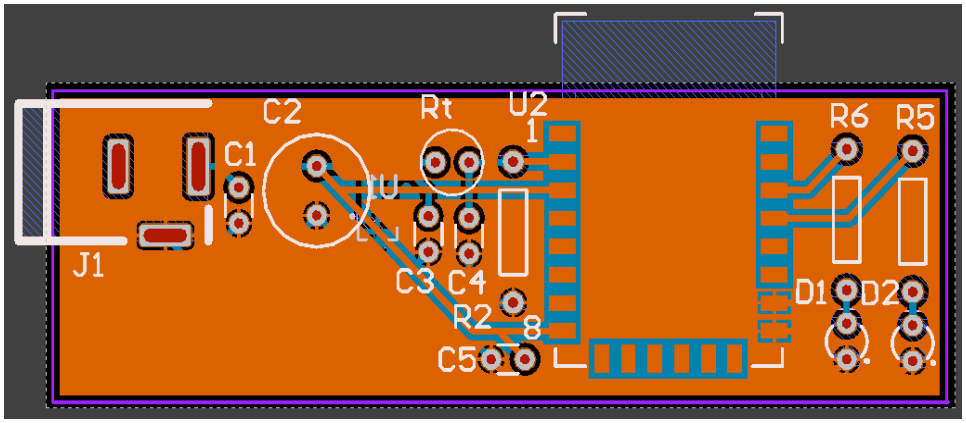


Figura 27. Cara superior de la PCB del nodo central

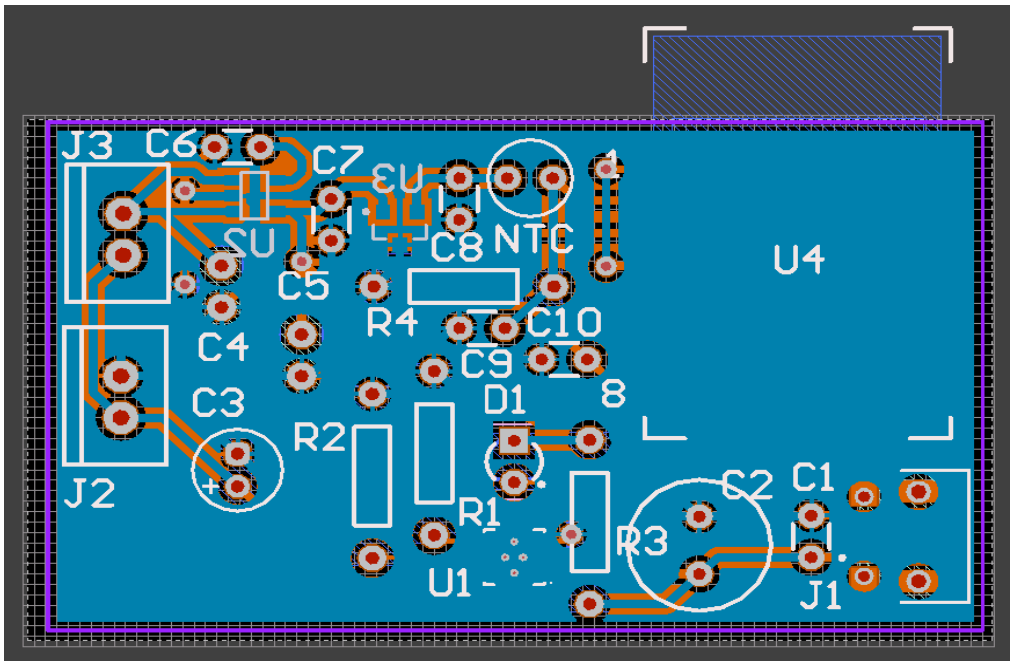


Figura 28. Cara inferior de la PCB del nodo periférico

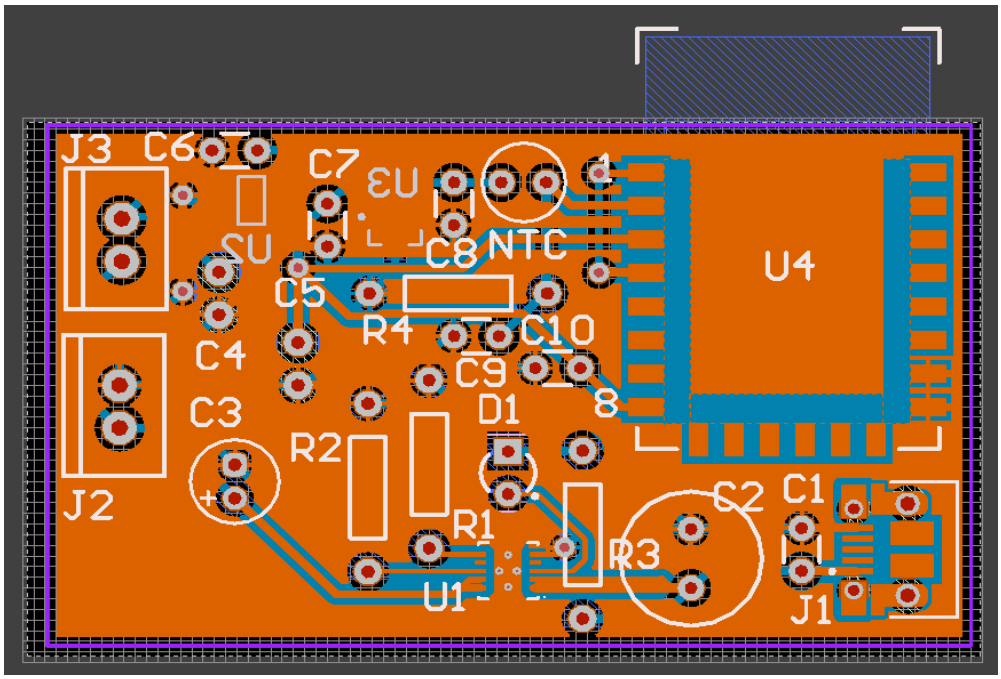


Figura 29. Cara superior de la PCB del nodo periférico

Las placas de las Fig. 26-29, son las que se han fabricado para montar los prototipos. Aunque son de pequeño tamaño, pensando en un diseño más cercano a un producto comercial, se podría reducir aún más si se incluyeran exclusivamente componentes de montaje superficial, en vez de componentes de agujeros pasantes. Para comprobar la reducción en el tamaño final de la PCB, se ha diseñado una placa con componentes SMD (Fig. 30 y 31), aunque esta no se ha fabricado.

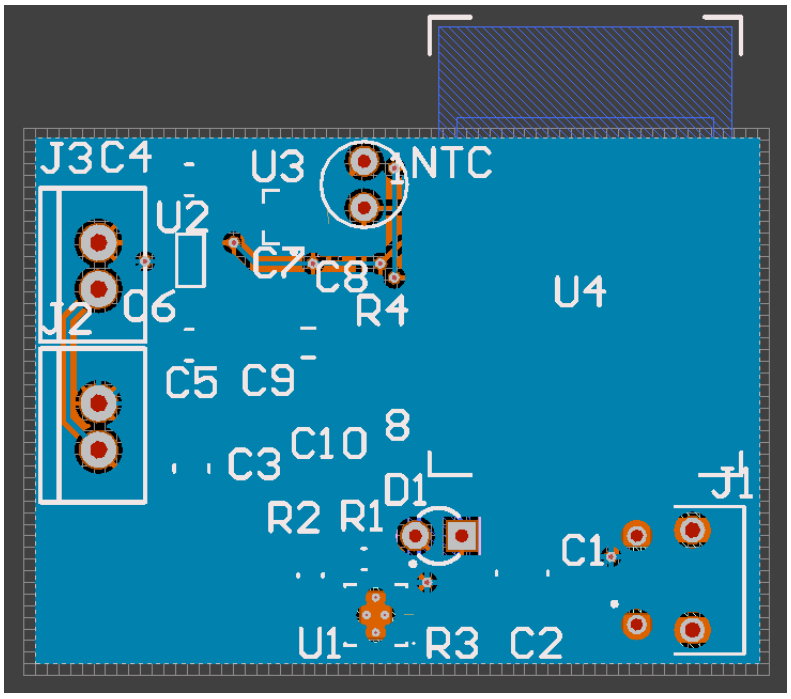


Figura 30. Cara inferior de la PCB diseñada con componentes SMD

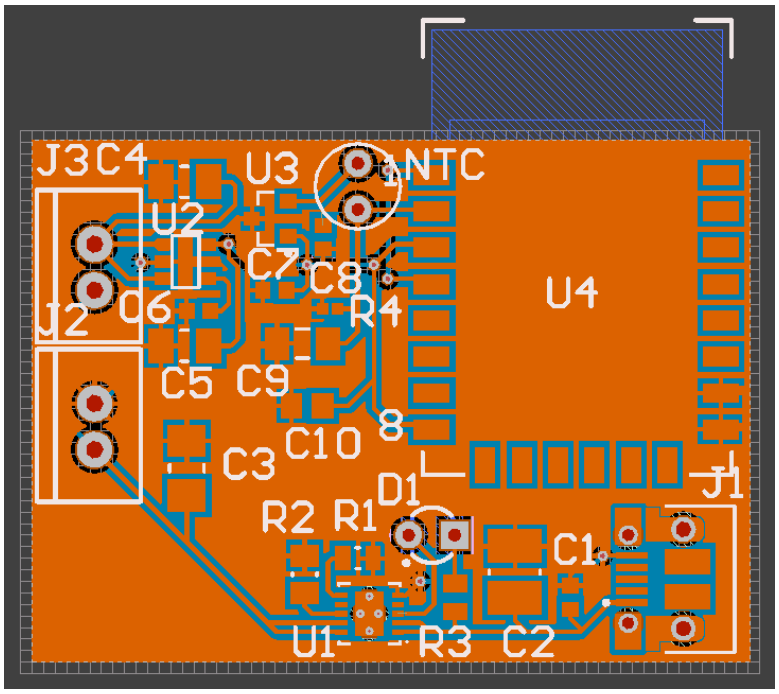


Figura 31. Cara superior de la PCB diseñada con componentes SMD

Empleando componentes SMD se pasaría de unas dimensiones de 50,868x29,913 mm a unas dimensiones de tan solo 39,497x28,702 mm.

4 Diseño del firmware y del software

En este capítulo se explica todo lo relacionado con el diseño del firmware del módulo Wi-Fi ESP-12E: la forma de almacenar y visualizar los datos, como se realizan las comunicaciones y el diseño de la página web. La programación del firmware se encuentra en los anexos III, IV y V, y el software de la página web se encuentra en los anexos VI, VII y VIII.

4.1 Módulo ESP-12E

Los módulos basados en el SoC ESP266 están integrados en el entorno de desarrollo de Arduino. Esto permite programar el módulo ESP-12E directamente desde el IDE de Arduino. El lenguaje de Arduino está basado en C++ y a efectos prácticos C++ y este lenguaje son prácticamente idénticos.

Antes de la integración en Arduino, estos módulos se programaban en LUA, un lenguaje interpretado. Desde la integración de estos módulos en el IDE de Arudino, LUA ha ido perdiendo fuerza. Investigando un poco en Internet se observa que la mayor parte de la documentación y de los ejemplos están desarrollados en este lenguaje basado en C++. Por la gran cantidad de ejemplos existentes y por la simplicidad que brinda el IDE de Arduino, se opta por usar este lenguaje para el desarrollo del firmware del proyecto.

4.2 Almacenamiento de la información

Como se mencionó al inicio del documento, en este proyecto el almacenamiento de la información se lleva a cabo siguiendo dos caminos distintos. Por un lado, las temperaturas medidas se almacenan en la memoria flash del ESP-12E del nodo central. Por el otro lado, las temperaturas se almacenan mediante un servicio de Internet de las cosas.

4.2.1 Almacenamiento en memoria flash

A la hora de almacenar las medidas de temperatura en la memoria flash, se deben de almacenar dos elementos: la temperatura medida y la fecha y hora a la que esa medida fue tomada. Para almacenar estos dos elementos se emplearán ficheros de texto que son enviados al navegador para ser descargados o representados.

La memoria flash del módulo ESP-12E disponible para almacenar dichos archivos de texto es de 3 MB a los que hay que restarles los 9954 bytes que ocupan los archivos de la página web. La fecha y la hora a la que fue tomada la temperatura se especifican en dicho archivo escribiendo los segundos transcurridos desde el 1 de enero de 1970 a las 00:00. Este sistema es conocido como tiempo UNIX.

Puesto que la memoria es relativamente reducida y la temperatura varía lentamente, se decide separar las mediciones en intervalos de 5 minutos. Ya que las medidas están tomadas cada 5 minutos y el tiempo UNIX ocupa bastante memoria en comparación con las propias medidas, dicho tiempo se escribe solo cada 12 medidas o lo que es lo mismo cada hora. El resto de fechas y horas se obtienen sumándole 5 minutos a la anterior.

Distintos elementos están escritos en distintas líneas del fichero. La temperatura se escribe en el fichero de texto con un decimal. El tiempo UNIX se escribe delante de la medida de temperatura y separado de esta mediante una coma. Además, si entre medidas transcurren más de 350 segundos o se reinicia el nodo central, se escribe una línea que únicamente contiene un espacio. En la siguiente línea siempre se escribe el elemento precedido del tiempo UNIX.

En la Fig. 32 vemos un ejemplo de cómo es un fichero:

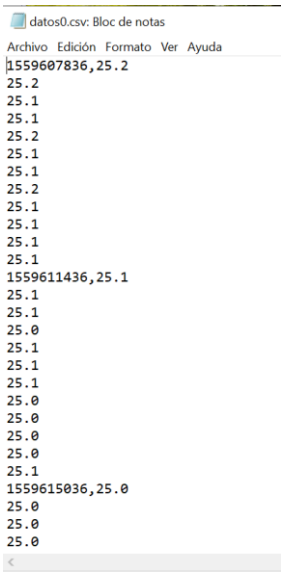


Figura 32. Ejemplo de fichero de texto

Si la comunicación se realiza mediante el protocolo UDP sujeto a posibles fallos, si la temperatura se sale de los rangos del sensor (15-35 °C), no es un número válido o varía más de 5 grados entre medidas consecutivas, se escribe en la línea una “E”.

En función de los sensores operativos, se calcula cuál es el número máximo de elementos que puede contener cada fichero. Cuando este tamaño se supere, se eliminan como mínimo los primeros 500 elementos del fichero, correspondiéndose estos con los más antiguos. Una vez eliminados los 500 elementos se siguen eliminando elementos hasta que el primer elemento del fichero contenga el tiempo UNIX.

La memoria disponible en el módulo Wi-Fi del nodo central para almacenar los archivos de texto es la siguiente: (3MB * 1024KB/MB * 1024B/KB) – 9954 bytes (ficheros página web)) = 3135774 bytes. Cada elemento del fichero sin tiempo UNIX ocupa 6 bytes, y cada elemento con tiempo UNIX ocupa 11 bytes más. De media cada elemento ocupa aproximadamente 7 bytes.

Puesto que el espacio ocupado por cada uno de los ficheros en el sistema de archivos es ligeramente mayor que su tamaño real, se reserva un 10% del espacio disponible para que no haya problemas. Por lo tanto, 3135774*0,9 bytes / 7 bytes/elemento = 403170 elementos que se pueden almacenar.

Dependiendo del número de sensores activos, se pueden almacenar temperaturas durante más o menos tiempo. Teniendo en cuenta que para eliminar datos de un fichero es necesario copiarlo primero, el número de elementos que se pueden almacenar por sensor es:

$$nº\ de\ elementos\ máximos = \frac{403170\ elementos}{nº\ de\ sensores + 1}$$

Ecuación 7. Nº de elementos máximos que se pueden almacenar por sensor

De esta forma y teniendo en cuenta que cada elemento se almacena cada 5 minutos, si tuviéramos únicamente un sensor, se podrían almacenar 201585 elementos que se corresponden con 1007927 minutos, es decir unos 700 días de muestras.

Si se tienen operativos 9 sensores, el máximo de elementos por cada sensor sería de 40317 elementos que serían unos 140 días de muestras.

4.2.2 Almacenamiento en la nube

Para almacenar los datos en la nube empleamos el servicio de Internet de las cosas ofrecido por Thinger.io. Cada 5 minutos, el módulo Wi-Fi se conecta con los servidores de Thinger.io para mandarle tanto la temperatura tomada por el nodo central como todas las temperaturas recibidas por los nodos periféricos.

Existen multitud de empresas que ofrecen servicios del internet de las cosas. Se escoge Thinger.io porque ofrece herramientas para la visualización de la información y porque es gratis si no se conectan más de dos dispositivos a sus servidores.

Esta manera de almacenar los datos simplifica mucho el proceso. Simplemente introduciendo las librerías de Thinger.io en nuestro firmware, es posible conectar el módulo Wi-Fi a sus servidores y enviar las temperaturas tomadas. Thinger.io es el encargado de gestionar y almacenar la información.

4.3 Comunicaciones

4.3.1 Comunicación entre nodos

La comunicación entre el nodo periférico y el nodo central se realiza mediante conexión Wi-Fi. El nodo central, además de estar conectado a una red Wi-Fi, genera otra a la que se conectan los modos periféricos al reiniciarse y salir del modo *deep-sleep*.

El firmware de los nodos periféricos es relativamente sencillo. Nada más reiniciarse, se encargan de medir la tensión de entrada del pin analógico. Tras obtener una medida de esta tensión comprendida entre el 0 y 1023, el módulo Wi-Fi transforma dicha medida en una tensión entre 0 y 1 V y posteriormente en una temperatura.

Habiendo obtenido la temperatura, la envía al nodo central. Para esto existen dos posibilidades. En la primera, el nodo periférico se conecta como cliente al servidor creado por el nodo central y hace una petición “GET”. En el URI de la petición se encuentra tanto la temperatura medida como el número del sensor. Para hacer la petición al servidor se emplea el protocolo HTTP mediante el cual nos aseguramos de que no hay pérdida de información.

La segunda opción es usar el protocolo UDP. Este protocolo envía la información de la temperatura y a que sensor corresponde sin hacer ninguna conexión previa y sin comprobar que

la información ha llegado correctamente. Esto puede producir la pérdida de las mediciones si la intensidad de la señal no es lo suficientemente buena. Por el otro lado, el tiempo y la energía necesarios disminuyen en comparación con el protocolo HTTP.

El nodo central está programado para recibir y almacenar las mediciones enviadas por ambos métodos. El primer carácter recibido por el nodo central se corresponde al "número de identificación del sensor" que es un numero comprendido entre el 1 y el 8. Una vez enviada la información, el módulo Wi-Fi entra en modo *deep-sleep* durante aproximadamente 300 segundos.

Elegir un modo de comunicar los datos u otro dependerá de las necesidades del usuario. Si los módulos deben estar relativamente separados o la autonomía no es de gran importancia, se empleará protocolo HTTP. Si la pérdida de ciertas medidas no es relevante, ambos módulos están próximos (buena intensidad de la señal Wi-Fi) y la autonomía es relevante, será mejor transmitir la información mediante el protocolo UDP.

A continuación, se puede observar la diferencia entre la primera gráfica de temperatura (Fig. 33) en la que los datos se han enviado mediante HTTP y la segunda gráfica de temperatura (Fig. 34) en la que los datos se han enviado mediante el protocolo UDP. En ambos casos los módulos están a una distancia de 8-9 metros aproximadamente.

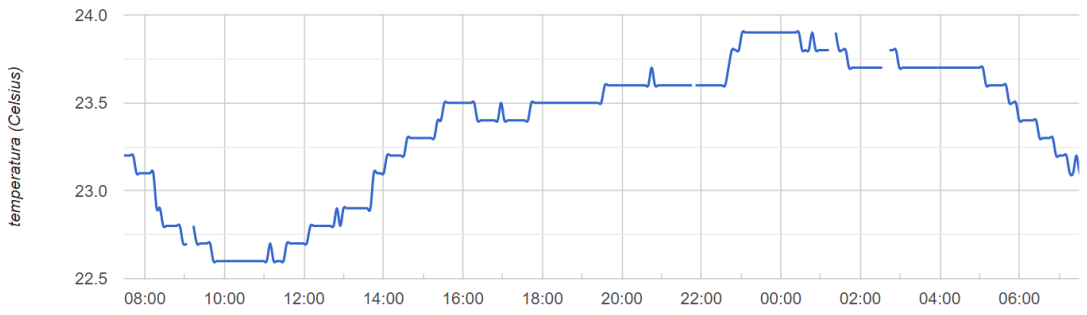


Figura 33. Gráfica de temperatura extraída de la página web (HTTP)

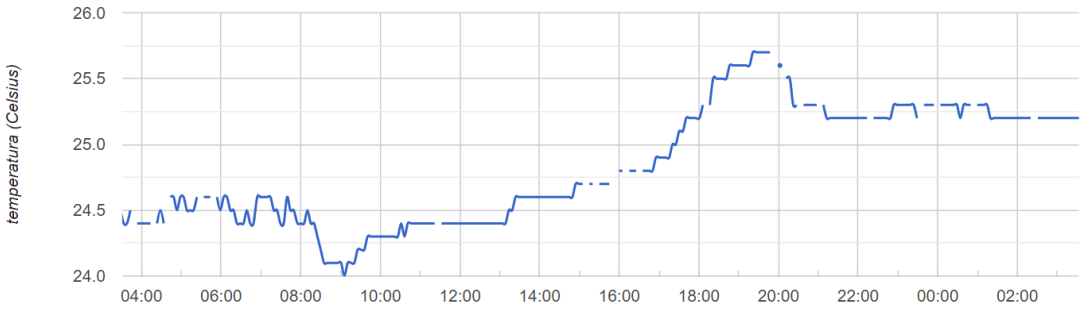


Figura 34. Gráfica de temperatura extraída de la página web (UDP)

Aunque incluso con el protocolo HTTP se pierde alguna muestra, la cantidad de muestras que se pierden es mucho menor que con el protocolo UDP.

4.3.2 Comunicación con el servidor de tiempo NTP

Para conocer el tiempo y la hora en la que se toma una medida es necesario obtener dicha información a través de Internet. Para ello, se establece una comunicación con un servidor de tiempo a través del protocolo NTP (*Network Time Protocol* o Protocolo de Hora en Red). Este es un protocolo ampliamente usado para transferir el tiempo a través de una red [13] [14].

Existen muchos servidores NTP, se ha escogido uno español por cercanía. Concretamente se ha escogido el que ofrece el Real Instituto y Observatorio de la Armada, San Fernando. Cuanto más cerca está el servidor, más rápida es la comunicación y menos probabilidades de fallos existen. Además, este servidor es "stratum 1", es decir, obtenemos la hora directamente del reloj de cesio sin intermediarios.

Para obtener la fecha y la hora, se manda un paquete de 48 bytes por el puerto correspondiente a solicitudes NTP, el 123. El valor del primer byte determina la configuración de distintos parámetros. Una vez enviado el paquete se espera una respuesta. Si dicha respuesta no llega, se vuelve a enviar una solicitud y así hasta 20 veces. En caso de no recibir ninguna respuesta, se considera que el módulo se ha desconectado e intenta volver a conectarse a la red.

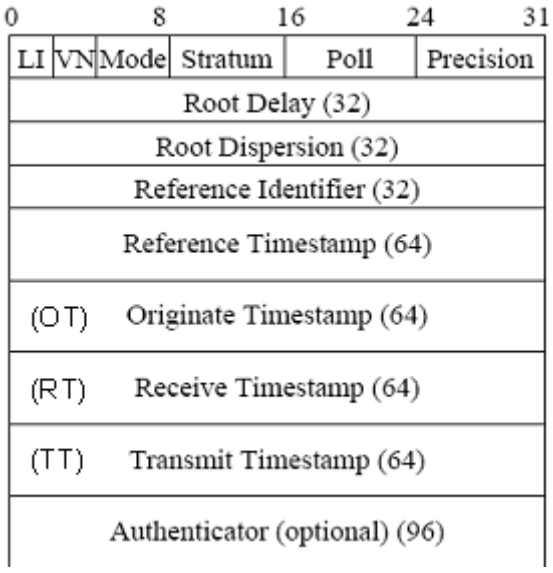


Figura 35. Esquema de campos del protocolo NTP

En la Fig. 35 [15], se puede observar un resumen de los campos del mensaje de 48 bytes del protocolo NTP. Este protocolo está pensado para enviar y recibir varias "*timestamp*" que contienen los milisegundos transcurridos desde el 1 de enero de 1900 a las 00:00 horas, para hacer cálculos y lograr que el tiempo obtenido sea muy preciso. Sin embargo, en este proyecto no es necesario obtener el tiempo con una gran precisión. Por lo tanto, todos los campos se enviarán a '0' excepto el primer byte.

En el primer byte se especifica que no se hacen modificaciones en el último minuto del día, que la versión de NTP empleada es la 1 y que el módulo es un cliente. De los 4 primeros bytes del "transmit Timestamp" se obtiene el tiempo transcurrido en segundos desde 1900.

4.3.3 Comunicación con el navegador

Para poder comunicarse con el navegador actuando como servidor, se emplea la librería “ESP8266WebServer”. Esta librería le permite al módulo Wi-Fi ESP-12E actuar como un pequeño servidor. Para conseguir esto, se crea un objeto de esta clase y al final de la función “loop” se llama al método “handleClient”. Este método nos permite atender a los clientes que están realizando una petición al servidor.

Al inicializar el servidor, se establecen funciones para responder a las distintas peticiones. Así, si se hace una solicitud “POST” con el URI "/action_cambiarNombres", el módulo ESP-12E ejecuta la función “cambiarNombres” encargada de leer los campos de los nombres y generar un fichero con los nuevos nombres de los sensores. Si se hace una solicitud “POST” con el URI "/action_borrar", se ejecuta la función “borrarFichero” encargada de eliminar el fichero que está seleccionado en ese momento en el cuadro de selección de la página web.

Si el URI de la petición no concuerda con ninguno de los dos anteriores, el ESP-12E ejecuta una función para comprobar si el URI coincide con el nombre de alguno de los ficheros almacenados en el sistema de archivos. Si coincide con alguno de los ficheros, lo envía al cliente.

Si el URI no se corresponde con ninguno de los nombres de los ficheros almacenados, el módulo Wi-Fi comprueba si el URI contiene la información correspondiente con una medida de temperatura, y de ser así, llama a una función correspondiente para almacenarla.

Si el URI tampoco contiene una medida de temperatura, entonces se responde con el código de estado 404 y se informa de que no se encuentra el archivo.

4.4 Diagrama de flujo general

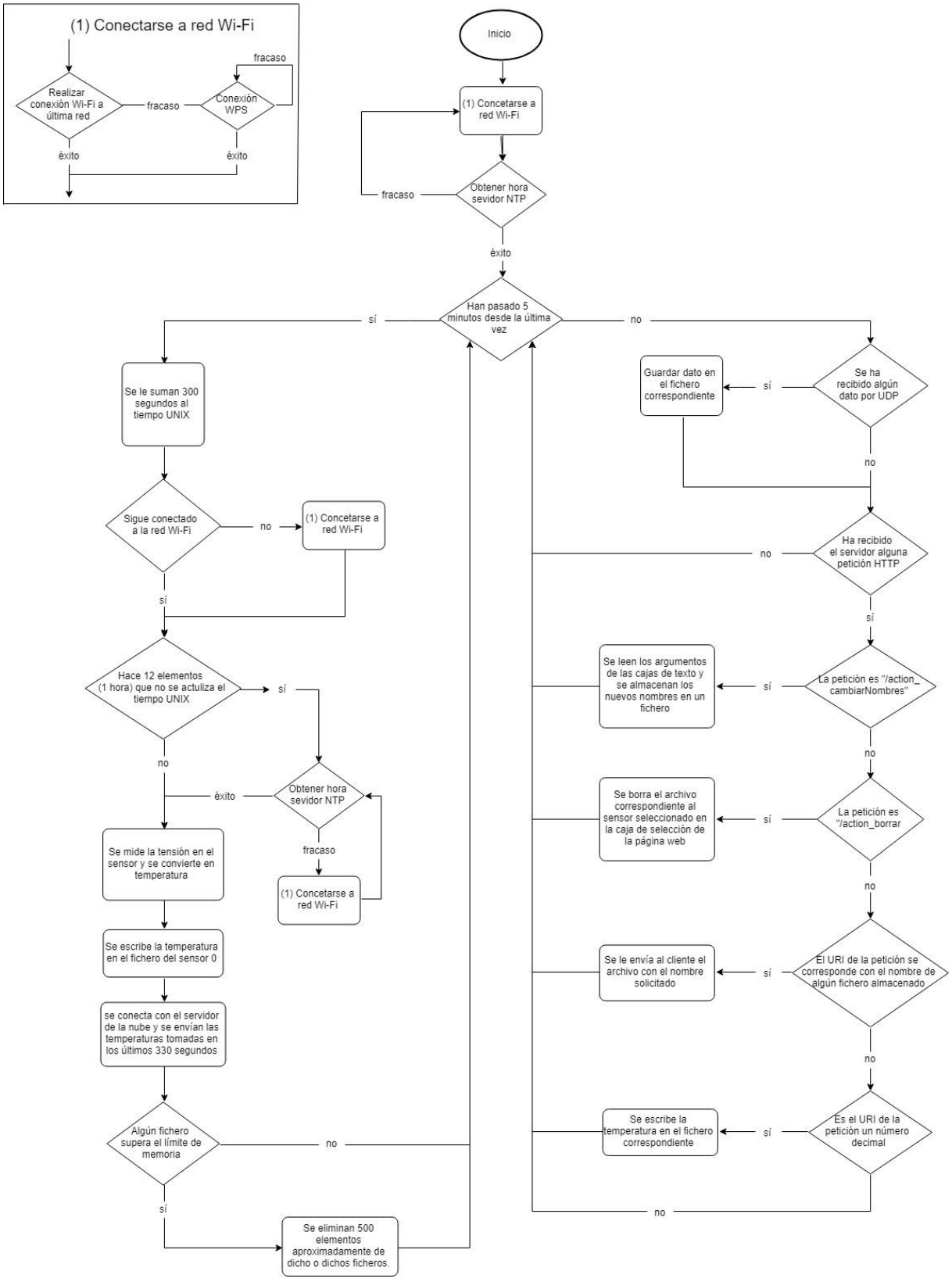


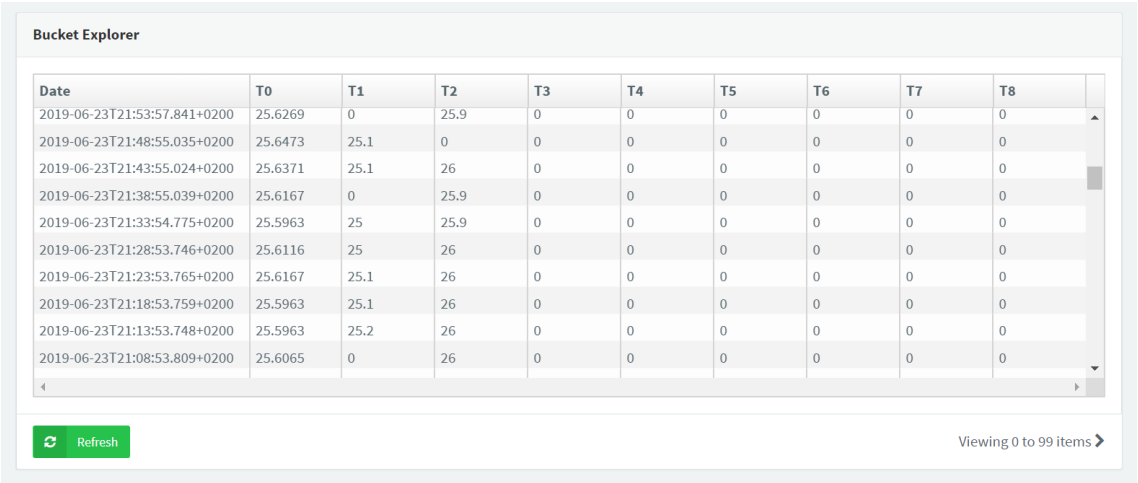
Figura 36. Diagrama de flujo del firmware del nodo central

4.5 Visualización de los datos

Podemos visualizar las temperaturas tomadas, o bien desde la nube empleando las herramientas que thinger.io nos proporciona, o bien conectándonos a la misma red que el nodo central y usando el navegador de nuestro dispositivo.

4.5.1 Visualización en la nube

Las temperaturas que toma y recibe el nodo central son enviadas a thinger.io que las almacena en lo que llaman un "*Data Bucket*". En este "cubo" quedan almacenadas todas las medidas tomadas en los últimos 330 segundos junto con la hora en la que el nodo central se conectó al servidor de thinger.io. Si el sensor no está activo el nodo central envía un 0. La Fig. 37 está tomada directamente de la página de thinger.io de un "*Data Bucket*".



Date	T0	T1	T2	T3	T4	T5	T6	T7	T8
2019-06-23T21:53:57.841+0200	25.6269	0	25.9	0	0	0	0	0	0
2019-06-23T21:48:55.035+0200	25.6473	25.1	0	0	0	0	0	0	0
2019-06-23T21:43:55.024+0200	25.6371	25.1	26	0	0	0	0	0	0
2019-06-23T21:38:55.039+0200	25.6167	0	25.9	0	0	0	0	0	0
2019-06-23T21:33:54.775+0200	25.5963	25	25.9	0	0	0	0	0	0
2019-06-23T21:28:53.746+0200	25.6116	25	26	0	0	0	0	0	0
2019-06-23T21:23:53.765+0200	25.6167	25.1	26	0	0	0	0	0	0
2019-06-23T21:18:53.759+0200	25.5963	25.1	26	0	0	0	0	0	0
2019-06-23T21:13:53.748+0200	25.5963	25.2	26	0	0	0	0	0	0
2019-06-23T21:08:53.809+0200	25.6065	0	26	0	0	0	0	0	0

Figura 37. Ejemplo de un "Data Bucket" de thinger.io

Además, como podemos observar en la Fig. 38, Thinger.io permite hacer gráficas con los datos almacenados en los cubos. Para ello, solo es necesario configurar algunos parámetros, como el intervalo temporal y el sensor del que se quieren representar las temperaturas.

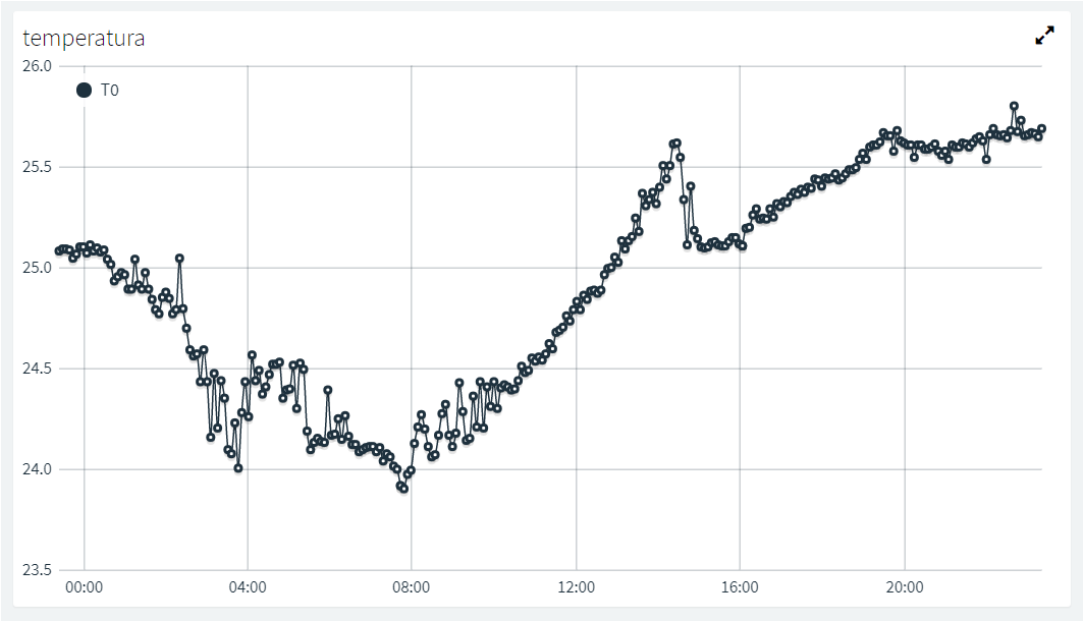


Figura 38. Gráfica de temperatura generada de un "Data Bucket"

4.5.2 Visualización en el navegador

Para visualizar las temperaturas en el navegador es necesario diseñar una página web. Para diseñar la página web se necesitan varios archivos que hay que cargar al sistema de ficheros del ESP-12E:

En primer lugar, es necesario un archivo HTML y otro CSS para describir la estructura y la apariencia de la página web principal. También es necesario otro archivo HTML definiendo la estructura de la página empleada para cambiar los nombres de los sensores.

Para la gestión de la información y la representación de los archivos de texto es necesario un archivo programado en JavaScript. Por último, se carga en el ESP-12E el fichero del icono de la página y un archivo JSON encargado de definir ese fichero como el icono de la página.

Con el nodo central conectado y funcionando, si se escribe la IP (que está configurada como una IP fija) del módulo Wi-Fi en el navegador, se envía una petición al módulo Wi-Fi que responde enviando la página web. El aspecto de la página es el siguiente (Fig. 39):

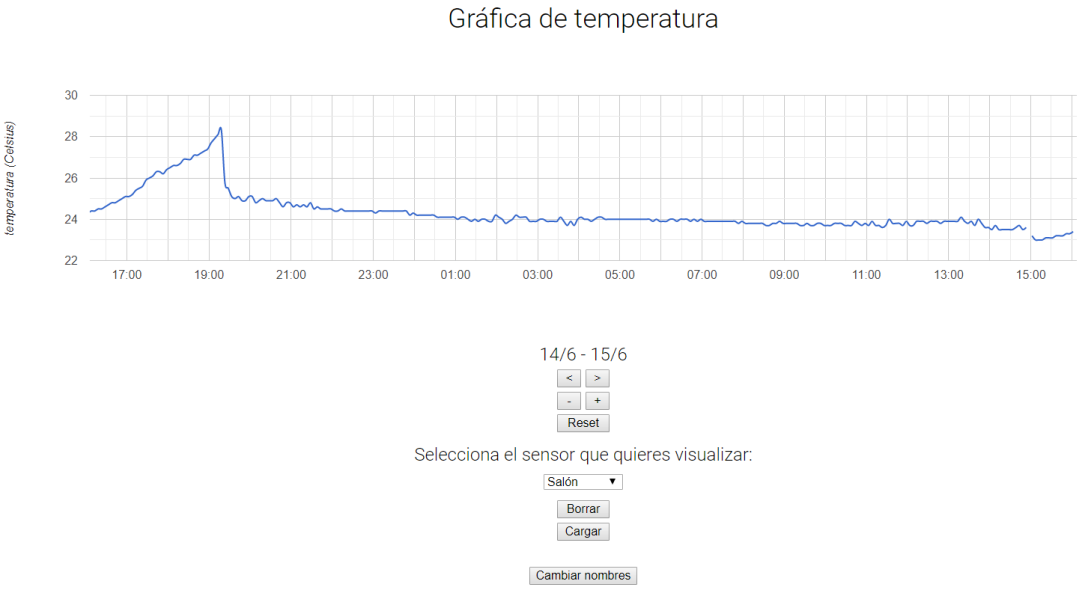


Figura 39. Vista general de la página web

La página permite visualizar cualquiera de los sensores seleccionándolo en la caja de selección y pulsando cargar. Mediante los botones "<" ">" "-" "+" se puede desplazar la gráfica o aumentar y reducir el zoom. Si se pulsa "Borrar", se elimina de la memoria del módulo Wi-Fi el fichero correspondiente al sensor seleccionado en la caja de selección. Por último, si se pulsa el botón "Cambiar nombres", el navegador es redirigido a otra página donde se pueden cambiar los nombres de los sensores que por defecto son números del 1 al 8 (Fig. 40).

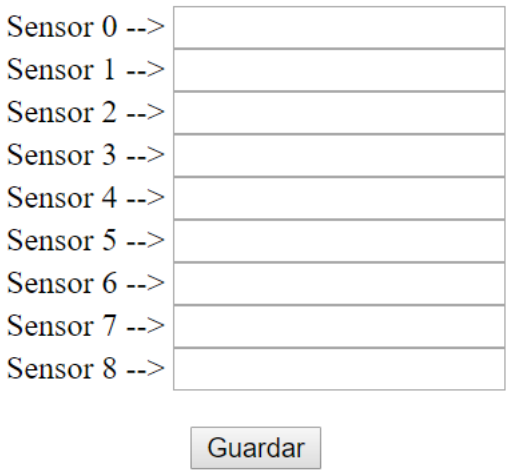


Figura 40. Interfaz de usuario para cambiar los nombres de los sensores

El diseño de la página web está basado en un proyecto encontrado en el tutorial "*A Beginner's Guide to the ESP8266*" [16]. Tomando como base el diseño de la página web del proyecto "*Temperature logger*", se realizan los cambios pertinentes para adaptar dicha página a las necesidades del proyecto.

Entre otros, los cambios realizados son: la estructura de la página, la forma de leer e interpretar los archivos de texto, la habilitación de la posibilidad de representar hasta nueve sensores distintos y la adición de las funciones de borrar y de cambiar los nombres.

5 Prototipos desarrollados

Los 3 prototipos tienen un cable soldado al pin TX, otro al RX y otro al GPIO 0. El objetivo de esto es poder programar el módulo directamente en la PCB.

A continuación, en la Fig. 41, se observa la imagen del nodo central. En el nodo central, en un inicio, el termistor NTC iba soldado directamente a la placa de circuito impreso. Se observó que por proximidad al módulo Wi-Fi la temperatura se incrementaba en 3 o 4 grados. Por este motivo se optó por conectarla a la PCB mediante dos cables.

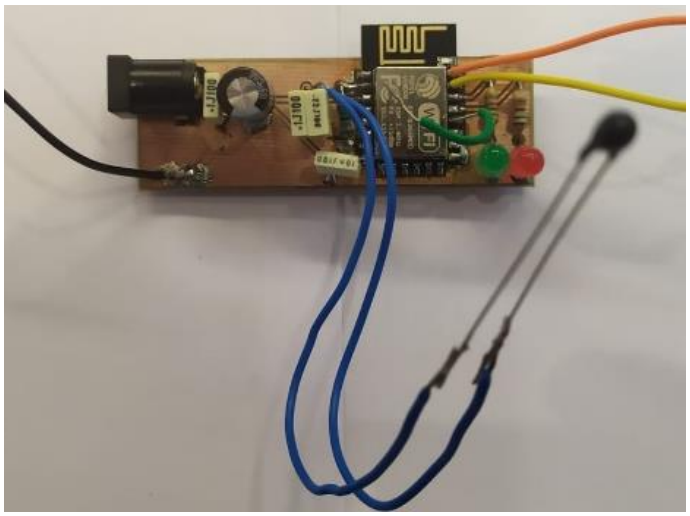


Figura 41. Prototipo del nodo central

En la Fig. 42 se observa una foto de ambos nodos periféricos:

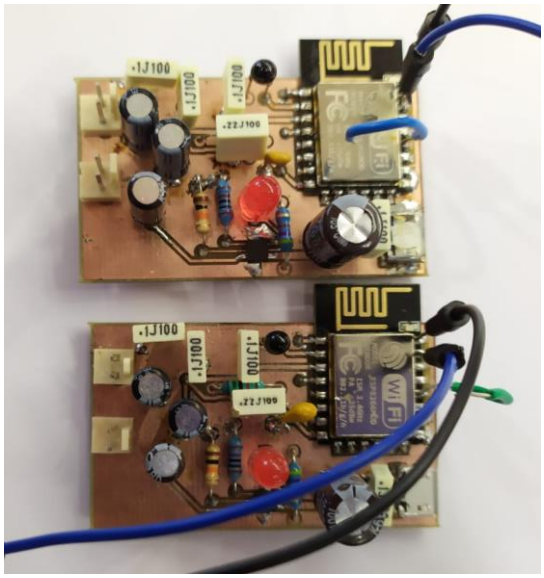


Figura 42. prototipos de los nodos periféricos

La Fig. 43 es también de uno de los nodos periféricos, pero esta vez con la batería y el botón de encendido conectados a la placa:

Figura 43. Prototipo con batería e interruptor conectados

Tras varios días de prueba queda patente que el nodo central, así como ambos nodos periféricos, funcionan correctamente.

En el anexo IX pueden encontrarse los presupuestos. A la hora de buscar componentes baratos ha sido una limitación el poder comprar únicamente a los proveedores habituales de la universidad, como RS o Farnell. Por ejemplo, la batería del proyecto, que cuesta 22 €, se podría haber encontrado por menos de 10 € buscando en otras tiendas o distribuidores.

6 Conclusiones y trabajo futuro

En este trabajo fin de grado se han desarrollado, fabricado y puesto en marcha dos circuitos IoT, un nodo central y un nodo sensor, de bajo consumo y tamaño y coste reducidos, que permiten la medida de temperaturas con almacenamiento de datos en la nube y visualización desde teléfonos móviles o computadores.

Evaluando el resultado del trabajo realizado, se puede concluir que se han cumplido los objetivos inicialmente planteados. Se ha implementado una red de sensores IoT económica, con dispositivos pequeños y con un consumo bastante reducido. Tras probar durante varios días los prototipos y comparar los resultados con los de sensores más precisos, se ha constatado que los prototipos funcionan perfectamente, proporcionando errores de medida inferiores a los inicialmente estimados

Además, se ha conseguido que el sensor sea fácilmente reproducible, con el único inconveniente de tener que calibrar manualmente los conversores analógico digitales, por la referencia de tensión poco precisa incorporada en los módulos ESP-12E.

Entre las aplicaciones posibles del proyecto, la red de sensores podría ser utilizada por un usuario preocupado por conocer la temperatura de su casa en diversas zonas, o por un equipo de profesionales que decidan realizar un estudio energético para el cual deban conocer como varía la temperatura durante un periodo de tiempo en un determinado entorno.

Dependiendo del tipo de proyecto y de la autonomía necesaria, fácilmente se podría sustituir la batería por otra capaz de almacenar más o menos energía o incluso por pilas.

En caso de ser necesario más almacenamiento, sin incurrir en muchos cambios en el diseño, se podría sustituir el módulo ESP-12E por la placa de desarrollo Wemos d1 mini pro, con hasta 15 MB disponibles de memoria flash para almacenar archivos, es decir, 5 veces más.

Entre las posibilidades de mejora, se encuentra la de conseguir reducir aún más el tamaño del nodo periférico. Esto se conseguiría escogiendo elementos SMD adecuados y buscando la disposición más eficiente de estos sobre la PCB. Otra forma de mejora podría consistir en sustituir la antena impresa en la PCB del módulo ESP-12E por una de mayor alcance y así ampliar la separación máxima entre nodos.

Conclusiones personales:

Hacer este proyecto ha requerido de mucho tiempo y esfuerzo, pero todo ese esfuerzo se ha visto compensado con el resultado final y con todo lo aprendido.

Durante el transcurso de este proyecto no ha habido ningún hito que haya representado una dificultad especial, pero cada paso ha representado un pequeño reto que superar. Desde el diseño del sensor hasta el desarrollo del firmware, todo ha requerido de investigación y aprendizaje previos que más tarde darían forma a una solución.

Este proyecto ha representado para mí una primera aproximación a la práctica de la ingeniería desde un punto de vista más realista que durante las asignaturas del grado. Lejos de parecerse a una asignatura teórica en la que el fin último es aprobar un examen, he aprendido

a buscar soluciones prácticas a problemas reales. Y lo más importante, lo he hecho por mí mismo.

Este proyecto me ha dado la oportunidad de aprender un poco más sobre un tema que creo que es importante, prácticamente para cualquier ingeniero, y del cual no existe ninguna asignatura en la carrera, las comunicaciones, y más concretamente, el Internet de las cosas. También me ha permitido poner en práctica mis conocimientos de electrónica y de instrumentación.

7 Bibliografía

[1] M. Burgess, «Wired,» 16 febrero 2018. [En línea]. Available: https://www.wired.co.uk/article/internet-of-things-what-is-explained-iot. [Último acceso: 13 abril 2019].

[2] «Techopedia,» [En línea]. Available: https://www.techopedia.com/definition/28247/internet-of-things-iot. [Último acceso: 13 mayo 2019].

[3] Adafruit, «Creative Commons,» [En línea]. Available: https://farm2.staticflickr.com/1739/27914025297_411b2b98f1_b.jpg.

[4] V. Ventura, «Polaridad.es,» 2 junio 2016. [En línea]. Available: https://polaridad.es/esp8266-modulo-wifi-elegir-caracteristicas/. [Último acceso: 23 mayo 2019].

[5] SparkFunElectronics, «Flickr,» [En línea]. Available: https://www.flickr.com/photos/sparkfun/17248860884.

[6] Adafruit, «Creative Commons,» [En línea]. Available: https://farm5.staticflickr.com/4671/25021435897_db9865b932_b.jpg.

[7] M. M. DE, «Wikimedia Commons,» 17 noviembre 2016. [En línea]. Available: https://commons.wikimedia.org/wiki/File:Nodemcu_amica_bot_02.png.

[8] Quel.soler, «Wikimedia Commons,» 16 junio 2018. [En línea]. Available: https://upload.wikimedia.org/wikipedia/commons/3/32/Wemos_D1_mini_pro.png.

[9] M. Araya, «Sensores de temperatura,» [En línea]. Available: http://snsoresdetemperatura.blogspot.com/2009/05/sensores-de-temperatura.html.

[10] L. d. V. Hernández, «Programarfacil,» [En línea]. Available: https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/. [Último acceso: 24 junio 2019].

[11] Anónimo, «Docit,» 18 julio 2016. [En línea]. Available: https://docit.tips/download/termistor-ntc_pdf.

[12] T. P. Santamaría, «38 El ruido en los sistemas digitales,» de *Electrónica Digital*, Zaragoza, Universidad De Zaragoza, pp. 171-188.

[13] «Wikipedia, Network Time Protocol,» [En línea]. Available: https://es.wikipedia.org/wiki/Network_Time_Protocol. [Último acceso: 12 junio 2019].

[14] N. T. Protocol, «Blog DAVANTEL,» 18 abril 2018. [En línea]. Available: https://blog.davantel.com/protocolo-ntp. [Último acceso: 15 junio 2019].

[15] D. exb, «Wikimedia Commons,» 2009, [En línea]. Available: https://commons.wikimedia.org/wiki/File:Format_d%27un_message_NTP_v3.png.

[16] P. P., «A Beginner's Guide to the ESP8266,» 3 agosto 2017. [En línea]. Available: https://tttapa.github.io/ESP8266/Chap01%20-%20ESP8266.html.

[17] I. Grokhotkov, «GitHub, ESP8266 Arduino Core,» 2017. [En línea]. Available: https://arduino-esp8266.readthedocs.io/en/latest/.