Universidad Nacional de San Martín

PROYECTO FINAL INTEGRADOR PARA OBTENER EL GRADO DE INGENIERO ELECTRÓNICO

MEMORIA DEL TRABAJO FINAL

Sistema de domótica

Autor:
Mauricio Bernardo Tallatta

Tutor:
Esp. Ing. Milton Eduardo Sosa

Jurados:
Ing. Patricio Dubini (ELEMON)
Ing. Marcelo Romeo (UNSAM)
Esp. Ing. Sergio Alberino (UTN)

Este trabajo fue realizado en Guadalupe Norte, Santa Fé, Argentina, entre Febrero de 2020 y Septiembre de 2022.

Resumen

En la presente memoria se describe el desarrollo e implementación de un sistema de domótica, el cual inicialmente fue ideado con comunicación NB-IoT [1]. Debido a la ausencia de señal NB-IoT en la zona de instalación, se implementó con WiFi [2] y respaldo por GPRS [3]. El sistema es capaz de informar el estado de diferentes entradas, tanto analógicas como digitales y actuar sobre salidas configurables. Se compone de un módulo central encargado de la comunicación a un portal de visualización y de módulos esclavos que gestionan las entradas y salidas de manera inalámbrica.

[1] NB-IoT es el acrónimo en ingles de *Narrow band Internet of Things*, banda angosta internet de las cosas.

[2] WiFi es el acrónimo en inglés de *Wireless Fidelity*, Fidelidad Inalámbrica.

[3] GPRS es el acrónimo en inglés de *General Packet Radio Service*, servicio general de paquetes vía radio.

Agradecimientos

En primer lugar, quiero agradecer a mi familia por el acompañamiento constante durante esta etapa.

A Natalia Spessot por impulsarme constantemente durante el desarrollo del proyecto, por todo el tiempo invertido y la ayuda brindada para que pueda avanzar con las tareas y la redacción de esta memoria.

A mi tutor Ing. Milton Sosa, agradecerle su aporte desinteresado en la corrección de este documento.

Al coordinador de carrera y a todos los profesores, por los conocimientos brindados durante toda la formación.

A Gastón, Jere, Alan y Fabri por todas las horas compartidas dentro y fuera de la facultad.

Al Ing. David Trepat por permitirme anteponer mi estudio ante las obligaciones laborales.

Por último, quiero agradecer al Ing. Elisandro Colussi y a la empresa COLVEN, por el tiempo brindado, el aporte económico y logístico para la fabricación de los circuitos impresos y para la importación de componentes y herramental necesario para el proyecto.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Objetivos

El mercado IoT o "Internet de las cosas" se encuentra en pleno auge a nivel mundial, en especial en Argentina, donde aún hay mucho por explotar. Con el desarrollo de nuevas tecnologías de comunicación, de menor costo y consumo de energía, permiten implementar soluciones más ajustadas para cada aplicación.

1.1. Objetivo general

El objetivo principal del proyecto es desarrollar un sistema de alarma domiciliaria con una detección de intrusos en la casa, de bajo consumo para funcionar a batería y que utilice BLE-Mesh [4].Inicialmente el proyecto publicado por la UNSAM se definió como un sistema de alarma para casas rurales, específicamente la de la parte interesada Elida Hermida en Verónica, Prov de Buenos Aires. Luego de una visita para observar las necesidades, se redefinió específicamente Domótica de la casa.

1.2. Objetivos específicos

- Implementar una red BLE-mesh.
- Desarrollar una conexión de datos por *WiFi*, respaldada por NB-IoT.

1.3. Aclaraciones

Si bien el proyecto fue desarrollado como un producto, no es parte del alcance ni se cubren en este documento las etapas de manuales, análisis de fallas, proceso productivo o lanzamiento del producto. Inicialmente el proyecto fue presentado con NB-IoT como innovación en Agosto del 2019. En esa fecha la cobertura por parte de las prestadoras de servicios de comunicaciones en la Argentina era nula y ningún proveedor ofrecía cobertura ni chip SIM [5] para NB-IoT. Pasado un año y medio de la fecha de comienzo del proyecto, transcurría plena situación de pandemia cuando se definió con el profesor de la cátedra y avalado por el director de carrera prescindir de esta funcionalidad, reduciéndola a GPRS.

Adicionalmente, durante la etapa de desarrollo de la red BLE-Mesh se detectó una discordancia en la definición del proyecto, una red *Mesh* no puede ser implementada por dispositivos con funcionalidad de bajo consumo o hibernación, ya que se pierden nodos de salto. Al igual

[4]BLE-Mesh es el acrónimo en ingles de *Bluetooth Low Energy Mesh*, malla bluetooth de bajo consumo.

[5]SIM es el acrónimo en inglés de *Subscriber Identity Module*, en español módulo de identificación de abonado.

que con el protocolo NB-IoT, en concertación con el profesor de la cátedra se decidió modificar el proyecto, se migró de una red tipo *mesh* a una tipo estrella.

1.4. Alcances

En el presente proyecto se desarrollan los siguientes temas:

- Tecnología GPRS y ESP-NOW [6].
- Circuito de respaldo de energía.
- Analizar e implementar una red tipo estrella de rápido aprovisionamiento y bajo consumo.

[6] ESP-NOW es un protocolo de comunicación inlámbrica detallado en 2.4.2.

Capítulo 2

Marco Teórico

El siguiente capítulo presenta los conceptos necesarios para la comprensión del proyecto. Se describe el diagrama general del proyecto y se presentan los recursos de *hardware* y *software* utilizados para el desarrollo del trabajo.

2.1. Internet de las cosas

En los últimos años, el término Internet de las Cosas o en inglés *Internet of Things*, con su acrónimo IoT se vuelve más popular y nombrado en la sociedad. El término IoT se refiere a la red colectiva de dispositivos conectados y a la tecnología que facilita la comunicación entre los dispositivos y la nube, así como entre los propios dispositivos [10]. Con la expansión continua del Internet de las Cosas, se ha empezado a utilizar cada vez más dispositivos conectados, desde teléfonos móviles hasta vehículos autónomos o sistemas de control.

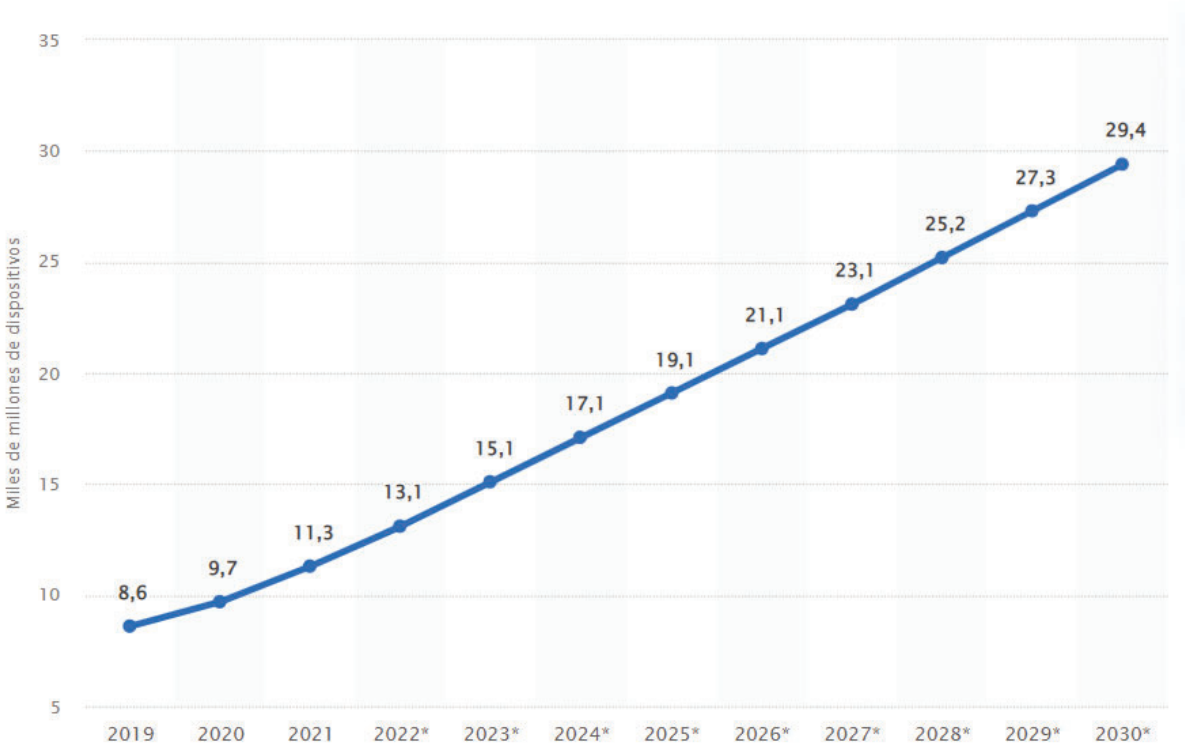

FIGURA 2.1. Cantidad de dispositivos en el mundo. Imagen tomada de [1].

A fines de 2018, se estimaba que había 22 mil millones de dispositivos conectados a IoT en todo el mundo. Según Statista esta estimación no se cumplió, como observamos en la Figura 2.1 la cantidad de dispositivos conectados a IoT en todo el mundo será de 38,6 millones para

2025. La llegada al mundo de tecnologías como 5G, NB-IoT, CAT-M1/M2 [7], como así también, el acceso más rápido y amplio a la red son promotores del crecimiento del IoT[11][12].

2.2. Diagrama de bloques general del proyecto

El diagrama de bloques general del proyecto es presentado en la figura 2.2 y consta de tres bloques:

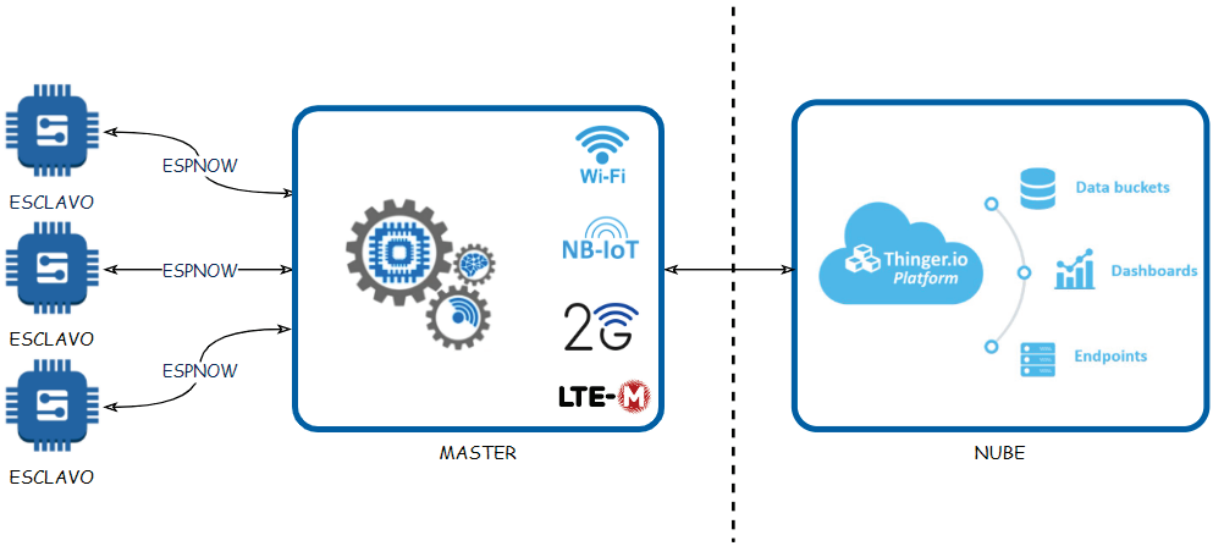

FIGURA 2.2. Diagrama de bloques general del proyecto

1. Dispositivos embebidos esclavos, serán los encargados de controlar las entradas/salidas, obedeciendo los comandos recibidos desde el máster.

2. Dispositivo embebido máster, compuesto por una lógica de funcionamiento y varios módulos de comunicación, será quien ejecute la lógica de cada modo de funcionamiento y envíe/reciba la información desde los servicios de backend.

3. Servicios de backend, compuesto por una base de datos y una GUI [8], permitirá la interacción del sistema con el usuario, permitiendo indicar toda la información del sistema y tomar acciones sobre el mismo.

Este último punto no forma parte del desarrollo del proyecto, se escogió una plataforma existente que se ajuste a las necesidades del mismo.

2.3. Componentes principales del hardware

2.3.1. Microcontrolador

La oferta de mercado de microcontroladores y circuitos integrados de comunicación es muy amplia, en específico para módulos *WiFi* y *Bluetooth* existen varias compañías que ofrecen una

[7] NB-IoT, CAT-M1, GPRS son diferentes tecnologías de telefonía móvil.

[8] GUI es el acrónimo en inglés de *Graphical User Interface*, Interfaz gráfica del usuario.

solución pre-certificada, acelerando drásticamente los tiempos de desarrollo y reduciendo costos de desarrollo si los volúmenes son pequeños. Dentro de estas compañías encontramos marcas como *Silicon Labs* [9], *Nordic Semiconductor* [10], *STMicroelectronics* [11] y *Espressif Systems* [12], siendo esta última la seleccionada para el proyecto.

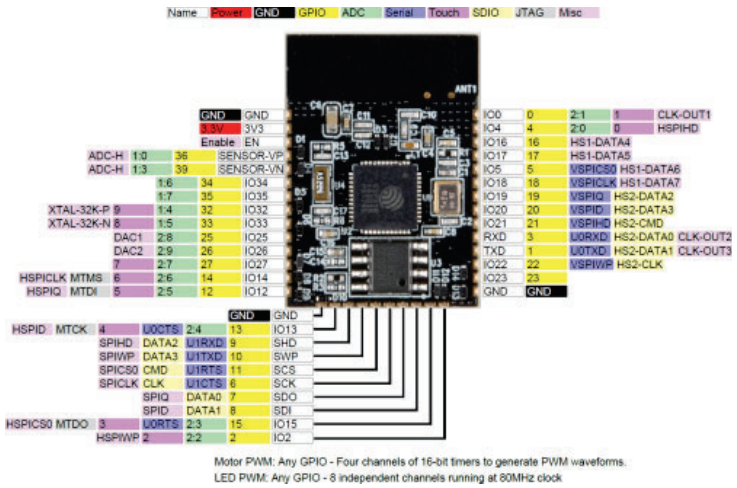

FIGURA 2.3. Imagen del módulo ESP32-WROOM. Imagen tomada de [2].

Es importante diferenciar entre SoC [13] y módulo. Este último contiene en su interior un SoC integrado junto a un cristal, antena, un circuito de adaptación de antena, una memoria flash y un regulador de tensión.

El módulo seleccionado tanto para el máster como para el esclavo fue el ESP32-WROOM[3], el cual podemos observar en la Figura 2.3. Internamente utiliza un SoC ESP32, este es capaz de gestionar tanto la comunicación como la lógica del microcontrolador. Integra características como *WiFi* y *Bluetooth* y es capaz de ejecutar tareas en tiempo real[13].

Se escogió este porque incluye todos las funcionalidades necesarias para el proyecto en un único módulo, reduciendo el área en el PCB, firmware, costo del dispositivo y los materiales necesarios.

En la Figura 2.4 se observa el diagrama de bloques del SoC. Algunas de las características técnicas más importantes del módulo ESP32-WROOM son:

- Dos microprocesadores Xtensa LX6 de 32 bits con una frecuencia de reloj de hasta 240 MHz y un rendimiento de hasta 600 DMIPS.
- Memoria SRAM de 520 KB para instrucciones y datos, memoria ROM de 448 KB para funciones de núcleo y boot.
- Memoria flash de 4/8/16 MB por SPI.
- Soporte de periféricos SD card, UART, SPI, SDIO, I2C, PWM, GPIO, ADC, DAC, CAN.
- Soporte para Wi-Fi 802.11 b/g/n (802.11n @ 2.4 GHz hasta 150 Mbit/s).

[9] https://www.silabs.com/

[10] https://www.nordicsemi.com/

[11] https://www.st.com

[12] https://www.espressif.com/

[13] SoC es el acrónimo en inglés de *System on Chip*, sistema en un chip.

- Soporte de características de seguridad del estándar IEEE 802.11, incluyendo WFA, WPA/WPA2 y WAPI.
- Soporte para Bluetooth v4.2 BR/EDR y BLE.

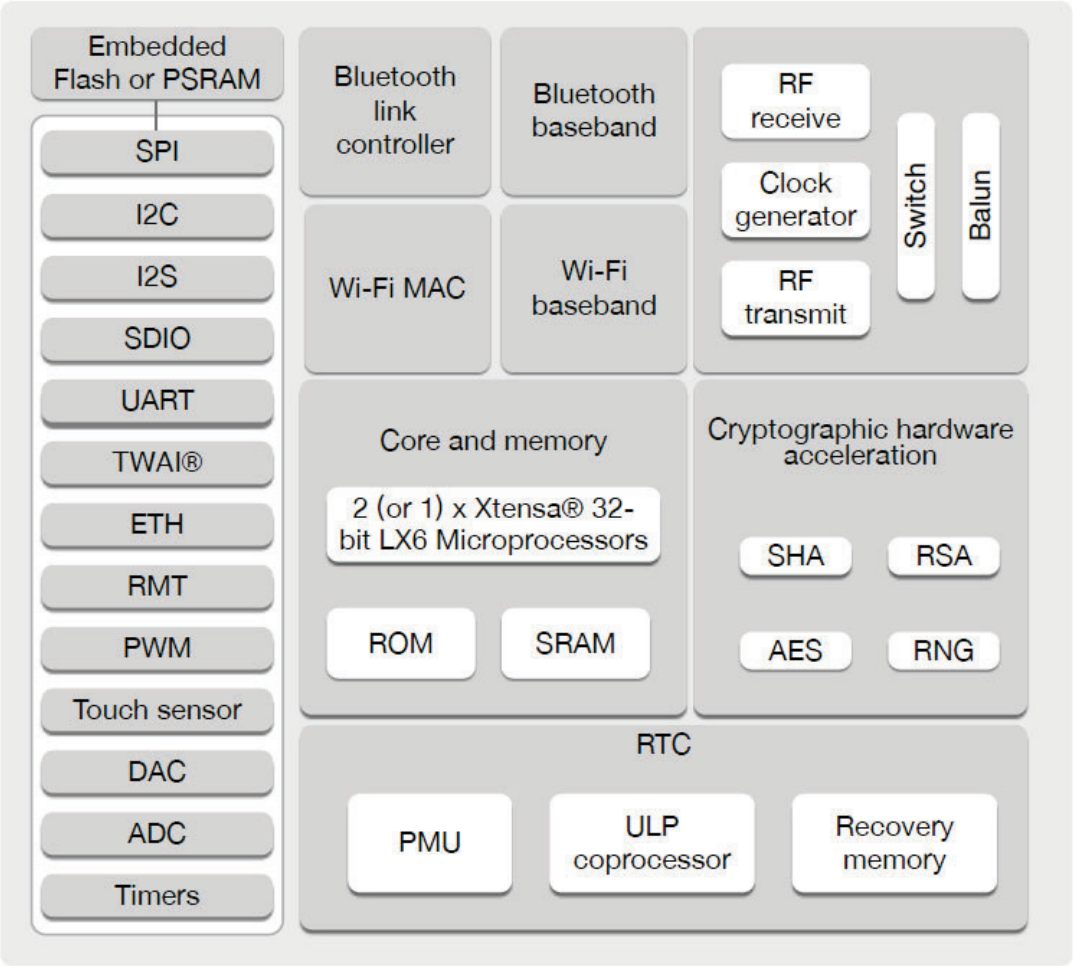

FIGURA 2.4. Diagrama de bloques del SoC ESP32. Imagen tomada de [3].

2.3.2. Módulo de comunicación

Como se describió en la Sección 1, el desarrollo se ideó con el protocolo de comunicación NB-IoT. Al momento de definir el módulo de comunicación se presumió que la implementación de este protocolo en Argentina podría llegar a demorarse; el módulo escogido para la comunicación fue el Quectel BG96, debido a que permite comunicación NB-IoT, CAT-M1 y GPRS, siendo esta última la utilizada en el proyecto. El encapsulado del módulo Quectel BG96 es del tipo BGA [14], como su propio nombre indica, es una matriz bidimensional de bolas que se disponen en la parte inferior del componente. Este tipo de encapsulado dificulta la fabricación de prototipos o proyectos de baja escala.

Lo más adecuado fue seleccionar un módulo de desarrollo de la marca MIKROE [15], estas placas de desarrollo existen en el mercado para acelerar los tiempos de desarrollo, prototipado

[14] BGA es el acrónimo en inglés de *Ball Grid Array*, matriz de rejilla de bolas.

[15] https://www.mikroe.com/

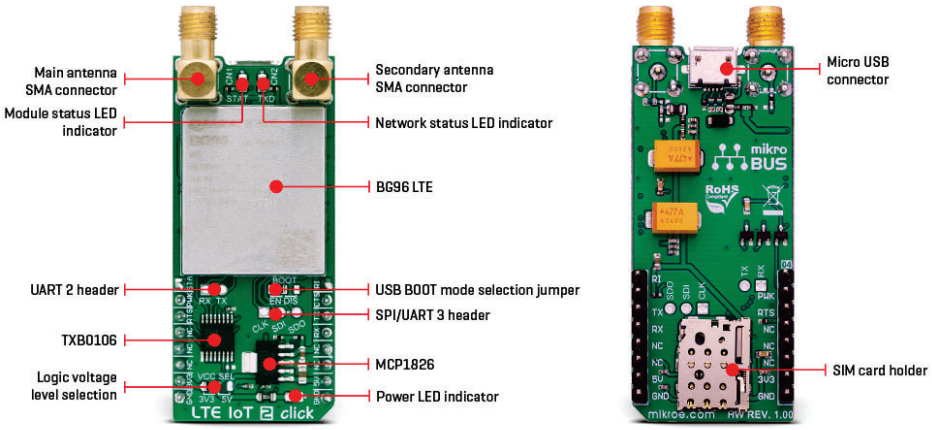

FIGURA 2.5. Módulo de comunicación LTE IoT 2 CLICK. Imagen tomada de [4].

y validación. Específicamente se seleccionó el módulo LTE-IoT 2 Click que utiliza un módulo Quectel BG96[4]; como se observa en la figura 2.5, resuelve el problema de montaje del mismo y agiliza el desarrollo de hardware, además de incluir el módulo de comunicación incluye todos los periféricos necesarios para su funcionamiento. La conexión del mismo es a través de dos tiras de pines tipo SIP [16] de paso 2,54 mm.

2.3.3. Respaldo por batería para dispositivo máster

Para proveer un respaldo de batería al dispositivo máster, el cual no puede entrar en un modo bajo consumo debido a que es el encargado de gerenciar la lógica de funcionamiento y con la condición de permanecer activo en la peor condición por al menos 4 días, se optó por utilizar una batería de gel de 12V.

El circuito integrado escogido para la gestión de la carga fue el UC3906DWTRG4, en la Figura 2.6 podemos observar el diagrama en bloques interno. Esta serie de controladores de carga de batería gobiernan tanto el voltaje de carga como la corriente del cargador a través de tres estados de carga diferentes, control de carga de alta corriente, otro de sobrecarga y por último uno de carga flotante o de *standby*[5].

2.3.4. Respaldo por batería para dispositivo esclavo

A diferencia del dispositivo máster, los esclavos pueden incorporar una lógica de bajo consumo, reduciendo drásticamente la cantidad de energía necesaria en la batería. Gracias a esta funcionalidad la batería escogida para los dispositivos esclavos fue una 18650 de Li-Ion [17]. Para gestionar la carga de la misma se optó por emplear un módulo disponible en el mercado, el control de carga es gestionado por el circuito integrado TP4056[14].

[16] SIP es el acrónimo en inglés de *Single in-line Pin*, pines en línea simple.

[17] Litio Iones.

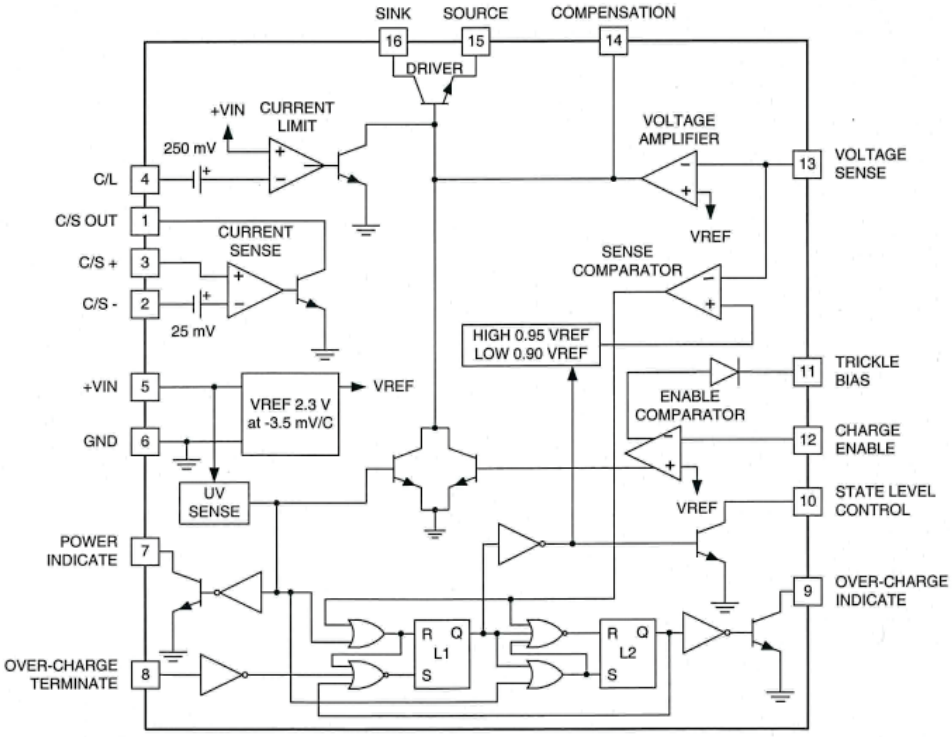

FIGURA 2.6. Diagrama de bloques del UC3906. Imagen tomada de [5].

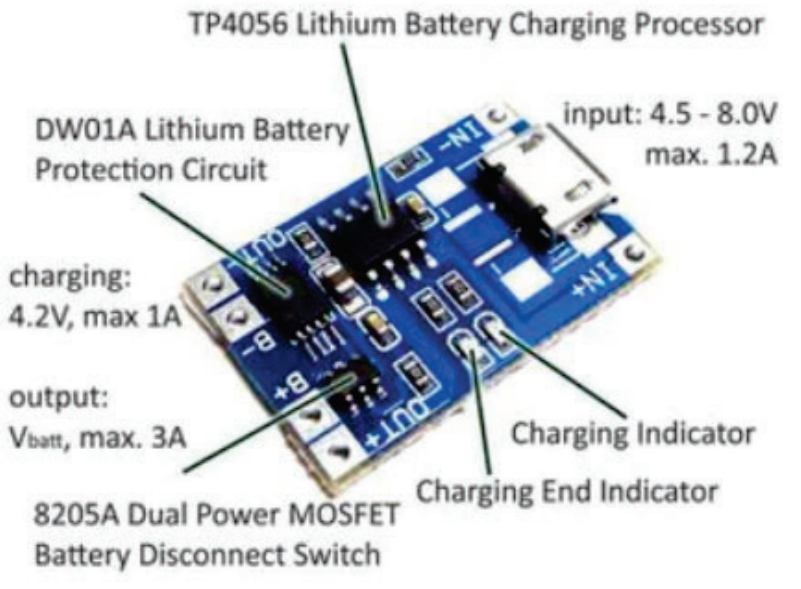

FIGURA 2.7. Módulo comercial para carga de baterías 18650. Imagen tomada de [6].

2.4. Componentes principales del software

2.4.1. Firmware ESP32

La empresa Espressif ofrece un IDE [18] denominado *Espressif IoT Development Framework.* o más conocido como ESP-IDF, se basa en consola de comandos y utiliza como lenguaje de programación nativo C/C++.

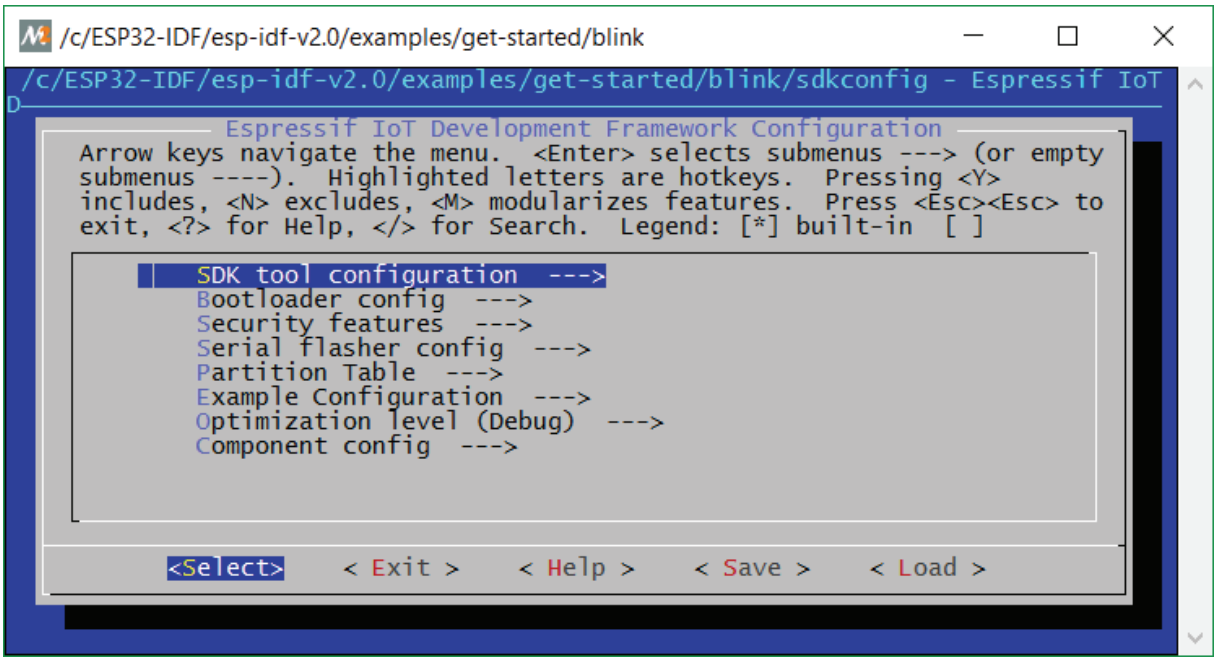

FIGURA 2.8. Consola de comandoes de ESP-IDF.

Como alternativa para el desarrollo se puede incorporar el ESP-IDF a diferentes IDE, obteniendo así un entorno de programación más ameno. Existen varias opciones como pueden ser *Eclipse*, *Platformio* a través de *Visual Studio Code* o el propio IDE de Arduino, este último de bajas prestaciones debido a que no ofrece modo depuración ni *cross reference*[15].

Como opción para el desarrollo se escogió el *Visual Studio Code*, que implementa el ESP-IDF como *plugin*, podemos observar el entorno de desarrollo en la Figura 2.9. Esta opción nos permite acceder a un SDK[19] completo del SoC ESP32, con todos los drivers de periféricos desarrollados por el fabricante, ejemplos de desarrollo, *debug* usando Open OCD o GDB y un control del proyecto con el entorno visual de VS. Para el proyecto se configuró la versión 4.2 del ESP-IDF.

2.4.2. ESP-NOW

ESP-NOW es un protocolo de comunicación inalámbrico definido por Espressif. Este protocolo está basado en la capa de enlace de datos del modelo OSI, reduce las capas de red, transporte, sesión, presentación y aplicación a una única capa. Además, ESP-NOW ocupa menos recursos de CPU y memoria flash que los protocolos de conexión tradicionales, mientras que coexiste con *WiFi* y *Bluetooth* [16].

[18] IDE es el acrónimo en inglés de *Integrated Development Environment*, Entorno de desarrollo integrado.

[19] SDK es el acrónimo en inglés de *Software Development Kit*, kit de desarrollo de software.

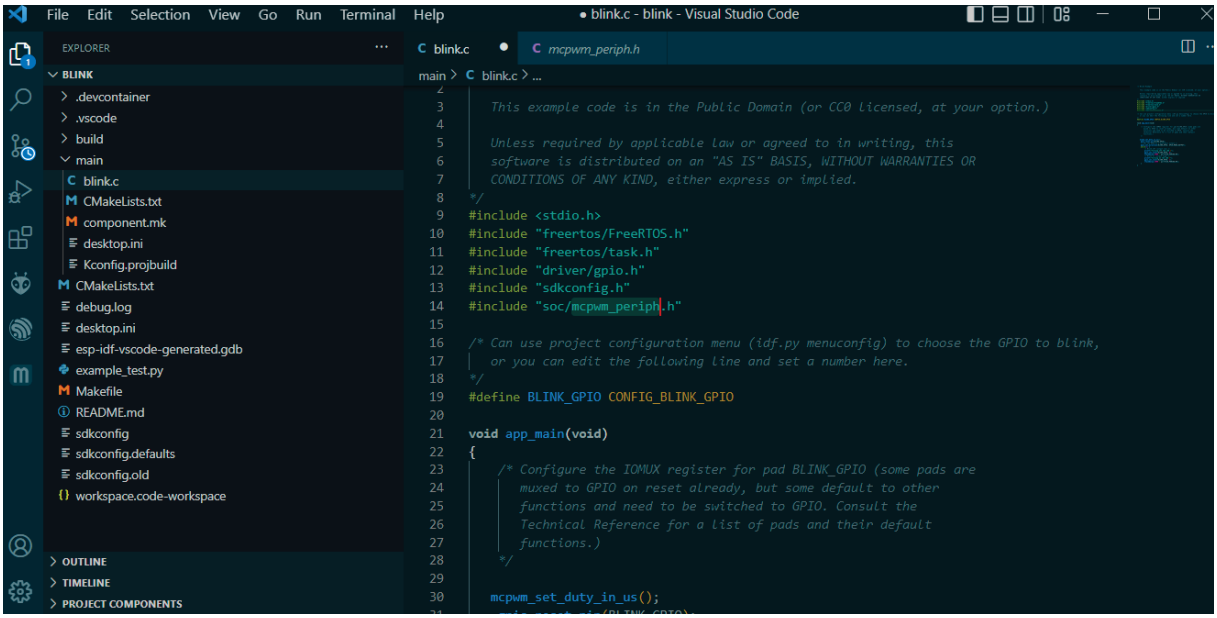

FIGURA 2.9. Imagen del IDE *Visual Studio Code*.

Proporciona una transmisión de datos flexible que es adecuada para conectar dispositivos punto a punto. En la Figura 2.10 se observan dos formatos de red diferentes, la topología utilizada en el proyecto fue tipo estrella.

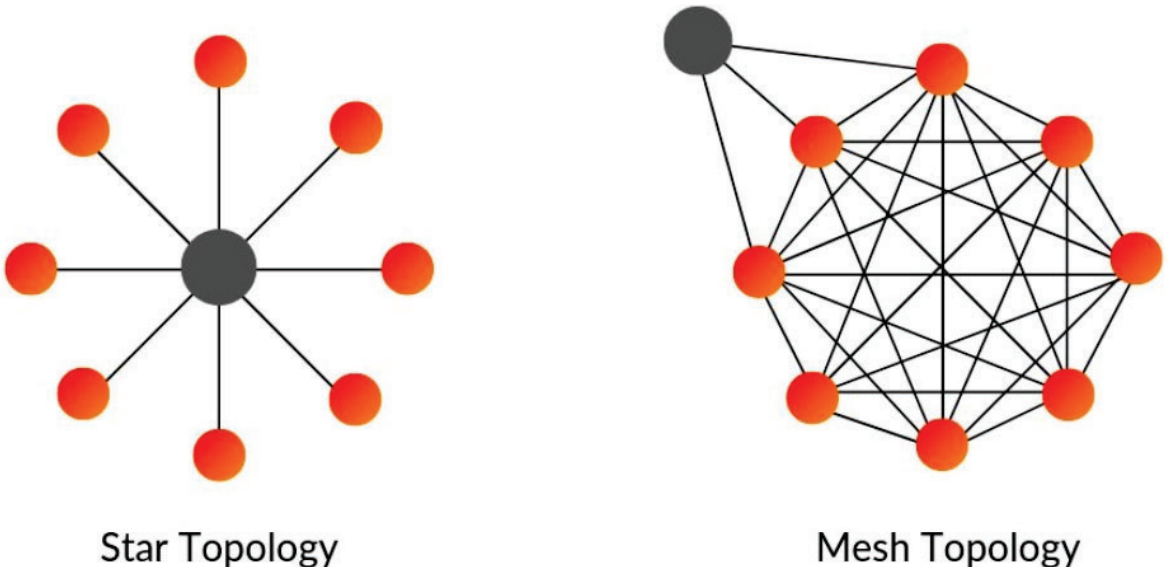

FIGURA 2.10. Comparación de diagrama de comuniación tipo *mesh* y estrella. Imagen tomada de [7].

En cuanto a la capa de seguridad, ESP-NOW nos ofrece encriptación por AES128-CCM [20], el dispositivo mantiene una clave PMK[21] y varias claves LMK [22], ambas de 16 bytes de longitud.

[20] Acrónimo en inglés de *counter with cipher block chaining message authentication code*, un método de encriptación descripto en el IEEE Std. 802.11-2012.

[21] PMK es el acrónimo en inglés de *Private Master KEY*, clave maestra privada.

[22] PMK es el acrónimo en inglés de "*Local Master KEY*, clave maestra local.

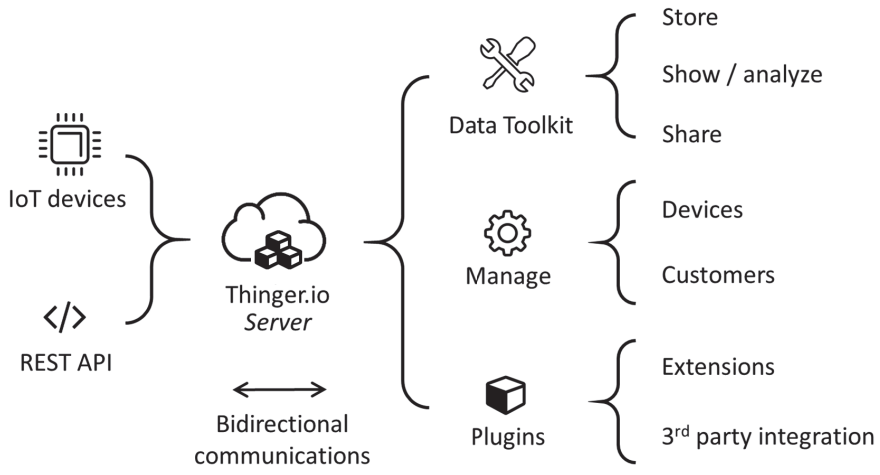

FIGURA 2.11. Diagrama de características de Thinger.io. Imagen tomada de [8].

2.4.3. Plataforma de visualización

Thinger.io es una plataforma IoT de código abierto en la nube, que proporciona todas las herramientas necesarias para crear prototipos, escalarlos y luego administrarlos de una manera muy simple.

La plataforma Thinger.io[23] está formada por dos productos principales: un *backend* (que es el servidor IoT real) y un *frontend* basado en la web. En la Figura 2.11 se muestran las principales características que brinda esta plataforma para proyectos IoT.

La plataforma ofrece bibliotecas específicas para microcontroladores Arduino compatibles y módulos Espressif, además de bibliotecas genéricas para conexión por *WiFi* y GPRS. Otras formas de conexión a la plataforma son por solicitudes HTTP[24] o MQTT [25].

Para el desarrollo del proyecto, luego de analizar 10 plataformas[17], se escogió Thinger.io debido a los siguientes puntos:

- Facilidad de configuración de los tableros de comando, pensando que esta tarea no es parte del proyecto y queda a cargo del usuario final.
- Ofrece un tipo de usuario sin cargo que se ajusta a las necesidades del proyecto.
- Biblioteca específica para ESP32 y módulos GPRS, los dos tipos de comunicación implementados en el proyecto.

[23]https://www.thinger.io

[24]HTTP es el acrónimo en inglés de *hypertext transfer protocol*, protocolo de transferencia de hipertextos.

[25]MQTT es el acrónimo en inglés de *message queuing telemetry transport*, es un protocolo de IBM destinado a mejorar la conectividad máquina a máquina.

Capítulo 3

Diseño e implementación

En este capítulo se presentan los componentes del *hardware* y *firmware* que se desarrollaron para el proyecto. También se expone el portal de acceso y visualización.

3.1. Diseño del hardware por etapas

En esta sección se detallan las diferentes etapas de los circuitos esquemáticos correspondientes a los módulos máster y esclavos.

3.1.1. Etapa de entrada y carga de batería módulo máster

Como se mencionó en la sección 2.3.3 el módulo máster se respalda por una batería de gel de 12 V. La corriente promedio del módulo máster es de 100 mA a 3,3 V en los momentos que ninguna salida se encuentra activa y el sistema no transmite por GPRS, esta sería la condición normal de uso. Además, la fuente *switching* detallada en la sección 3.1.2 logra una eficiencia del 95 %. Con esta información podemos calcular con la Ecuación 3.1 la corriente suministrada por la batería en estado normal del módulo.

$$Ibat = \frac{V_{switching}}{V_{bat}} * \frac{I_{mcu}}{F_{eff}} = \frac{5V}{12V} * \frac{0,1A}{0,95} = 0{,}044\ A \tag{3.1}$$

En la situación que el *WiFi* al que se encuentra conectado el módulo no posea conexión a internet y el sistema transmita por red celular, la corriente promedio del módulo asciende a 178mA y la corriente suministrada por la batería a 0.075A.

$$Duracion_{bat} = \frac{I_h}{I_{bat} * 24} = \frac{7Ah}{0,075A * 24h/dia} = 3,88\ dias \tag{3.2}$$

Para el proyecto se escogió una batería de 7 Ah y con la ecuación 3.2 se obtiene una duración de 3,88 días.

Como se presentó en la sección 2.3.3 el circuito integrado UC3906 gestiona la carga de la batería. En la hoja de datos del fabricante se presenta el circuito de aplicación utilizado, el cual se observa en la Figura 3.2. Si observamos la Figura 3.1 se encuentra el diagrama de estados de carga y las ecuaciones detalladas por el fabricante para la configuración de los valores que caracterizan la carga. Para el proyecto se definió:

- Voltaje de flote de la batería V_F = 13,5V.
- Voltaje de sobrecarga V_{OC} = 14,5V.
- Corriente de carga I_t = 0,25A.
- Corriente máxima I_{max} = 0,5A.

- Voltaje de entrada V_{in} = 17V.

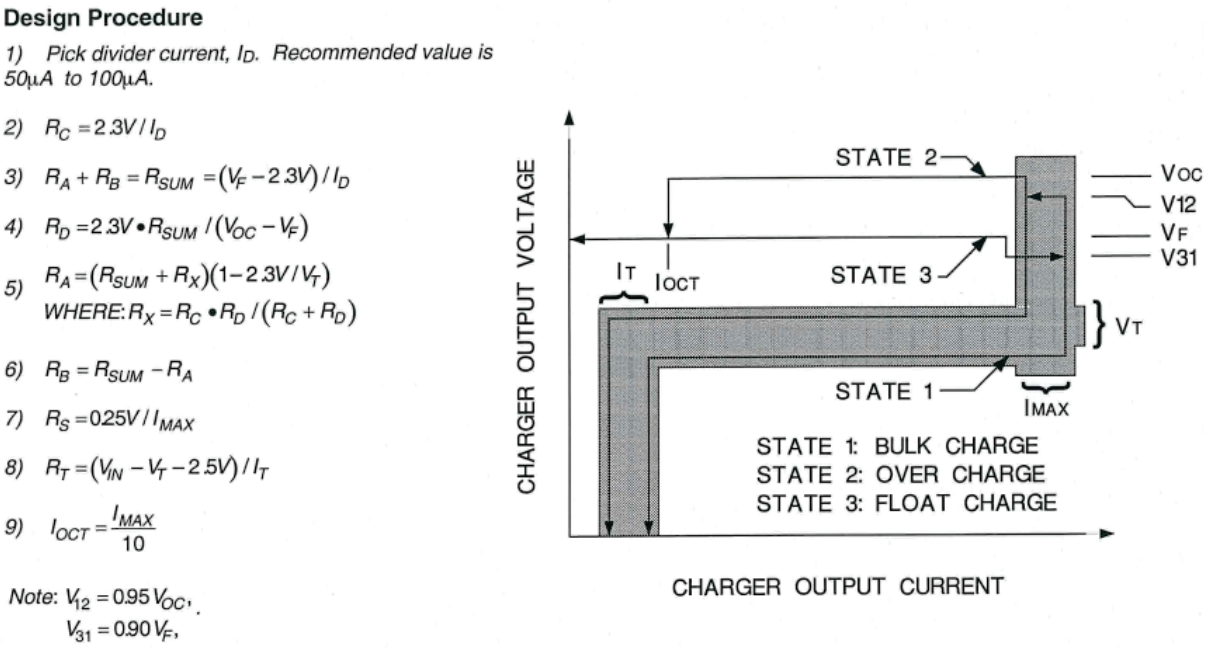

FIGURA 3.1. Diagrama de estados de carga. Imagen tomada de [5].

Con estos parámetros definidos y las ecuaciones detalladas en la Figura 3.1 se obtuvieron los valores de componentes utilizados en el circuito de la Figura 3.2. En la misma figura, en la entrada V_{IN} se encuentra un circuito de protección de corriente por *polyswitch* (F1) e inversión de polaridad (D1), seguido de una etapa de filtrado de ruido electromagnético conducido.

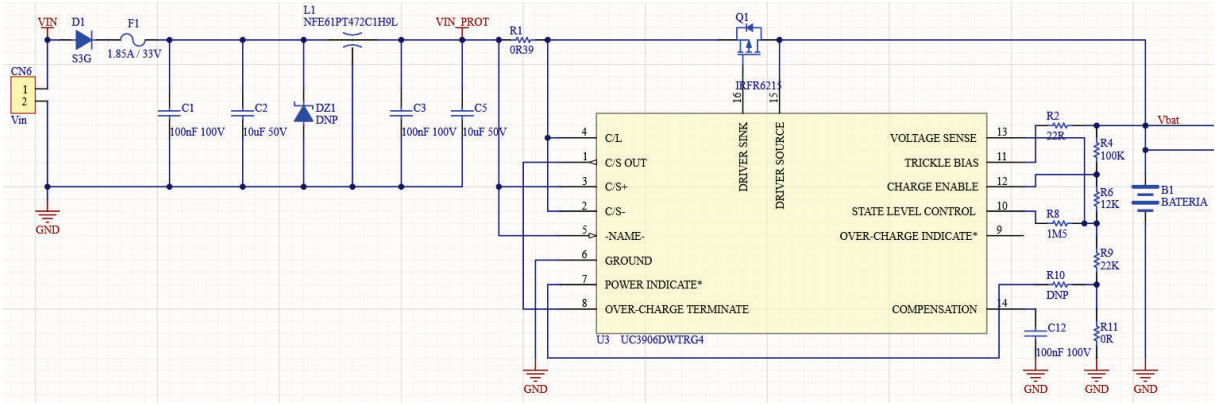

FIGURA 3.2. Circuito de entrada y cargador de batería.[26]

3.1.2. Etapa de fuentes módulo máster

Para el funcionamiento correcto de los diferentes componentes del módulo máster es necesario regular el voltaje de la batería a 5 V y a 3,3 V. Esto se realiza en dos etapas, inicialmente una fuente *switching* reduce el voltaje de la batería a 5 V, luego un regulador LDO [27] reduce de 5 V a 3,3 V. Para la fuente *switching* se decidió utilizar el circuito integrado AOZ1282CI[18],

[26] DNP es el acrónimo en inglés de *Do not populate*, componentes del circuito que no se deben montar.

[27] LDO es el acrónimo en inglés de *low drop out*, regulador de baja caída.

que proporciona una salida de 1.2 A desde 4,5 V hasta 36 V dependiendo de la configuración que se aplique. El circuito de aplicación que se observa en la Figura 3.3 fue tomado de la hoja de datos del fabricante, como así también los valores de los componentes pasivos, los cuales fueron calculados con las ecuaciones provistas[19].

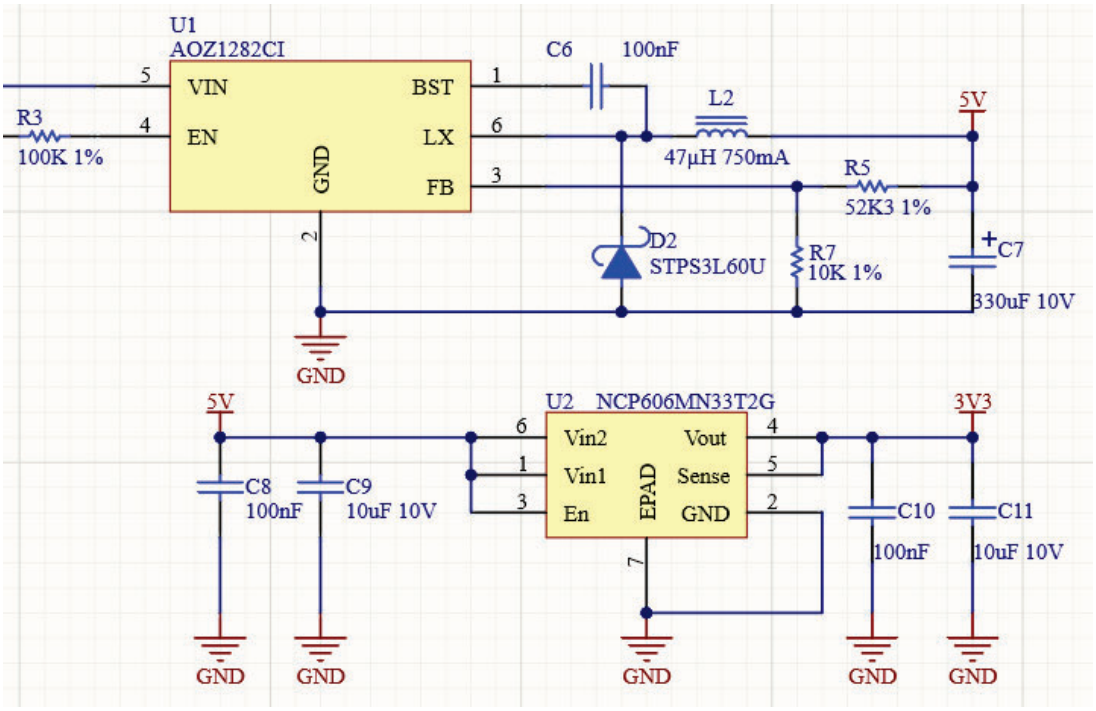

FIGURA 3.3. Circuito de regulación de fuente máster.

En el caso del regulador LDO de 3,3 V se utilizó el circuito integrado NCP606MN33T2G[20] que proporciona hasta 500mA.

3.1.3. Etapa de fuente y carga de batería módulo esclavo

Como se detalló en la sección 2.3.4 los módulos esclavos utilizan una batería recargable del tipo Li-Ion de 3,7 V de tensión nominal y 2500 mAh, la cual posee un módulo de protección BMS [28], este es el encargado de evitar accidentes derivados del comportamiento del litio en sus fases de carga y descarga. La carga es gestionada por un módulo TP4056 como el de la Figura 2.7, que posee LEDs de indicación de estado y un circuito de protección por sobre-descarga controlado por el circuito integrado DW01A. Para los módulos esclavos la duración de la batería es proporcionalmente directa a la activad del esclavo y será explicada en la sección 3.2.4. Nuevamente, para la regulación de fuente se utilizó el circuito integrado NCP606MN33T2G mencionado en la sección 3.1.2.

3.1.4. Microcontrolador

En la Figura 3.5 se observa el bloque esquemático del módulo ESP32. Los pulsadores S1 y S2 se utilizan para iniciar el módulo en modo arranque o reset respectivamente. Posee dos conectores tipo IDC [29] de paso 0,635 mm que se utilizan para conectar la herramienta de programación y depuración específica de *Espressif* llamada ESP-PROG[21], CN5 para la depuración

[28] BMS es el acrónimo de *Battery Management System*, sistema de gestión de batería.

[29] IDC es el acrónimo en inglés de *insulation-displacement connector*, conector por desplazamiento del aislante.

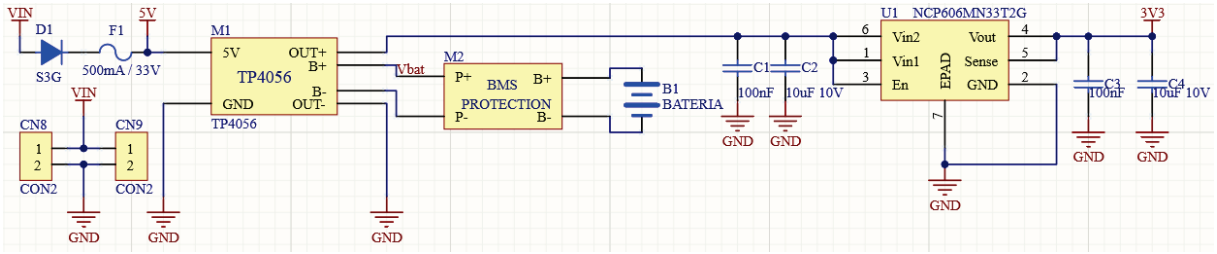

FIGURA 3.4. Circuito de carga y regulación de fuente esclavo.

y CN4 para la programación por UART [30]. El módulo de comunicación GPRS se comunica con el ESP32 a través de la UART, por lo tanto los *jumpers* CN7 y CN8 sirven para poder conectar o desconectar el ESP32, el módulo Quectel o la herramienta de programación entre sí. Las entradas IO34 e IO35 son dos divisores de tensión para digitalizar el voltaje de entrada V_{in} y el nivel de batería V_{bat}. Por último, pensando en futuros desarrollos se colocó un cristal (Y1) de 32,768 KHz[22] específico para el RTC [31].

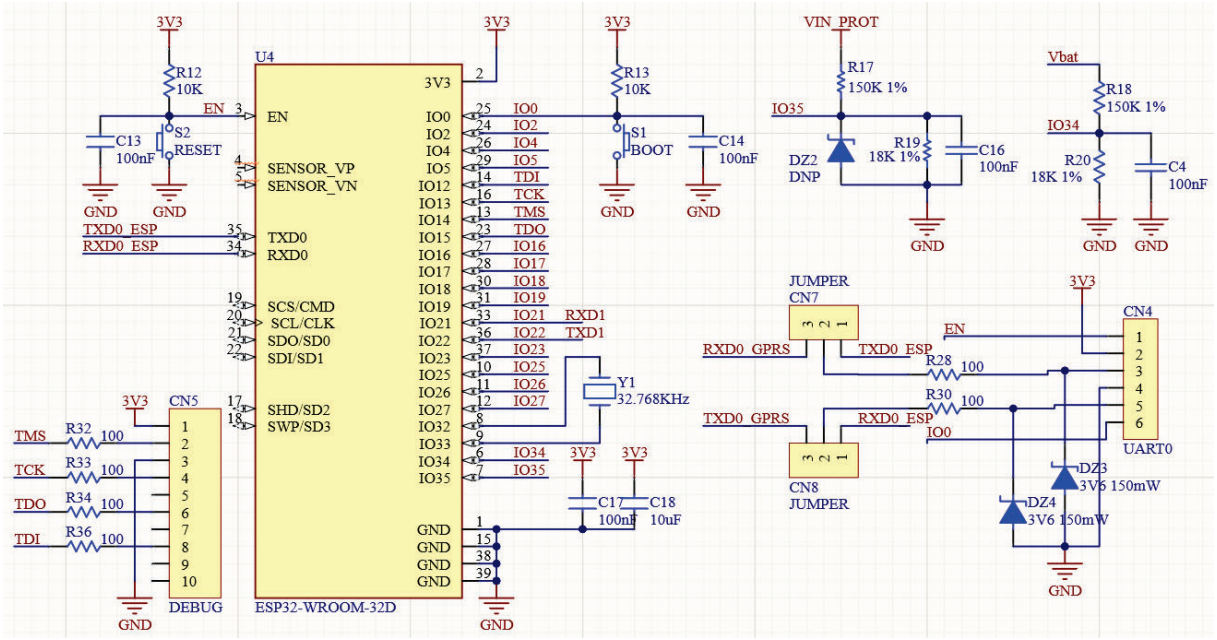

FIGURA 3.5. Módulo ESP32 máster con periféricos

La única diferencia en la configuración de *hardware* entre el módulo ESP32 máster y los módulos esclavos, es que estos últimos no poseen los *jumpers* para el módulo de comunicación, a su vez el divisor resistivo para la medición del nivel de batería está ajustado para 4,7 V.

3.1.5. Circuitos de entrada máster y esclavo

Como se observa en la Figura 3.6 hay tres tipos de circuitos configurados para entradas al MCU[32], estos fueron diseñados pensando en fabricar un único modelo de módulo esclavo polivalente. Analizando inicialmente la configuración del tipo sensor 1 y sensor 2, están diseñados

[30]UART es el acrónimo en inglés de *Universal Asynchronous Receiver-Transmitter*, Transmisor-Receptor Asíncrono Universal.

[31]RTC es el acrónimo en inglés de *Real Time Clock*, reloj de tiempo real.

[32]MCU es el acrónimo en inglés de *microcontroller unit*, microcontrolador.

para lectura por ADC [33] de un sensor del tipo resistivo con *pull-up*, a su vez se puede conectar otro tipo de sensor con salida analógica de tensión hasta 3,3 V. También, configurando la entrada del MCU como digital se puede conectar una llave o pulsador en S3.

Segundo, el circuito de configuración del sensor PIR [34] está pensado para los módulos de sensores integrados de este tipo que se comercializan con la lógica de control incluida, entregando una señal digital procesada. Las resistencias R20, R23 y R26 sirven para configurar la entrada con *pull-up/pull-down* y limitar la corriente si fuera necesario.

Finalmente, la entrada CN5 nos permite configurarla para diferentes usos dependiendo de los componentes montados en el PCB[35]. Los *polyswitches* F4 y F5 nos dejan escoger que valor de alimentación entregar en el pin 1 del conector. Las resistencias R22 y R25 nos permiten variar un divisor resistivo o colocar un *pull-down*, el diodo *zenner* nos permite una protección de tensión básica sobre el pin del microcontrolador. Configurando la entrada IO25 como analógica o digital sumado a la configuración de *hardware* que se monte en el PCB, se logra conectar una gran variedad de sensores analógicos o digitales.

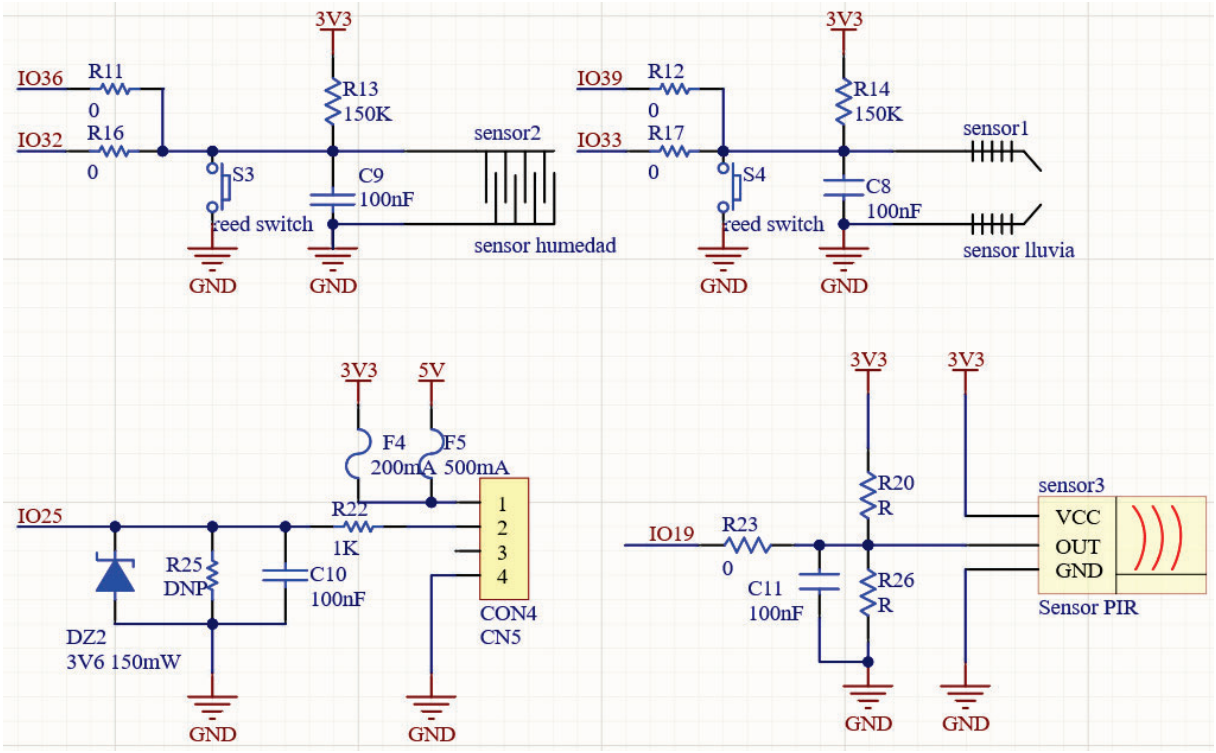

FIGURA 3.6. Circuitos de entradas analógicas y digitales.

Las entradas digitales y analógicas están configuradas de la misma forma para el módulo máster y para los esclavos, la única diferencia es que el módulo máster posee una sola entrada, configurada para sensor PIR.

3.1.6. Circuitos de salida máster y esclavo

Para el diseño de los circuitos de salida se tomó el mismo criterio que para las entradas, un único diseño de PCB que sea polivalente. Por este motivo, en la Figura 3.7 se pueden observar las diferentes configuraciones de las salidas.

[33] ADC es el acrónimo en inglés de *Analog-to-Digital Converter*, conversor analógico digital.

[34] PIR es el acrónimo en inglés de *Passive Infrared*, detector pasivo infrarojo.

[35] PCB es el acrónimo en inglés de *Printed circuit board*, placa de circuito impreso.

A la izquierda, se observa la salida por *relay*, la bobina es accionada a través del transistor pre-polarizado integrado DTC015EMT2L[23], en el caso de montar únicamente R21 la salida queda configurada en modo normal cerrado, en cambio si montamos R23 la salida queda configurada en modo normal abierto.

A la derecha, se observa un circuito de salida de señal, el cual dispone de varias opciones:

- Se puede alimentar la salida con 5 V o 3,3 V, según se monte F2 o F6 respectivamente.
- Si se monta R24 o R26, la salida de señal se puede referenciar a nivel de alimentación o a masa por *pull-up* o *pull-down* respectivamente.
- Montando R25 o Q3, el pin del MCU queda conectado directamente al pin 3 del conector o a través de un transistor en modo colector abierto.

Esta variedad de configuraciones nos permite controlar cargas por PWM o también conectar un servo en la salida.

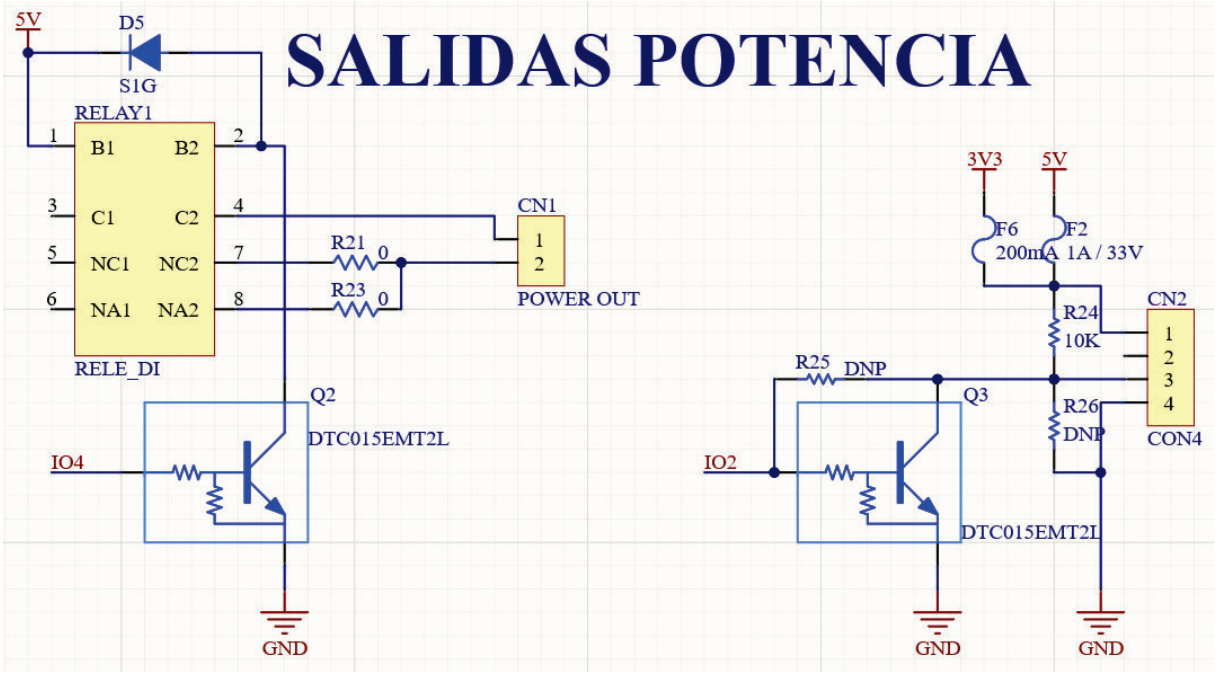

FIGURA 3.7. Circuitos de salidas.

Tanto el módulo máster como los esclavos disponen de una salida de señal, la diferencia es que estos últimos disponen de dos salidas tipo *relay* y el máster solo una.

3.1.7. Circuitos auxiliares

Proyectando futuras ampliaciones del proyecto, se implementaron dos conectores en el módulo máster, se pueden observar en la Figura 3.8.

El conector CN3 está configurado para conectar un *display* tipo LCD[36] con controlador ST7789. Asimismo el conector I1 nos permite obtener una UART extra o una conexión I^2C[37] dependiendo de las funciones que se activen en el MCU. En los módulos esclavos únicamente se encuentra disponible el conector I1.

[36] LCD es el acrónimo en inglés de *Liquid Cristal Display*, pantalla de cristal líquido.

[37] I^2C es el acrónimo en inglés de *Inter-Integrated Circuit.*, un bus serie de datos de dos hilos.

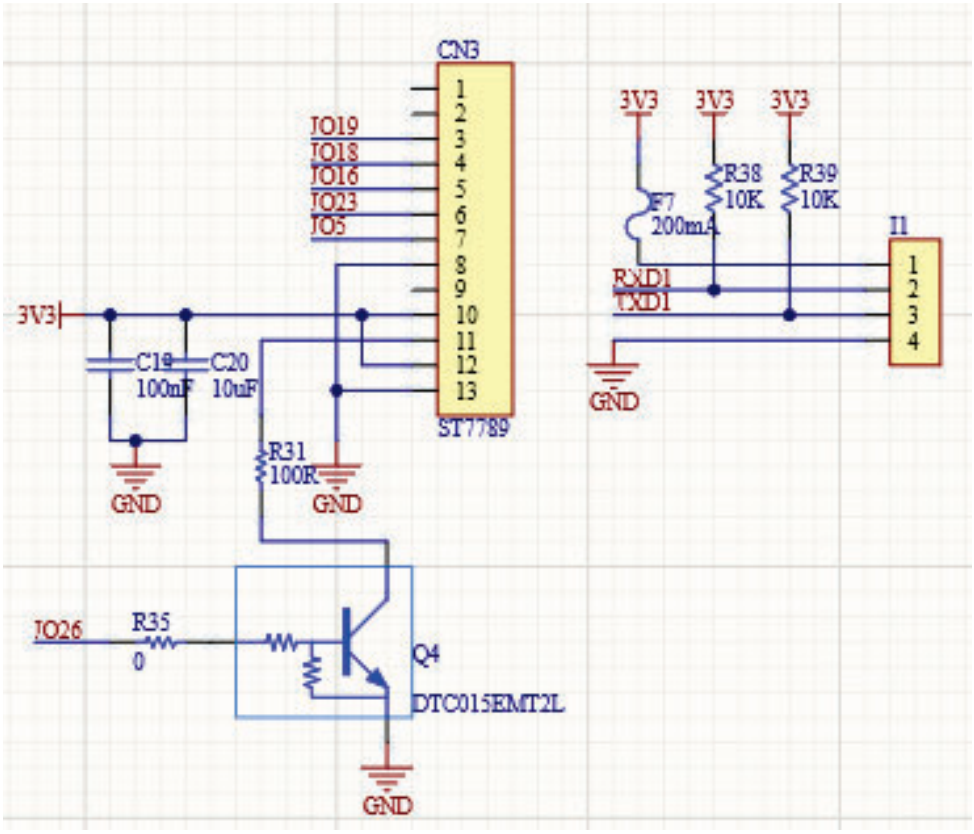

FIGURA 3.8. Circuitos auxiliares.

3.2. Firmware implementado

3.2.1. Diagrama de bloques esclavo

Cada módulo esclavo seguirá la lógica de programación que se observa en la Figura 3.9. Inicialmente, se procede a la inicialización del *hardware*, en específico el *watchdog*[38], luego se da curso a la función encargada de configurar el resto de los periféricos detallado en la Sección 3.2.3. El módulo esclavo está destinado principalmente a adquirir información de las entradas y enviarlas al máster a través de ESP-NOW. A su vez, se puede frenar el ciclo de funcionamiento por una interrupción de ESP-NOW en caso de recibir datos para actualizar las salidas.

Siempre que la alimentación de red se encuentre activa, los módulos conservarán el 100 % de sus funciones. Para lograr una mayor duración de la batería, en el momento que el MCU detecte un corte en la alimentación de red y si la batería se encuentre por debajo de un umbral definido, el equipo entrará en modo *sleep*, su funcionamiento se detalla en la sección 3.2.4.

3.2.2. Diagrama de bloques máster

El módulo máster presenta un funcionamiento diferente al esclavo, su lógica se presenta en la Figura 3.10. Al igual que el módulo esclavo, se procede a iniciar el *watchdog*, inmediatamente el resto de los periféricos. La lógica de funcionamiento, además de adquirir la información de las entradas del máster y tomar acciones sobre las salidas, se encarga de la transferencia de datos hacia el portal *Thinger.io*, detallado en la sección 3.3. Posee un modo alarma, detallado en la sección 3.2.8.

[38] Es un temporizador que permite reiniciar el sistema en caso que se haya bloqueado.

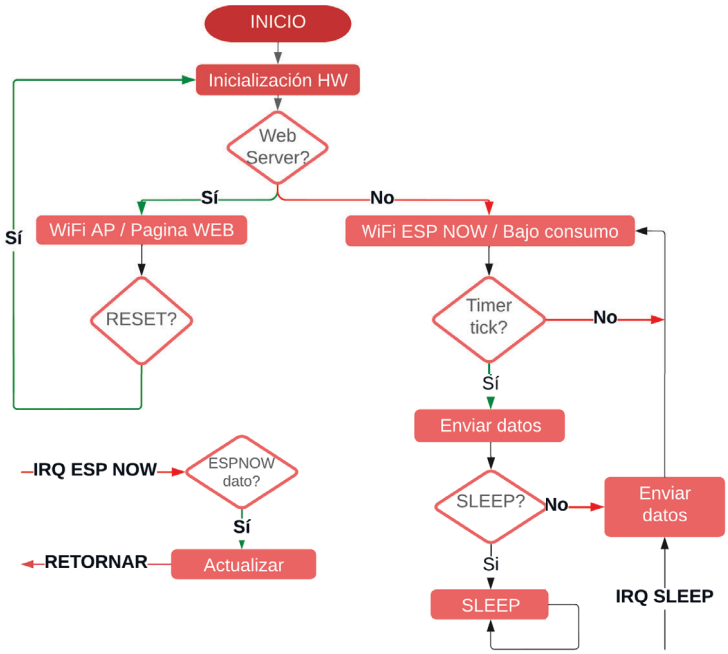

FIGURA 3.9. Diagrama de bloques de *firmware* módulo esclavo.

La lógica de funcionamiento no contiene un modo *sleep* debido a que el máster debe estar activo el 100 % del tiempo para recibir información de los esclavos y reportar la misma al portal sin perder conexión.

3.2.3. Inicialización y periféricos

Tanto en el módulo máster como en los esclavos, luego de la configuración del *watchdog*, la siguiente función a ejecutar es la configuración del *hardware*. La tabla 3.1 presenta la configuración asignada a cada pin físico del ESP32 esclavo, dirección y periférico asociado. Al igual

TABLA 3.1. Configuración de pines de los módulos esclavo

Pin	Puerto	Dirección	Periférico	Detalle
3	EN	entrada	GPIO	reset
6	IO34	entrada	ADC1_CH6	Vbat
7	IO35	entrada	ADC1_CH7	Vin
8	IO32	entrada	ADC1_CH4	sensor_2
9	IO33	entrada	ADC1_CH5	sensor_2
10	IO25	entrada	GPIO	CN5
24	IO2	salida	GPIO	RELE_2
25	IO0	entrada	GPIO	boot
27	IO16	salida	GPIO	PWM
30	IO18	salida	GPIO	RELE_1
31	IO19	entrada	GPIO	PIR
33	IO21	entrada salida	I2C	SDA
34	U0RXD	entrada	UART_0	115200 bps
35	U0TXD	salida	UART_0	115200 bps
36	IO22	entrada	I2C	SCL

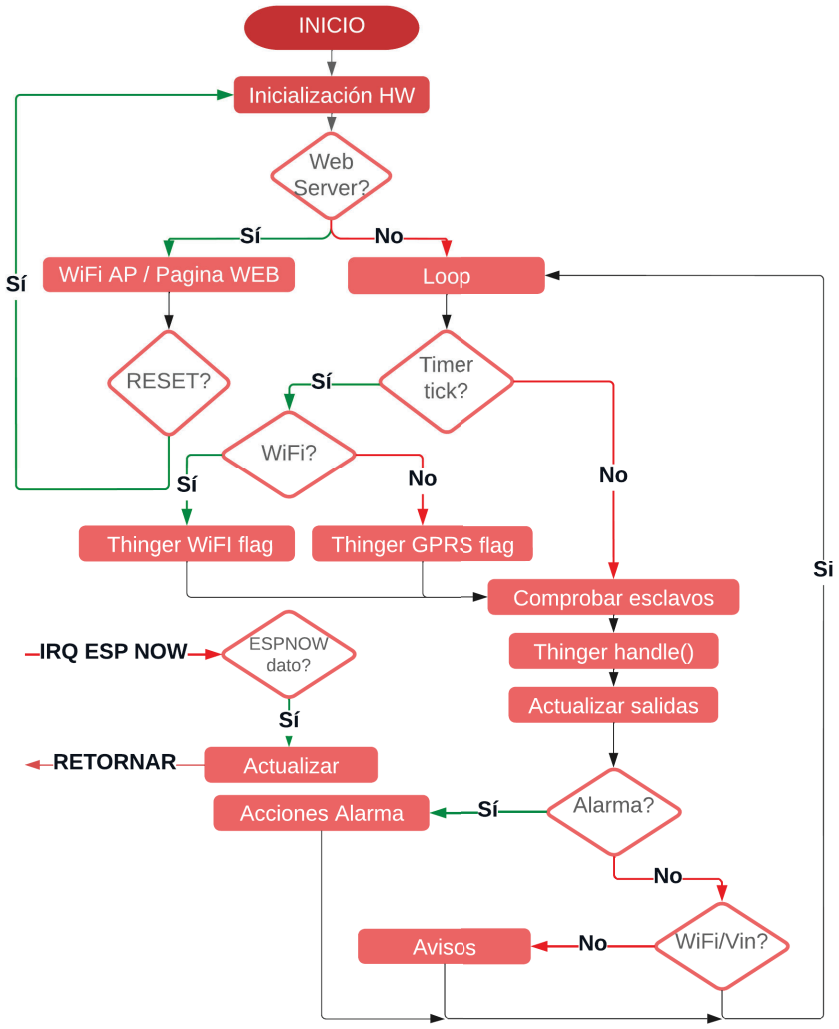

FIGURA 3.10. Diagrama de bloques de *firmware* módulo máster.

que un módulo esclavo, el máster realiza el mismo procedimiento de configuración de *hardware* pero con una asignación diferente, ya que la función principal de este es la comunicación al portal. Su configuración se presenta en la tabla 3.2.

3.2.4. Modo dormido o *sleep*

Como se mencionó en la sección 2.3.4, los módulos esclavos poseen una lógica de bajo consumo para extender la duración de la batería. En ausencia de alimentación de red y con el voltaje de la batería por debajo de 3,7 V, el MCU activa la función *sleep* reduciendo el consumo de corriente de 100 mA a solo 1 mA. Las condiciones para salir de este modo y transmitir información al máster son:

- Activación de las entradas sensor PIR o CON4 detalladas en la Figura 3.6.
- Ciclo de tiempo fijo, definido en 2 minutos.

Una vez transmitida la información al máster y transcurridos ocho segundos sin actividad en los sensores, el módulo entra nuevamente en modo *sleep*. La activación de una de las entradas tiene mayor prioridad que el ciclo de tiempo, por lo tanto, cada vez que el módulo se despierte por alguna de las entradas, el ciclo de tiempo se reinicia. Idealmente, si las entradas no se

TABLA 3.2. Configuración de pines del módulo máster

Pin	Puerto	Dirección	Periférico	Detalle
3	EN	entrada	GPIO	reset
6	IO34	entrada	ADC1_CH6	Vbat
7	IO35	entrada	ADC1_CH7	Vin
10	IO25	salida	GPIO	LTE PWK
12	IO27	entrada	GPIO	PIR
21	U2RXD	entrada	UART_2	115200 bps
22	U2TXD	salida	UART_2	115200 bps
24	IO2	salida	GPIO	PWM
25	IO0	entrada	GPIO	boot
26	IO4	salida	GPIO	RELE_1
28	IO17	entrada	GPIO	LTE STA
34	U0RXD	entrada	UART_0	115200 bps
35	U0TXD	salida	UART_0	115200 bps

activan, un módulo esclavo necesita una corriente promedio de 16 mA para realizar los ciclos de transmisión cada dos minutos, logrando así una duración de la batería de 6,5 días.

3.2.5. ESP-NOW

Como se detalló en la Sección 2.4.2, el protocolo implementado para la comunicación entre módulos es específico de *Espressif* y se llama ESP-NOW, este utiliza como base una comunicación *WiFi* con protocolo 802.11b.

Se realizó un análisis de la potencia necesaria para transmitir por *bluetooth low energy* y utilizando ESP-NOW con los valores proporcionados por el fabricante en la hoja de datos el módulo ESP32, esta información se encuentra detalla en la tabla 3.3. Si comparamos una trasmisión por BLE contra una transmisión por *WiFi*, la corriente necesaria es aproximadamente el doble para este último. En contraparte, el tiempo de recuperación de conexión desde el modo *sleep*, para el protocolo ESP-NOW es de 31 ms, mientras que para BLE es de 1.3 s. Analizando los tiempos de transmisión para 16 *bytes*, el protocolo BLE en modo estrella se demora 1056 μs mientras que el ESP-NOW 380 μs. Por lo tanto, la energía necesaria para despertar al MCU, realizar una conexión y transmitir la información necesaria es idealmente veinte y dos veces menor para ESP-NOW. Otro dato importante para comparar es el alcance de ambos protocolos, el alcance práctico aproximado de una red estrella del tipo BLE es de 120 m, para ESP-NOW es de aproximadamente 160 m [24].

TABLA 3.3. Corriente del módulo ESP32 detallada por el fabricante para cada modo

Modo	Mínimo	Típico	Máximo	Unidad
Transmit 802.11b, DSSS 1 Mbps, Pout = +19.5 dBm	-	240	-	mA
Transmit 802.11g, OFDM 54 Mbps, Pout = +16 dBm	-	190	-	mA
Transmit 802.11n, OFDM MCS7, Pout = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 $\sim$ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 $\sim$ 100	-	mA

Con los detalles sobre el consumo de energía, el alcance de ambos protocolos y la lógica del modo *sleep* detallada en la Sección 3.2.4, el protocolo ESP-NOW es el más adecuado para el desarrollo del proyecto.

Para sumar una capa de seguridad extra a la clave PMK de ESP-NOW, el máster filtra la información según la MAC[39] del dispositivo recibido, permitiendo únicamente admitir datos de los diez esclavos guardados en memoria.

3.2.6. Servidor web máster y esclavo

El módulo máster necesita de varios parámetros para funcionar correctamente: nombre y clave de acceso al *WiFi*, configuraciones del módulo GPRS dependientes de la compañía celular a utilizar, un número de celular de reporte inmediato en caso de alarma (detallado en la sección 3.2.8) y las MACs de los esclavos permitidos. Para que el programa sea flexible y permita la modificación de estos parámetros sin necesidad de reprogramar nuevamente el MCU, se decidió implementar un servidor web para la modificación de los mismos, el código en HTML [40] del mismo se observa en el Código 3.1. Se debe reiniciar el módulo por medio del pulsador S2 mencionado en la sección 3.1.4, al momento de liberarlo, presionar por tres segundos el pulsador S1 y el MCU generará un *WiFi* específico para el servidor web.

El archivo que contiene estos parámetros de configuración se guarda en la memoria *flash* del MCU, que es del tipo no volátil, para no perder la configuración si el MCU queda sin alimentación.

```
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html><head>
  <title>ESP Input Form</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script>
    function submitMessage() {
      alert("PARAMETRO GUARDADO EN MEMORIA");
      setTimeout(function(){ document.location.reload(false); }, 500);
    }
  </script></head><body>
  <h1>CONFIGURACION MODULO MASTER</h1><br>

  <h3>          WIFI</h3>

  <form action="/get" target="hidden-form">
    Red WiFi acceso internet (valor actual "%wifi_int_name%"): <input type="text"
    name="wifi_int_name">
    <input type="submit" value="Guardar" onclick="submitMessage()">
  </form><br>
  <form action="/get" target="hidden-form">
    Clave de red (valor actual "%wifi_int_pass%"): <input type="text" name="
    wifi_int_pass">
    <input type="submit" value="Guardar" onclick="submitMessage()">
  </form><br>

  <h3>          GPRS</h3>

  <form action="/get" target="hidden-form">
    GPRS APN (valor actual "%apn%"): <input type="text" name="apn">
    <input type="submit" value="Guardar" onclick="submitMessage()">
  </form><br>
  <form action="/get" target="hidden-form">
    GPRS user (valor actual "%gprsUser%"): <input type="text" name="gprsUser">
    <input type="submit" value="Guardar" onclick="submitMessage()">
```

[39] MAC es el acrónimo en inglés de *media access control*, también conocido como dirección física, y es única para cada dispositivo.

[40] HTML es el acrónimo en inglés de *HyperText markup language*, lenguaje de marcas de hipertexto.

```
33  </form><br>
34  <form action="/get" target="hidden-form">
35    GPRS pass (valor actual "%gprsPass%"): <input type="text" name="gprsPass">
36    <input type="submit" value="Guardar" onclick="submitMessage()">
37  </form><br>
38  <form action="/get" target="hidden-form">
39    SMS celular (ej: 5491145671234) (valor actual "%SMS_phone_number%"): <input type
      ="text" name="SMS_phone_number">
40    <input type="submit" value="Guardar" onclick="submitMessage()">
41  </form><br>
42
43  <h3>          THINGER.io</h3>
44
45  <form action="/get" target="hidden-form">
46    Thinger.io user name (valor actual "%thinger_user_name%"): <input type="text"
      name="thinger_user_name">
47    <input type="submit" value="Guardar" onclick="submitMessage()">
48  </form><br>
```

CÓDIGO 3.1. Código HTML del servidor web módulo máster.

A su vez, para verificar el correcto funcionamiento de un módulo esclavo, estos poseen también un servidor web básico que informa la lectura de cada entrada. Al igual que para el máster, se debe reiniciar el MCU y arrancarlo en modo servidor web. Al conectarse al *WiFi* que genera y acceder desde un navegador al ip 192.168.1.100, se observa la información igual que en la Figura 3.11.

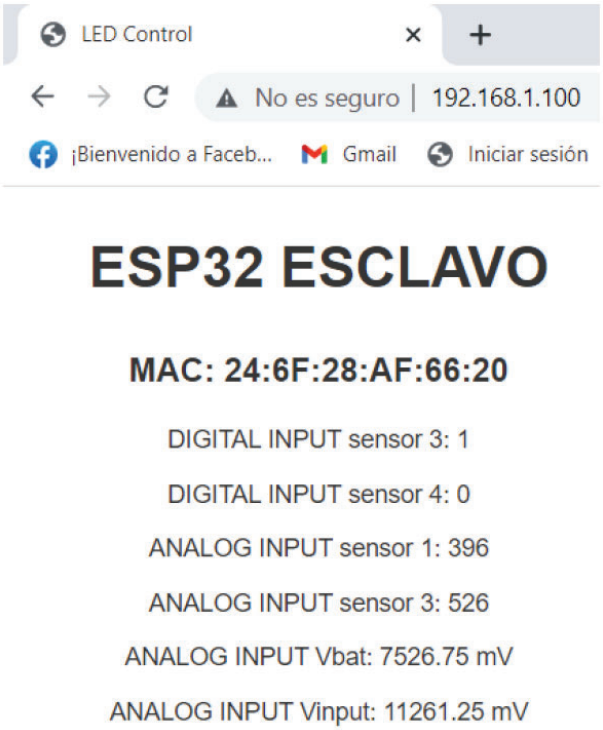

FIGURA 3.11. Servidor web de un módulo esclavo.

3.2.7. Interacción con Thinger.io

Como se indicó en la sección 2.4.3, el portal *Thinger.io* ofrece bibliotecas específicas para diferentes microcontroladores o placas de desarrollos, entre ellos el ESP32 utilizado en este proyecto. La biblioteca incluye toda la etapa de comunicación con el *backend* del portal. Para poder enlazar con este, se deben configurar los parámetros de usuario, contraseña y credencial de acceso por medio de una función. Además, se debe configurar la conexión de *WiFi* y GPRS para que la biblioteca tenga acceso a las funciones específicas. Estas funciones fueron editadas en la librería original debido a que no permitían ser invocadas si se les asignaba una variable, únicamente aceptaban punteros constantes del tipo *const char* *. En el Código 3.2 se pueden leer fragmentos del programa, se encuentran únicamente las declaraciones de datos a recibir en el *backend* del máster y el esclavo 0, las variables utilizadas al invocar las funciones se toman del archivo de configuración nombrado en la sección 3.2.6.

```
####Declaracion funcion thing####
ThingerESP32 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);
####Declaracion datos master####
thing["MASTER_0_inputs"] >> [](pson& out){
  out["sensor1"] = inputs_data.sensor1;
  out["sensor2"] = inputs_data.sensor2;
  out["sensor3"] = inputs_data.sensor3;
  out["sensor4"] = inputs_data.sensor4;
  out["vbat_mV"]    = inputs_data.Vbat_mV;
  out["vinput_mV"]  = inputs_data.Vinput_mV;
};
thing["MASTER_0_out_rele"] << inputValue(outputs_data.rele);
thing["MASTER_0_out_pwm"] << inputValue(outputs_data.pwm);
thing["MODO_ALARMA"] << inputValue(alarm_ON);
####Declaracion datos slave 0####
thing["SLAVE_0_inputs"] >> [](pson& out){
  out["temperatura"] = inputs_data_slaves[0].temp;
  out["presion"] = inputs_data_slaves[0].pres;
  out["sensor1"] = inputs_data_slaves[0].sensor1;
  out["sensor2"] = inputs_data_slaves[0].sensor2;
  out["sensor3"] = inputs_data_slaves[0].sensor3;
  out["sensor4"] = inputs_data_slaves[0].sensor4;
  out["vbat_mV"]    = inputs_data_slaves[0].Vbat_mV;
  out["vinput_mV"]  = inputs_data_slaves[0].Vinput_mV;
};
thing["SLAVE_0_out_rele"] << inputValue(outputs_data_slaves[0].rele);
thing["SLAVE_0_out_pwm"] << inputValue(outputs_data_slaves[0].pwm);
####Configuracion WiFi y GPRS Thinger.io####
gsm_gprs_begin(modem);
thing.add_wifi(&wifi_thing_name[0], (char*)&wifi_thing_pass[0]);
```

CÓDIGO 3.2. Código de configuración Thinger.io

Luego de finalizada la configuración, hay una tarea que debe ser invocada en el ciclo de programa. *Thing.handle()*, es la tarea que invoca diferentes instancias de la biblioteca, las cuales envían los datos tipo *out* (información de las entradas del MCU) a la plataforma; y de recibir datos tipo *inputValue* desde la plataforma, para guardarlos en variables que serán cargadas en las salidas de los diferentes módulos.

3.2.8. Modo alarma y notificaciones

El módulo máster en su salida de *relay* posee conectada una sirena de 110 dB. La misma se activa automáticamente por la lógica del módulo máster, si se desea, puede ser activada

manualmente desde el portal. La activación automática ocurre únicamente si el módulo máster se encuentra configurado en "modo alarma", esta función está embebida directamente en el MCU y no es dependiente del portal.

El servidor web de configuración admite hasta 10 esclavos. En los esclavos del 0 al 4, si se activa alguna de las entradas de sensores PIR o CON4 detalladas en la Figura 3.6 y el modo alarma se encuentra activo, la sirena se activará por cinco minutos o hasta que se cancele manualmente desde el portal. Además, se enviará un mensaje de texto al teléfono celular configurado y un correo electrónico de activación de alarma al destinatario guardado. Estas funciones fueron programadas en bajo nivel dentro del MCU en vez de ser ejecutadas por el portal, para asegurar su funcionamiento en caso de ausencia de conexión a internet, tanto de la conexión por *WiFi* como por GPRS.

Sumado al modo mencionado anteriormente, el dispositivo enviará notificaciones si pierde o recupera la conexión a internet por WiFi, lo mismo ocurre con la energía de red.

3.3. Portal usuario

3.3.1. Modelo de estructura de datos

Para la organización de la estructura de datos se adoptó el modelo programado en el Código 3.3. Se aprecia la declaración de las variables que almacenan la información de las entradas, también los valores calculados en mV de los voltajes de entrada y batería. Este modelo se adoptó tanto para los esclavos como para el máster, es idéntico al que se observará en la estructura de datos del portal.

```
####estructura datos de entrada####
struct inputs_struct
{
      float temp;
      uint32_t pres;
      bool sensor3;
      bool sensor4;
      uint32_t sensor1;
      uint32_t sensor2;
      uint32_t Vbat;
      uint32_t Vinput;
      float Vbat_mV;
      float Vinput_mV;
};
####estructura datos de salida####
struct outputs_struct
{
      bool rele_1;
      bool rele_2;
      uint8_t pwm;
};
```

CÓDIGO 3.3. Estructuras de datos

3.3.2. Tablero de visualización

El alcance del proyecto es hasta el enlace de la información con el portal, el mismo no incluyó el desarrollo del *dashboard* de visualización y control. Como se mencionó en la Sección 2.4.3, Thinger.io ofrece un entorno amigable y de fácil configuración. La plataforma nos ofrece tableros de apariencia editable y de rápida configuración, para indicadores de tipo texto, aguja

FIGURA 3.12. Dashboard de ejemplo de la plataforma Thinger.io. Imagen tomada de [9].

o barra, gráficos y control de dispositivos por medio de llaves o pulsadores. Esta información puede ser administrada en tiempo real o consumida desde una pila configurable; permite separar el *dashboard* en diferentes pestañas, asignarle nombre, símbolos y colores de fondo a cada *widget*. En la Figura 3.12 podemos observar un *dashboard* de ejemplo que nos enseña el potencial del portal.

Capítulo 4

Ensayos y resultados

En esta sección se presenta el *hardware* físico del proyecto, los resultados obtenidos del mismo y el análisis de su desempeño. Se verifica el cumplimiento de los requisitos, se exponen los resultados y se presentan los problemas ocurridos en la instalación.

4.1. Circuitos impresos desarrollados

La placa que se observa en la Figura 4.1 pertenece al módulo máster, en la parte superior se observa la etapa de carga de batería y fuente, en el centro podemos observar al MCU con todos los componentes periféricos y en la zona inferior se observa en módulo LTE-IoT-2-click conectado en las tiras de pines. Complementariamente en la Figura 4.2 observamos un módulo

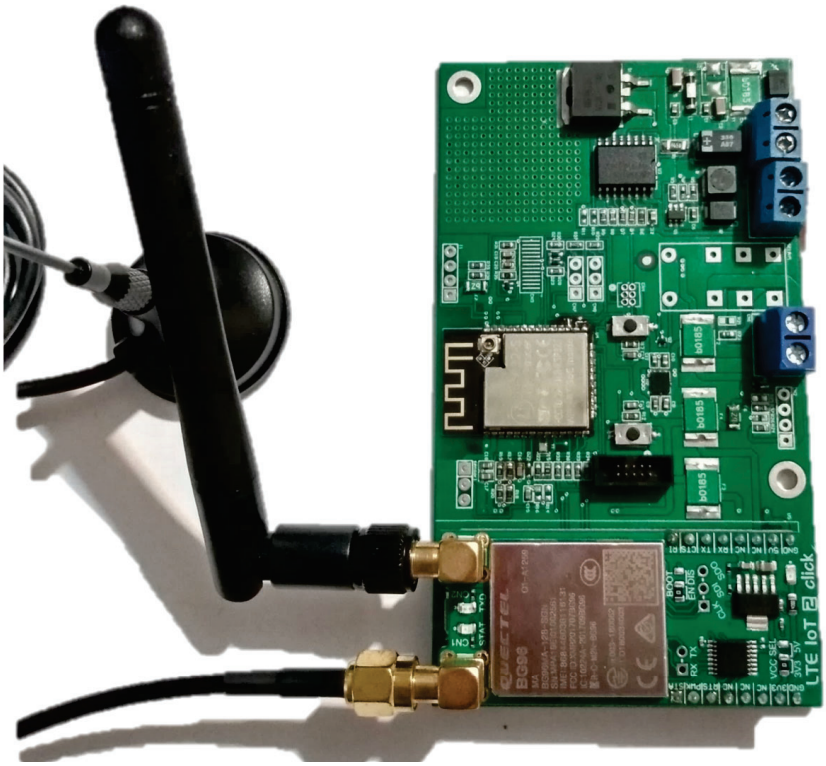

FIGURA 4.1. Placa montada del módulo master

esclavo de los 5 que se fabricaron para el proyecto. Con estas placas se realizaron los ensayos y validaciones del diseño. Comenzar el trabajo de programación con las placas máster y esclavo ya montadas logró evitar fallas eléctricas típicas de un protoboard, como falsos contactos o errores en los protocolos de comunicación. La única diferencia entre las placas esclavo y el

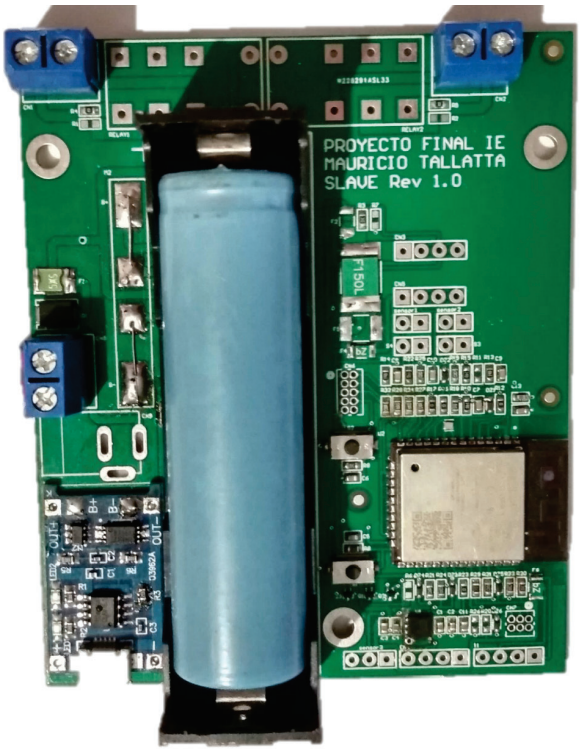

FIGURA 4.2. Placa montada de un módulo esclavo

archivo Altium del diseño, es la conexión al puerto UART de programación, los pines Tx y Rx se encuentran invertidos por un error durante la etapa de diseño. Para mitigar esta falla en el desarrollo, se cortaron las pistas y se invirtieron con *wirewrap*.

4.1.1. Gabinetes y piezas 3D

Para minimizar el tamaño de los módulos esclavo se diseñó un gabinete a medida para la instalación de la placa y el sensor PIR. El mismo se observa en la Figura 4.3, está fabricado en impresión 3D de filamento de PLA [41]. Para el módulo máster se utilizó un gabinete estanco de

FIGURA 4.3. Gabinete módulo esclavo.

165x165x110 mm para almacenar la batería y la placa. Para mantener un aspecto similar entre

[41] Acido poliláctico, un polímero sintético.

esclavo y máster, en este último se instaló una tapa del gabinete esclavo. En la Figura 4.4 se observa la instalación incluyendo la sirena de alarma.

FIGURA 4.4. Instalación módulo máster.

Para el jardín se fabricó un módulo esclavo que contiene un sensor de lluvia, sumado a esto, permite medir la humedad de la tierra y posee una llave de agua comandada a través de un servomotor de 5 V, diseñada para la aplicación y fabricada nuevamente por impresión 3D, el diseño se observa en la Figura 4.5.

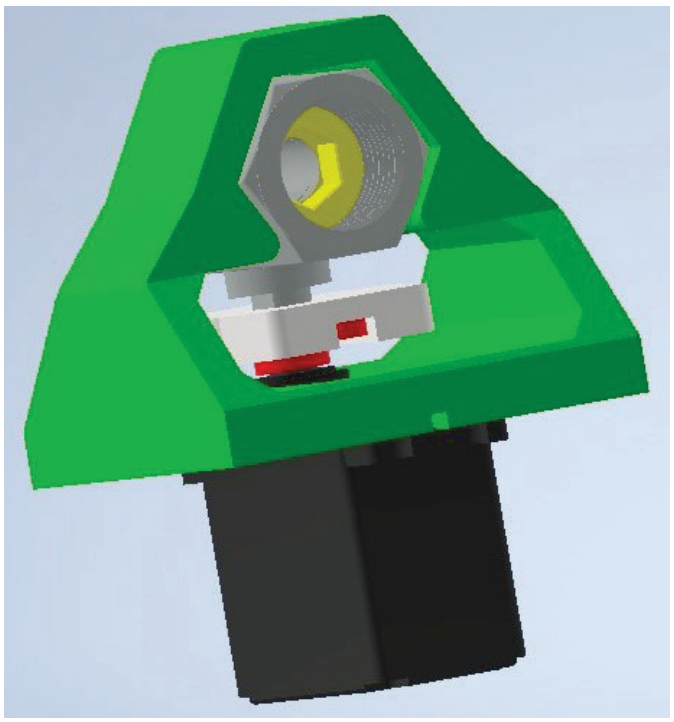

FIGURA 4.5. Diseño 3D de la llave accionada por servomotor.

Al poseer alimentación solar un comando por electroválvula queda descartado por su gran consumo de energía, se optó por esta solución ya que tiene un consumo de energía considerable únicamente en el momento de accionamiento, en el resto del tiempo solo hay un corriente mínima para la lógica de control del servo. En la Figura 4.6 podemos observar el módulo del

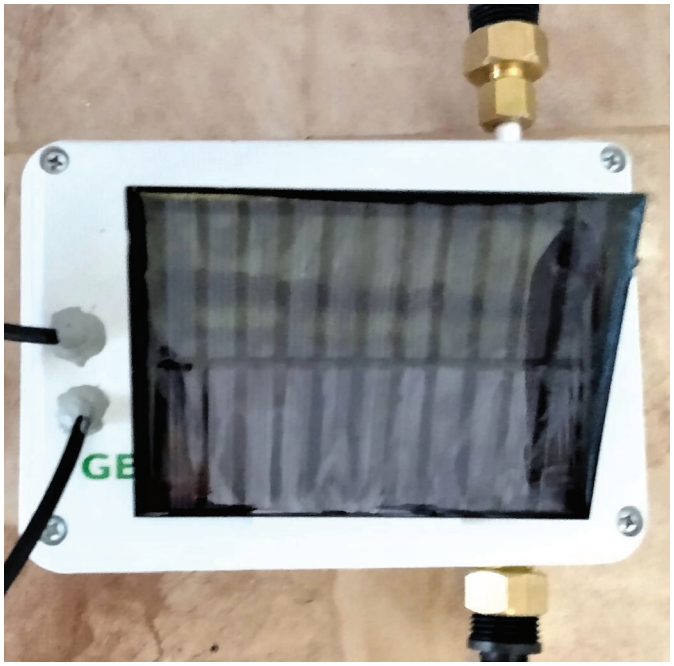

FIGURA 4.6. Gabinete módulo esclavo jardín.

jardín con su panel solar instalado, respetando el ángulo de inclinación apropiado para la latitud donde fue instalado[25]. En la zona superior e inferior del lado izquierdo se observa la salida roscada para la conexión de agua de la llave comandada.

4.2. Ensayos de desarrollo

4.2.1. Ensayos de alcance

Los ensayos de alcance máximo no se realizaron en la zona de instalación, sino donde se desarrolló el proyecto. Para la prueba, el módulo máster se ensayó con la antena *on board* que trae el módulo ESP32, mientras que en el esclavo se conectó una antena externa de 4 dBi. Colocando la placa máster sobre una mesa de madera y en una esquina de la casa donde no hay ventanas cercanas, se procedió a caminar por la calle alejándose de la misma. Continuamente se mantuvo una visión directa entre el módulo esclavo y la esquina de la casa donde se encontraba el máster. Una vez que se perdía la señal con el máster, se reducía la distancia hasta volver a obtener el enlace de datos ESP-NOW de forma correcta; la distancia de reconexión en todos los casos fue menor a la de pérdida de señal, siendo esta última la registrada para la prueba en 84 metros. En la Figura 4.7 se observa una imagen satelital de la la zona con la medición incluida.

4.2.2. Verificación de carga y consumo

Para los ensayos de carga y consumo se utilizó un multímetro Fluke 189 calibrado, facilitado por la empresa Colven. Se realizaron las mediciones para verificar el correcto funcionamiento de lo planteado en la sección 3.1.

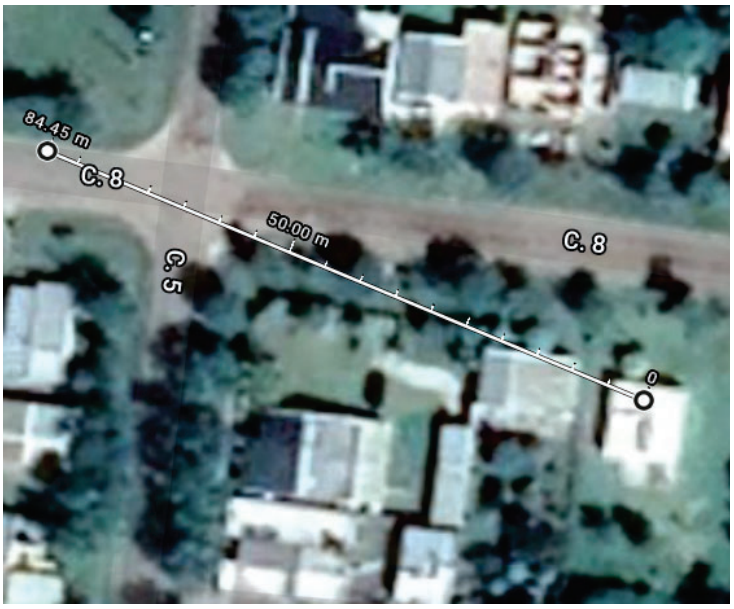

FIGURA 4.7. Distancia máxima obtenida

Respecto al módulo máster, con un voltaje de fuente de 18,05 V se obtuvo un voltaje máximo en la batería de 12,97 V, con una corriente de carga de 260 mA, estos valores se condicen con lo calculado en la sección 3.1.1. Desconectando la fuente de energía, la batería entregaba al circuito una corriente de 39 mA, con el módulo GPRS desenergizado y sin la salida activa, mientras que activando el módulo GPRS y eliminando la conexión a internet del WiFi, para obligar al máster a transmitir por GPRS, entregaba una corriente de 164 mA. Estos valores son menores a los calculados, se puede atribuir esta diferencia a que la cantidad de datos que el proyecto envía por segundo es menor a la que se utilizó en los valores de referencia.

Enfocándonos en el módulo esclavo, la corriente suministrada desde la batería al MCU en el modo dormido fue de 3,52 mA, mientras que durante una activación de un sensor y mientras se transmitía al máster fue de 230 mA. Nuevamente observamos un menor valor en el momento de la transmisión pero no así en el modo dormido. Esta diferencia de mayor consumo se debe a la alimentación del sensor PIR, la corriente derivada para las mediciones ADC y la protección de la batería, que no fueron tenidas en cuenta en el cálculo presentado en la sección 3.2.5. A pesar de esta diferencias, se obtiene una duración ideal de la batería de 5,8 días, la cual sigue siendo mayor a la duración del máster.

En el módulo del jardín se verificó la carga del panel solar. La medición se realizó a mitad de Julio, se verificó a las 11, a las 12 y a las 14 horas de un día de pleno sol. Se obtuvo una medición de 117 mA, 150 mA y 132 mA respectivamente, por lo que para realizar una carga de batería completa se necesitarían en promedio 18,79 horas de sol a una carga de 133 mA.

4.2.3. Instalación en el lugar

La casa donde se instaló se compone de 3 edificaciones: la casa principal, un anexo (garage y habitación) y un galpón. En el diagrama de instalación de la Figura 4.8 se puede observar una foto satelital de la zona. La estrella amarilla y negra es el punto donde se instaló el módulo máster, como se observó en la Figura 4.4. La distancia al galpón marcada con la flecha amarilla es de 36 m, al anexo marcada con la flecha roja existe una distancia de 12 m y por último a la zona del jardín, marcado en azul, hay 15 m.

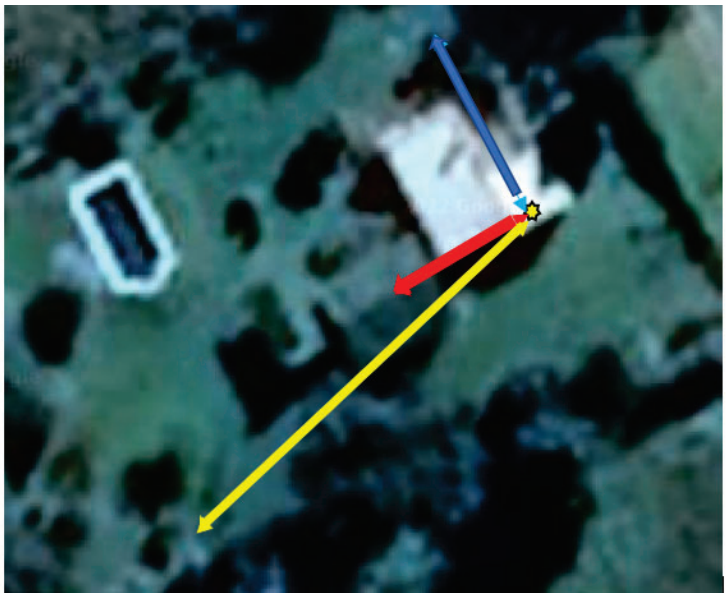

FIGURA 4.8. Mapa satelital de instalación

4.2.4. Instalación y funcionamiento

Inicialmente se ensayó toda la instalación con el módulo máster en otro punto de instalación más centrado en la propiedad, pero no había recepción de señal de todos los esclavos, en especial en el galpón ya que estaba construido íntegramente de chapa, reduciendo sustancialmente la transmisión del módulo. La dueña de la casa comentó que las paredes tenían una gran cantidad de hierro, porque el anterior dueño así la construyó, no es el caso de la zona de instalación (estrella amarilla y negra), que es una ampliación reciente sin hierro en las paredes, esto facilitó la comunicación por ESP-NOW.

Otro problema de funcionamiento ocurrió con los sensores PIR, al ensamblar todo el proyecto completo comenzó una falla aleatoria de actuación de los sensores. Al acercar los sensores al módulo ESP32, los mismos comenzaban a fallar aleatoriamente indicando movimiento de forma errónea. Como solución inicial se les realizó una protección completa en cinta de aluminio, conectando la misma a masa, esta modificación redujo la falla pero la misma continuó. La solución final fue lograda por *software*, se implementó la lógica observada en el Código 4.1, los contadores *PIR_ticks* y *ticks_to_reset_PIR* se incrementan por timer, *time_filter_ticks* y *filter_counter* son dos umbrales definidos para el filtro. Esta lógica ralentiza el funcionamiento del sensor pero filtra las activaciones erróneas.

```
1  if(gpio_get_level(GPIO_INPUT_IO_1)){
2      PIR_counter++;
3    }
4    if(PIR_ticks >= time_filter_ticks){
5      PIR_ticks = 0;
6      if(PIR_counter >= filter_counter){
7        inputs_data_read.sensor3 = 1;
8        ticks_to_reset_PIR = 0;
9      }
10     PIR_counter = 0;
11   }
12   if((inputs_data_read.sensor3) & (ticks_to_reset_PIR >= time_to_reset_PIR)){
13     inputs_data_read.sensor3 = 0;
14   }
```

CÓDIGO 4.1. Lógica de corrección para sensor PIR

Finalmente, se verificó la correcta recepción de los datos de los esclavos y del máster en el portal *Thinger.io*, también se ensayó el modo alarma y la activación de la sirena. Se configuró toda la información necesaria a través del servidor web y se ensayó el envío de SMS, correos electrónicos y el cambio de conexión de internet desde *WiFi* a GPRS, todas las funciones fueron satisfactorias.

Capítulo 5

Conclusiones

5.1. Conclusiones generales

El proyecto desarrollado Master-Esclavo, de acuerdo con el objetivo general, informa los cambios de estados de los sensores en caso de presencia. Complementando el objetivo general y el requerimiento planteado por el cliente, se logró un desarrollo exitoso del proyecto, tanto de *hardware* como de *firmware*, escogiendo una plataforma IoT adecuada para la interfaz con el proyecto. La plataforma funcionó adecuadamente durante la etapa de desarrollo, instalación y período de pruebas con el cliente, validando el análisis previo realizado antes de seleccionarla. La documentación en línea y bibliotecas de implementación de *Thinger.io* colaboraron para acortar en buena manera los tiempos de desarrollo e implementación.

Comenzar el desarrollo en un entorno de programación profesional como el *Visual Studio Code* facilitó el trabajo debido a la posibilidad de depurar el código en el MCU ESP-32, que otros entornos de desarrollo no lo permiten.

El tiempo de desarrollo no fue el planeado para el proyecto y se debieron realizar cambios importantes en el mismo, debiéndose en parte a la pandemia sufrida durante los dos años de desarrollo y por otra parte las políticas sobre tecnología del país, que son ajenas al proyecto. El reemplazo que se realizó en la comunicación celular, la sustitución de NB-IoT por GPRS conllevaron a que el punto más importante del proyecto no sea innovador, esta modificación fue inevitable.

Los ensayos de consumo en el modo *sleep* fueron acertados con los cálculos, la autonomía planteada inicialmente se logró tanto para el módulo máster como para el esclavo.

La experiencia laboral previa fue fundamental para el diseño y desarrollo de los circuitos impresos, se logró fabricar para el máster como para los esclavos, una única versión de placa que contempló diferentes variantes de montaje, desempeñándose con éxito en cada aplicación. Sumado a esto, la fabricación temprana logró mitigar fallas de falso contacto que hubieran ocurrido utilizando los módulos de desarrollo cableados, acelerando la etapa de programación.

El acceso a compras en el exterior del país por parte de la empresa COLVEN S.A. facilitaron la adquisición de circuitos impresos y semiconductores, reduciendo drásticamente el costo de desarrollo del proyecto y mitigando el gran problema de abastecimiento mundial en el mercado de semiconductores.

5.2. Trabajo a futuro

Con el cumplimiento de los requerimientos y la finalización del proyecto, se han detectado oportunidades de mejora a futuro en el desarrollo de *hardware*, *firmware* y la implementación en el portal, tales como:

- OTA o *Firmware over the air*, esta funcionalidad permitirá actualizar de manera remota el *firmware* tanto para el módulo máster como de los módulos esclavos. Las nuevas versiones de *firmware* posibilitan corregir errores de funcionamiento, mejorar funciones

desarrolladas o implementar nuevas funcionalidades. Esta mejora es indispensable si del proyecto presentado se desea crear un producto comercial, admitiría la actualización en cadena de todos los módulos instalados.

- Segmentación del *hardware*, los módulos esclavos fueron desarrollados de manera de facilitar la fabricación para el proyecto, al momento de lanzar un producto deberían existir módulos esclavos específicos para diferentes tipos de entradas o salidas, esto permitirá reducir sustancialmente la dimensión.
- Circuitos impresos con componentes en ambos lados, sumado a la segmentación del *hardware* colaborará en la reducción del tamaño.
- Ultra bajo consumo, los módulos que únicamente implementen entradas, si se realiza un diseño de circuito de ultra bajo consumo y un *firmware* con una lógica de interrupción únicamente por activación de sensores, permitiría módulos que funcionen a pila en vez de baterías.
- Terminal Thinger.io, esta función permitiría observar a bajo nivel que ocurre en cada máster y esclavo conectado al portal, con la posibilidad de realizar un diagnóstico de fallas remoto, activar o desactivar funciones y realizar las configuraciones de funcionamiento directamente desde el portal.

Bibliografía

[1] Statista. *Dispositivos conectados (Internet de las cosas) a nivel mundial de 2019 a 2030*. Disponible: 31/8/2022. URL: https://es.statista.com/estadisticas/517654/prevision-de-la-evolucion-de-los-dispositivos-conectados-para-el-internet-de-las-cosas-en-el-mundo/.

[2] Sparkfun. *ESP32*. Disponible: 31/8/2022. URL: https://cdn.sparkfun.com/assets/home_page_posts/2/0/1/7/esp-wroom-03-top_bottom.jpg.

[3] espressif. *ESP32 Series Datasheet*. Disponible: 2022-09-14. URL: https://espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[4] MIKROE. *LTE IOT 2 CLICK*. Disponible: 2022-09-14. URL: https://www.mikroe.com/lte-iot-2-click.

[5] Unitrode Products from Texas Instruments. *Sealed Lead-Acid Battery Charger*. Disponible: 2022-09-14. URL: https://www.ti.com/lit/gpn/uc3906.

[6] Starware. *Módulo de carga genérico TP4056*. Disponible: 31/8/2022. URL: https://tienda.starware.com.ar/producto/modulo-carga-generico-tp4056-micro-usb-5v-1a-bateria-litio-li-ion/.

[7] BOST HIGH. *How Does Mesh Network Allow IoT Devices To Communicate?* Disponible: 31/8/2022. URL: https://boosthigh.com/mesh-network-for-iot-devices/.

[8] Thinger.io. *Overview*. Disponible: 31/8/2022. URL: https://docs.thinger.io/.

[9] Jansutris Apriten Purba. *Open Source Internet of Things (IoT) Platforms*. Disponible: 31/8/2022. URL: https://jansutris10.medium.com/open-source-internet-of-things-iot-platforms-9afd187465ca.

[10] ¿Qué es IoT? *Amazon*. Disponible: 31/8/2022. URL: https://aws.amazon.com/es/what-is/iot/.

[11] SAS. *Transformación digital*. Disponible: 2022-09-14. URL: https://www.sas.com/es_ar/insights/big-data/internet-of-things.html.

[12] SAP. *¿Que es IoT y cómo funciona?* Disponible: 2022-09-14. URL: https://www.sap.com/latinamerica/insights/what-is-iot-internet-of-things.html.

[13] espressif. *ESP32 Technical Reference Manual*. Disponible: 2022-09-14. URL: https://espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

[14] Acoptex. *Basics: Project 082a Lithum battery charger TP4056*. Disponible: 2022-09-14. URL: https://acoptex.com/project/9446/basics-project-082a-lithum-battery-charger-tp4056-at-acoptexcom/.

[15] espressif. *vscode-esp-idf-extension*. Disponible: 2022-09-14. URL: https://github.com/espressif/vscode-esp-idf-extension.

[16] Espressif Systems. *ESP-NOW*. Disponible: 2022-09-14. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html.

[17] GEEKFLARE. *11 plataformas y herramientas de Internet de las cosas (IoT) de código abierto*. Disponible: 2022-09-14. 2016. URL: https://geekflare.com/es/iot-platform-tools/.

[18] Alpha and Omega Semiconductor. *AOZ1282CI*. Disponible: 2022-09-14. URL: http://aosmd.com/res/data_sheets/AOZ1282CI.pdf.

[19] Texas Instruments. *PCB thermal calculator*. Disponible: 2022-09-14. URL: https://www.ti.com/design-resources/design-tools-simulation/models-simulators/pcb-thermal-calculator.html.

[20] Onsemi. *LDO Regulator* -. Disponible: 2022-09-14. URL: https://www.onsemi.cn/pdf/datasheet/ncp605-d.pdf.

[21] Espressif Systems. *ESP-Prog*. Disponible: 2022-09-14. URL: https://docs.espressif.com/projects/espressif-esp-dev-kits/en/latest/other/esp-prog/user_guide.html#hardware-reference.

[22] ECS Inc International. *ECX-1210*. Disponible: 2022-09-14. URL: https://ecsxtal.com/store/pdf/ECX-1210.pdf.

[23] Rohm Semiconductor. *DTC015E series*. Disponible: 2022-09-14. URL: hhttps://fscdn.rohm.com/en/products/databook/datasheet/discrete/transistor/digital/dtc015eebtl-e.pdf.

[24] Neupane, Sizen. «A Comparative study of Wireless Star Networks Implemented with Current Wireless Protocols». Tesis de mtría. 2019. URL: https://scholarworks.gvsu.edu/theses/920/.

[25] Openhacks. *Solar Panel Unit - 5V/220mA/1.1W/PET*. Disponible: 2022-09-14. URL: https://www.openhacks.com/page/productos/id/2991/title/Solar-Panel-Unit---5V-220mA-1.1W-PET#.YyJyoXbMJPZ.