

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

Sistema de monitoreo de cultivos agrícolas

Autor:

Ing. Mario Fernando Aguilar Montoya

Director:

Esp. Ing. Julian Bustamante Narvaez (TECREA SAS)

Jurados:

Dr. Ing. Javier Andrés Redolfi (UTN-FRSF)
Mg. Lic. Leopoldo Zimperz (FIUBA)
Esp. Ing. Felipe Calcavecchia (FIUBA)

*Este trabajo fue realizado en la ciudad de Tarija,
entre junio de 2022 y diciembre de 2023.*

Resumen

En la presente memoria se aborda el diseño e implementación de un sistema de adquisición de datos para el monitoreo de cultivos agrícolas realizado como emprendimiento personal. Este trabajo pretende ayudar al agricultor a gestionar de mejor manera sus recursos. Para su desarrollo fueron fundamentales los conocimientos adquiridos en la carrera tales como conceptos de modularización, testing de software, sistemas operativos en tiempo real, protocolos de comunicación y programación de microcontroladores.

Agradecimientos

En primer lugar a mi familia y amigos que siempre me brindan su apoyo y confianza.

A mi director por su paciencia, guía y consejo en todo momento.

A mis compañeros y maestros de la CESE, por los aportes y enseñanzas que me dieron a lo largo de la especialización.

Índice general

Resumen	I
1. Introducción general	1
1.1. Internet de las cosas en la agricultura	1
1.1.1. Internet de las cosas	1
1.1.2. Sistemas de monitoreo de cultivos agrícolas	2
1.2. Estado del arte	2
1.2.1. Libelium	2
1.2.2. Nodo RF-M1 DropControl	3
1.3. Objetivo y alcances	4
1.3.1. Objetivo	4
1.3.2. Alcances	4
2. Introducción específica	5
2.1. Componentes principales de hardware	5
2.1.1. Plataforma de desarrollo STM32 NUCLEO-L432KC	5
2.1.2. Módulo de comunicación LTE IOT 2 CLICK	6
2.1.3. Sensor AHT10	7
2.1.4. Sensor ML8511	7
2.1.5. Sensor de humedad de suelo HL-69 (Resistivo)	7
2.2. Herramientas de software y testing utilizados	8
2.2.1. STM32 CubeIDE	8
2.2.2. FreeRTOS	8
2.2.3. CEDLING	9
2.3. Protocolos de Comunicación	9
2.3.1. UART	9
2.3.2. I2C	9
2.3.3. MQTT	9
2.4. Plataformas IoT	10
2.4.1. Ubidots	10
2.4.2. ThingsBoard	10
3. Diseño e implementación	11
3.1. Diagrama de bloques general del sistema	11
3.2. Arquitectura de firmware	12
3.2.1. Capa HAL	12
3.2.2. Capa drivers	13
3.2.3. Capa aplicación	13
3.3. Desarrollo del firmware	14
3.3.1. Tarea loop del sistema	14
3.3.2. Tarea adquisición de datos	15
3.3.3. Tarea manejador de alarmas	16
3.3.4. Tarea manejador del servidor	17

3.4.	Controladores implementados	18
3.4.1.	Controlador para el sensor AHT10	18
3.4.2.	Controlador para el módulo de comunicación BG96	19
3.5.	Desarrollo del hardware	21
3.5.1.	Esquemático	21
3.5.2.	PCB del hardware	23
3.5.3.	Fabricación del circuito impreso	24
3.6.	Paneles de visualización	24
3.6.1.	Panel principal	24
3.6.2.	Panel nodo sensor	25
4.	Ensayos y resultados	27
4.1.	Pruebas unitarias	27
4.2.	Pruebas de la plataforma IoT	29
4.2.1.	Prueba de inyección de mensajes	29
4.2.2.	Prueba del widget de mapa	30
4.2.3.	Prueba de la tabla de alarmas del panel visualización	31
4.2.4.	Prueba de persistencia de datos	31
4.3.	Pruebas de hardware	32
4.3.1.	Prueba de comunicación entre el sensor AHT10 y el micro-controlador	32
4.3.2.	Prueba de comunicación entre el módulo BG96 y el micro-controlador	33
4.4.	Pruebas funcionales del sistema	33
4.4.1.	Pruebas de lectura de los sensores	34
	Caso de uso 1	34
	Caso de uso 2	34
4.4.2.	Prueba del envío de datos al broker MQTT	34
4.4.3.	Prueba de envío de alarmas por SMS	35
5.	Conclusiones	37
5.1.	Resultados obtenidos	37
5.2.	Próximos pasos	38
	Bibliografía	39

Índice de figuras

1.1. Módulo Smart Agriculture PRO.	3
1.2. Módulo RF-M1 de DropControl.	3
2.1. Plataforma de desarrollo NUCLEO-L432KC ¹	6
2.2. Módulo LTE IOT 2 CLICK ²	6
2.3. Sensor AHT10 ³	7
2.4. Módulo Sensor ML8511	7
2.5. Módulo sensor HL-69 ⁴	8
2.6. Logo FreeRTOS ⁵	8
2.7. Arquitectura de publicación/suscripción de MQTT ⁶	9
2.8. Ejemplo interfaz gráfica Ubidots ⁷	10
2.9. Ejemplo interfaz gráfica ThingsBoard ⁸	10
3.1. Diagrama general del sistema IoT.	11
3.2. Capas del firmware.	12
3.3. Diagrama de flujo de inicialización del firmware.	14
3.4. Diagrama de flujo de la tarea loop.	15
3.5. Diagrama de flujo tarea de adquisición de datos.	16
3.6. Diagrama de flujo de la tarea manejador de alarmas.	16
3.7. Diagrama de flujo de la tarea conexión server MQTT.	17
3.8. Máquina de estados down servidor.	17
3.9. Máquina de estados up servidor.	18
3.10. Esquemático página raíz.	22
3.11. Conector módulo BG96 y NUCLEO-L432KC.	22
3.12. Esquemático interfaz de debug.	22
3.13. Esquemático conectores sensores.	23
3.14. Esquemático conectores de alimentación.	23
3.15. PCB del trabajo.	23
3.16. Modelo 3D de la tarjeta.	24
3.17. PCB ensamblado.	24
3.18. Panel principal de la interfaz gráfica.	25
3.19. Panel nodo sensor.	26
4.1. Resultados de los tests realizados a los drivers.	28
4.2. Informe de cobertura driver AHT10.	29
4.3. Informe de cobertura driver BG96.	29
4.4. Envío de datos por el cliente MQTT de mosquitto.	29
4.5. Recepción de datos en el broker MQTT.	30
4.6. Ubicación del nodo sensor implementado.	30
4.7. Tabla de alarmas activas.	31
4.8. Persistencia de datos.	31
4.9. Trama de escritura al sensor AHT10.	32
4.10. Trama de lectura del sensor AHT10.	32

4.11. Bytes de lectura.	32
4.12. Envío y recepción de comandos por puerto UART.	33
4.13. Instalación del prototipo.	33
4.14. Panel de visualización con las lecturas obtenidas en el caso de uso 1.	34
4.15. Panel de visualización con las lecturas obtenidas en el caso de uso 2.	35
4.16. Comandos para enviar datos al broker MQTT.	35
4.17. Comandos para enviar un SMS.	36
4.18. Recepción del SMS con el mensaje de alarma.	36

Índice de tablas

Capítulo 1

Introducción general

En este capítulo se hace una breve introducción a la necesidad que condujo al desarrollo del trabajo. Se presenta el concepto de internet de las cosas o IoT (del inglés *Internet of Things*) y el estado del arte de dispositivos similares. Asimismo, se explica el objetivo y los alcances del trabajo.

1.1. Internet de las cosas en la agricultura

En los últimos años la agricultura ha enfrentado muchos desafíos, desde una creciente población mundial a ser alimentada, hasta requisitos de sostenibilidad y restricciones ambientales debido al cambio climático y el calentamiento global.

La agricultura es uno de los sectores que más sufre la escasez de agua que existe actualmente en el mundo, uno de los objetivos de implementar la tecnología IoT en este sector, es el de lograr una gestión eficiente y sostenible de los recursos hídricos.

Esto obliga a implementar soluciones que permitan modernizar las prácticas agrícolas. En este contexto, la Agricultura 4.0 representa la última evolución de la agricultura de precisión. La misma se encuentra basada en el concepto de agricultura inteligente, donde convergen el uso de internet de las cosas, computación en la nube, aprendizaje automático para el análisis de grandes volúmenes de datos, vehículos no tripulados y robótica [1].

1.1.1. Internet de las cosas

El concepto de internet de las cosas se refiere a la interconexión digital de dispositivos y objetos a través de una red, es decir, dispositivos como sensores y/o actuadores, equipados con una interfaz de comunicación, unidades de procesamiento y almacenamiento. Estos dispositivos tienen la capacidad de adquirir, intercambiar y transferir datos a la red mediante alguna tecnología de comunicación inalámbrica [2].

El IoT puede usarse a favor de la sostenibilidad, no cabe duda de que Internet es un facilitador de iniciativas sostenibles. De acuerdo con el Foro Económico Mundial la mayoría de los proyectos con internet de las cosas se centran en la eficiencia energética en las ciudades, energías sostenibles y el consumo responsable [3].

Por ejemplo:

- Eficiencia energética: en este sector se interconectan sensores, algoritmos y redes de comunicación para anticipar la demanda eléctrica y así realizar una distribución sostenible de la energía para reducir el precio del kW.

- Uso del agua: esta tecnología pone en funcionamiento máquinas para recoger datos en tiempo real que permitan hacer uso eficiente del agua y reducir su consumo.

1.1.2. Sistemas de monitoreo de cultivos agrícolas

Los sistemas de monitoreo de cultivos agrícolas se encargan de monitorear las distintas variables ambientales a las que están expuestos los cultivos agrícolas, los datos adquiridos ayudan a la toma de decisiones y a manejar de una manera eficiente los recursos con los que cuentan los agricultores.

Cuentan con tres partes fundamentales:

- El nodo sensor, que en sí sería la parte física o hardware, es generalmente de bajo consumo.
- El firmware que abarca la lógica del sistema y se encarga de realizar la adquisición, procesamiento y transferencia de datos que puede o no estar sobre un sistema operativo de tiempo real.
- Nube o plataforma IoT, que ofrecen diferentes servicios como ser almacenamiento, procesamiento, análisis, visualización, etc. Esta parte del sistema permite al usuario del sistema poder visualizar los valores de las variables medidas y así poder tomar decisiones con respecto a las mediciones.

1.2. Estado del arte

Durante la etapa de investigación del trabajo se realizó la búsqueda de productos comerciales en el mercado local e internacional. Se encontraron algunos productos de similares características al que se pretende realizar, un dato interesante a resaltar es que todos los productos encontrados son del mercado internacional, no se encontró ningún producto o empresa que ofrezca este tipo de soluciones en el mercado local.

A continuación se describen los productos encontrados, estas opciones varían con respecto a la tecnología que utilizan.

1.2.1. Libelium

Smart Agriculture PRO figura 1.1 es un módulo de IoT que está diseñado para realizar monitoreo de viñedos para mejorar la calidad del vino, riego selectivo en campos de golf y control de condiciones en invernaderos, entre otros. Permite monitorear múltiples parámetros ambientales que involucran una amplia gama de aplicaciones, desde el análisis del desarrollo en crecimiento hasta la observación del clima. Para ello se ha dotado de sensores de temperatura y humedad del aire y del suelo, luminosidad, radiación solar, velocidad y dirección del viento, precipitaciones, presión atmosférica, humedad de las hojas, distancia y diámetro del fruto o tronco [4].



FIGURA 1.1. Módulo Smart Agriculture PRO.

1.2.2. Nodo RF-M1 DropControl

El nodo RF-M1 es adecuado para tareas de monitoreo simples como parte de una red DropControl o por sí solo. Posee una combinación de entradas que le permite realizar múltiples tareas de monitoreo y almacenarlas en la nube. En la figura 1.2 se muestra el módulo físicamente [5]. Características del dispositivo:

- Redes RF *mesh* o comunicación celular.
- Energía autónoma, solar + batería.
- Actualización del firmware vía aérea, configuraciones y soporte por internet.
- Protección externa IP65.
- Amplia variedad de compatibilidad con sensores.
- Unidad de bajo costo para resolver necesidades básicas de monitoreo.



FIGURA 1.2. Módulo RF-M1 de DropControl.

1.3. Objetivo y alcances

1.3.1. Objetivo

El objetivo principal del trabajo es el diseño e implementación de un prototipo funcional de un sistema de monitoreo de cultivos agrícolas.

1.3.2. Alcances

- Implementación de un prototipo funcional con hardware de bajo consumo.
- Desarrollo del firmware sobre un sistema operativo de tiempo real.
- Transmisión de la información por red celular.
- Visualización de los datos en ThingsBoard [6].

Capítulo 2

Introducción específica

En el presente capítulo se describen los componentes de hardware, software, protocolos de comunicación y plataformas IoT utilizados para realizar el trabajo.

2.1. Componentes principales de hardware

2.1.1. Plataforma de desarrollo STM32 NUCLEO-L432KC

La placa STM32 Nucleo-L432KC que se muestra en la figura 2.1 proporciona una forma asequible y flexible para que los usuarios prueben nuevos conceptos y construyan prototipos eligiendo entre las diversas combinaciones de funciones de rendimiento y consumo de energía que proporciona el microcontrolador STM32L4KC [7].

Características:

- Microcontrolador STM32L4KC en paquete 32 de pines.
- Led de usuario.
- Pulsador de reset.
- Conector de expansión Arduino Nano V3.
- Conector USB Micro-AB para ST-LINK.
- Opciones flexibles de fuente de alimentación.
- Depurador/programador ST-LINK integrado.
- Compatibilidad con una amplia variedad de entornos de desarrollo integrado.
- Oscilador de cristal de 24 MHz.
- Compatible con Arm Mbed Enabled.

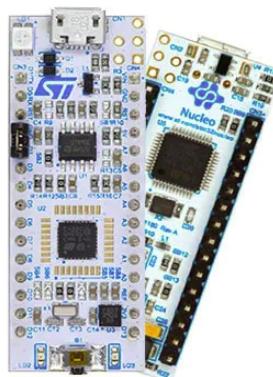


FIGURA 2.1. Plataforma de desarrollo NUCLEO-L432KC¹.

2.1.2. Módulo de comunicación LTE IOT 2 CLICK

LTE IoT 2 click que se muestra en la [2.2](#) está equipado con el módulo BG96 LTE de Quectel Wireless Solutions, que admite tecnologías LTE CAT M1 y NB1, desarrolladas para aplicaciones IoT. Además, admite EGPRS a 850/900/1800/1900 MHz, lo que significa que se puede usar globalmente; no está restringido a ninguna región [8].

Características:

- Protocolos de internet integrados (TCP/UDP/PPP).
- Conectores SMA integrados.
- Leds de alimentación e indicación de estado.
- Conector USB para conectarlo con la aplicación de software de Quectel.
- Interfaz UART.
- Tensión de alimentación 5 V o 3,3 V.



FIGURA 2.2. Módulo LTE IOT 2 CLICK².

¹Datasheet https://www.st.com/resource/en/user_manual/um1956-stm32-nucleo32-boards-mb1180-stmicroelectronics.pdf

²Imagen tomada de la página <https://www.mikroe.com/lte-iot-2-click>

2.1.3. Sensor AHT10

El sensor AHT10 presentado en la figura 2.3 permite obtener lecturas de temperatura y humedad, es de bajo costo y excelente rendimiento. Se utiliza este sensor en aplicaciones de control automático de temperatura, aire acondicionado, estaciones meteorológicas, aplicaciones en el hogar, regulador de humedad y temperatura [9].



FIGURA 2.3. Sensor AHT10³.

2.1.4. Sensor ML8511

El módulo ML8511 presentado en la figura 2.4 es un sensor de luz ultravioleta (UV), entrega una señal de tensión analógica que depende de la cantidad de luz UV que detecta. Sensor ideal para proyectos de monitoreo de condiciones ambientales como el índice UV, aplicaciones meteorológicas, cuidado de la piel, medición industrial de nivel UV. El sensor ML8511 detecta luz con una longitud de onda entre 280-390 nm, este rango cubre tanto al espectro UV-B como al UV-A. La salida analógica está relacionada linealmente con la intensidad UV (mW/cm^2) [10].



FIGURA 2.4. Módulo Sensor ML8511.⁴

2.1.5. Sensor de humedad de suelo HL-69 (Resistivo)

El módulo HL-69 presentado en la figura 2.5, un sensor de humedad de suelo que utiliza la conductividad entre dos terminales para determinar parámetros relacionados al agua, líquidos y humedad [11].

³Imagen tomada de la página <https://esphome.io/components/sensor/aht10.html>

⁴Manual del sensor <https://learn.sparkfun.com/tutorials/ml8511-uv-sensor-hookup-guide/all>

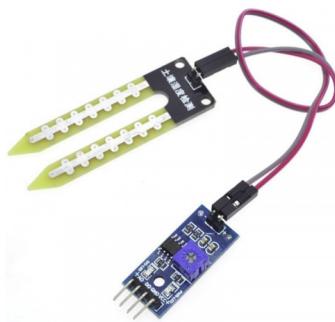


FIGURA 2.5. Módulo sensor HL-69⁵.

2.2. Herramientas de software y testing utilizados

2.2.1. STM32 CubeIDE

STM32CubeIDE es una herramienta de desarrollo multi-OS todo en uno, que forma parte del ecosistema de software STM32Cube. Se trata de una plataforma de desarrollo C/C++ que permite compilar y depurar código para los microcontroladores STM32. Se basa en el marco Eclipse y la cadena de herramientas GCC para el desarrollo y GDB para la depuración. Permite la integración de los cientos de plugins existentes que completan las funcionalidades del IDE de Eclipse [12].

Integra las funcionalidades de configuración y creación de proyectos de STM32 de STM32CubeMX para ofrecer una experiencia de herramienta todo en uno y ahorrar tiempo de instalación y desarrollo [12].

2.2.2. FreeRTOS

FreeRTOS es un sistema operativo en tiempo real (RTOS) líder en el mercado para microcontroladores y pequeños microprocesadores. Distribuido libremente bajo la licencia de código abierto del MIT, FreeRTOS incluye un núcleo y un conjunto creciente de bibliotecas adecuadas para su uso en todos los sectores de la industria. FreeRTOS está diseñado con énfasis en la confiabilidad, la accesibilidad y la facilidad de uso [13]. El logo de FreeRTOS se muestra en la figura 2.6.



FIGURA 2.6. Logo FreeRTOS⁶.

⁵Tienda <https://tienda.sawers.com.bo/hl-69-modulo-sensor-humedad-suelo>

⁶Imagen tomada de la página <https://www.freertos.org/>

2.2.3. CEDDLING

Ceedling es un sistema de compilación diseñado para proyectos en lenguaje C, que podría describirse como una extensión del sistema de compilación Rake (similar a make) de Ruby. Ceedling está dirigido principalmente al desarrollo basado en pruebas en C y está diseñado para reunir CMock, Unity y CException [14].

2.3. Protocolos de Comunicación

2.3.1. UART

UART (*Universal Asynchronous Receiver / transmitter*, por sus siglas en inglés) define un protocolo o un conjunto de normas para el intercambio de datos en serie entre dos dispositivos. UART es sumamente simple y utiliza solo dos hilos entre el transmisor y el receptor para transmitir y recibir en ambas direcciones. Ambos extremos tienen una conexión a masa. La comunicación en UART puede ser simplex (los datos se envían en una sola dirección), semidúplex (cada extremo se comunica, pero solo uno al mismo tiempo), o dúplex completo (ambos extremos pueden transmitir simultáneamente). Los datos se transmiten en forma de tramas [15].

2.3.2. I2C

El protocolo I2C (*Inter-Integrated Circuit*) es un protocolo diseñado para permitir la comunicación entre múltiples circuitos integrados digitales y uno o más chips controladores. Similar a la interfaz periférica en serie, está destinado a comunicaciones de corta distancia dentro de un solo dispositivo. Al igual que las interfaces seriales asíncronas, solo requiere dos cables de señal para el intercambio de información [16].

2.3.3. MQTT

MQTT es un protocolo de mensajería estándar de OASIS para IoT. Está diseñado como un transporte de mensajería de publicación/suscripción extremadamente liviano que es ideal para conectar dispositivos remotos con un espacio de código pequeño y un ancho de banda de red mínimo. MQTT hoy en día se utiliza en una amplia variedad de industrias, como la automotriz, la manufactura, las telecomunicaciones, el petróleo y el gas, etc [17]. En la figura 2.7 se muestra la arquitectura del protocolo MQTT.

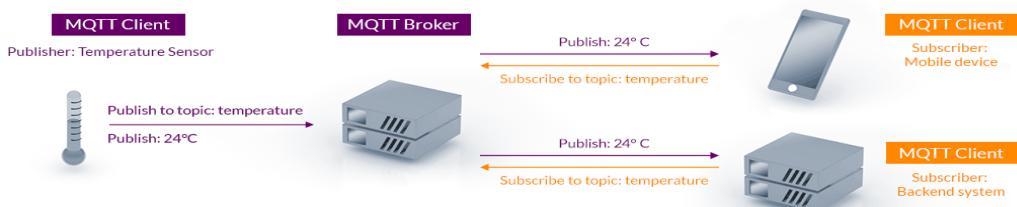


FIGURA 2.7. Arquitectura de publicación/suscripción de MQTT⁷.

⁷Imagen tomada de la página <https://mqtt.org/>

2.4. Plataformas IoT

2.4.1. Ubidots

Ubidots permite enviar datos de sensores a la nube, configurar tableros y alertas, conectarse con otras plataformas, usar herramientas de analítica y arrojar mapas de datos en tiempo real [18]. En la figura 2.8 se muestra un ejemplo de interfaz gráfica en Ubidots.

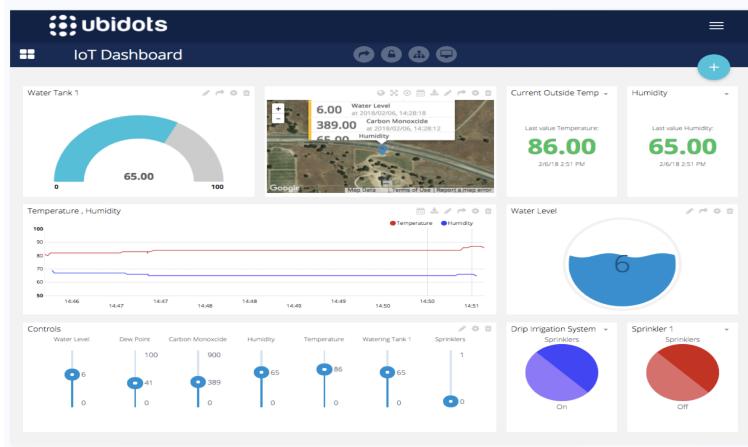


FIGURA 2.8. Ejemplo interfaz gráfica Ubidots⁸.

2.4.2. ThingsBoard

ThingsBoard es una plataforma IoT de código abierto para la recopilación, el procesamiento, la visualización y la gestión de dispositivos. Permite la conectividad de dispositivos a través de protocolos estándares de la industria IoT: MQTT, CoAP y HTTP. ThingsBoard combina escalabilidad, tolerancia a fallas y rendimiento [6]. En la figura 2.9 se muestra un ejemplo de una interfaz gráfica desarrollada en ThingsBoard.

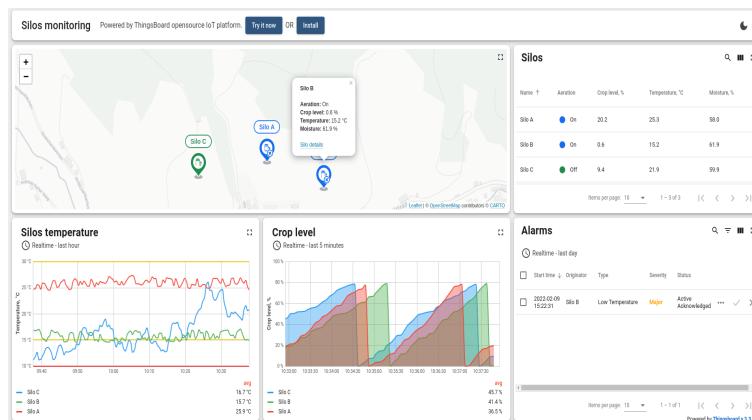


FIGURA 2.9. Ejemplo interfaz gráfica ThingsBoard⁹.

⁸Imagen tomada de la página <https://ubidots.com/platform/time-series>

⁹Imagen tomada de la página <https://thingsboard.io/smart-farming/>

Capítulo 3

Diseño e implementación

En este capítulo se abordará la descripción de la arquitectura general del sistema, arquitectura del firmware, código de los controladores desarrollados, desarrollo del hardware y la configuración de la plataforma IoT.

3.1. Diagrama de bloques general del sistema

En la figura 3.1 se muestra el diagrama en bloques general del sistema donde se describe la arquitectura IoT aplicada al trabajo que consta de tres capas: percepción, red y aplicación.

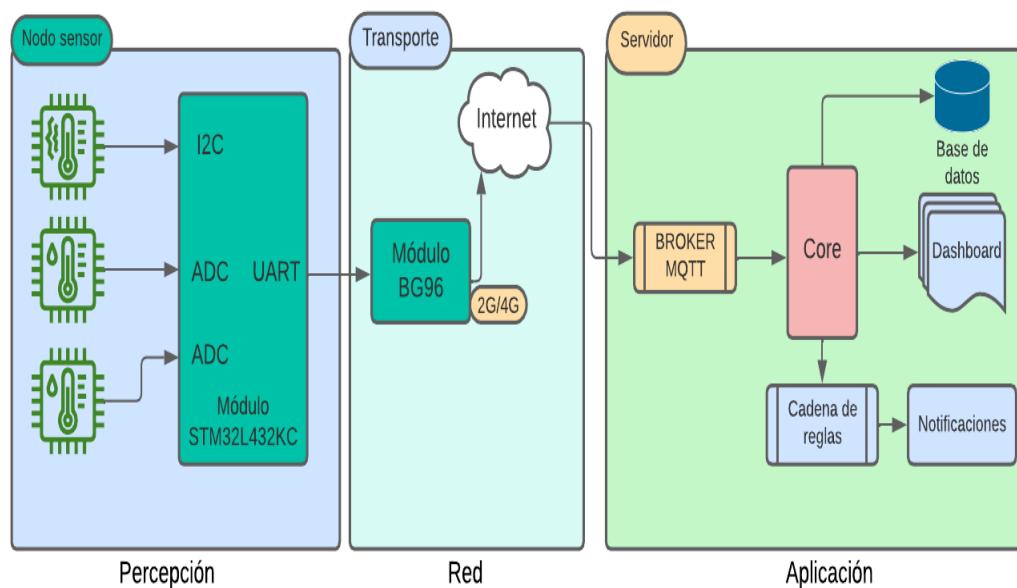


FIGURA 3.1. Diagrama general del sistema IoT.

En cada una de las capas se despliegan tecnologías y componentes de hardware y software. A continuación se describe cada una de las capas.

- Capa de percepción: en la capa de percepción se encuentra el nodo sensor que es el encargado de medir las variables físicas, hacer un preprocesamiento y posteriormente enviar los datos a la capa de red. Para su desarrollo se utilizó la tarjeta de prueba STM32L432KC que contiene el firmware del sistema, además, se utilizaron los siguientes sensores: sensor de humedad y temperatura ambiente AHT10, sensor de humedad de suelo HL-69 y el sensor de luz UV ML8511.

- Capa de red: en lo que respecta a la conectividad de red, se empleó un módulo Quectel BG96 que es capaz de establecer conexión de manera automática con las redes 2G, 4G y NB-IoT, según las condiciones de red en el lugar de implementación del nodo sensor. Este módulo se comunica con el microcontrolador mediante comandos AT a través del puerto UART.
- Capa de aplicación: en la capa de aplicación, se utilizó ThingsBoard como plataforma IoT que brinda los microservicios de broker MQTT como puerta de entrada al servidor, base de datos para el almacenamiento, interfaz gráfica para la visualización de los datos y permite gestionar las alarmas del sistema.

3.2. Arquitectura de firmware

El desarrollo del firmware fue la tarea más compleja del trabajo debido a que uno de los objetivos fue lograr un firmware estructurado en capas para facilitar el desarrollo y reducir la complejidad del código. La figura 3.2 muestra la división en capas del firmware desarrollado.

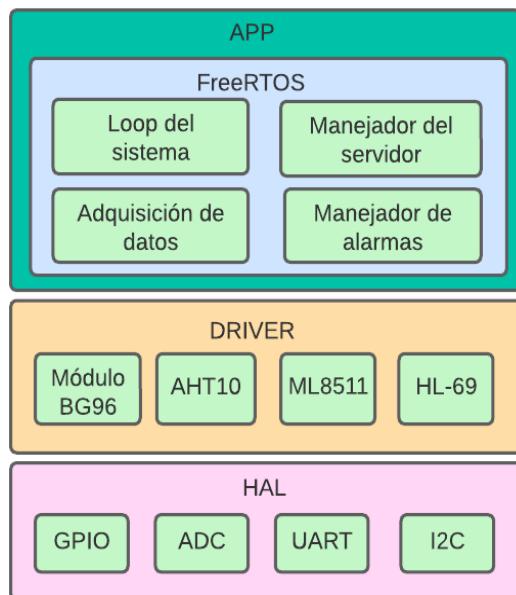


FIGURA 3.2. Capas del firmware.

3.2.1. Capa HAL

La capa HAL (*Hardware Abstraction Layer*) proporcionada por el fabricante del microcontrolador es la más baja del sistema, proporciona a las capas superiores la capacidad de interactuar con los periféricos del microcontrolador a través de funciones en lenguaje C.

- GPIO: la API proporciona funciones para gestionar las entradas y salidas del microcontrolador. Fueron utilizadas por la capa de APP para el control de los leds de debug y para el encendido y reset del módulo de comunicación.

- ADC: proporcionan funciones para la configuración, lectura y escritura de los pines del microcontrolador para trabajar con señales analógicas. Se utilizaron estas funciones para hacer la lectura de los sensores de humedad del suelo y el sensor de luz UV.
- UART: brinda funciones para la lectura y escritura del puerto UART del microcontrolador. El firmware utiliza estas funciones para la comunicación con el módulo BG96.
- I2C: proporciona funciones para la lectura y escritura por protocolo I2C. El driver del sensor AHT10 utiliza estas funciones para hacer la lectura de los datos.

3.2.2. Capa drivers

La capa drivers (manejador de dispositivos) está compuesta por los manejadores que se desarrollaron para interactuar con el hardware externo al microcontrolador. Se desarrollaron dos drivers que se describen a continuación:

- Driver BG96: las funciones más importantes que proporciona el driver son:
 - Estado del módulo.
 - Descripción del módulo.
 - Configuración APN de la red.
 - Conexión TCP.
 - Conexión al broker MQTT.
- Driver AHT10: se desarrolló utilizando la hoja de datos del sensor, proporciona funciones de inicialización y lectura de humedad y temperatura obtenidos por el sensor.

Para los sensores ML8511 y HL-69 que son sensores analógicos no se desarrollaron drivers, sino que se crearon funciones para convertir el valor analógico entregado a un valor significativo para el usuario con respecto a la variable física medida.

3.2.3. Capa aplicación

La capa de APP o aplicación es la de mayor nivel jerárquico. Se desarrolló sobre freeRTOS que permite hacer un código más escalable.

Se implementaron cuatro tareas, que se describen a continuación:

- Loop del sistema: esta tarea es la que brinda la secuencialidad del sistema.
- Manejador del servidor: se encarga de manejar la conexión a la red y al broker MQTT.
- Adquisición de datos: se encarga de hacer la lectura de los sensores.
- Manejador de alarmas: esta tarea se encarga de hacer el control de las alarmas del sistema.

3.3. Desarrollo del firmware

Para el desarrollo del firmware se utilizó STM32CubeIDE que es el entorno de desarrollo oficial de STMicroelectronic.

El firmware fue desarrollado sobre freeRTOS, se utilizaron algunas de sus funcionalidades como colas, semáforos, tareas e interrupciones.

En la figura 3.3 se muestra en diagrama de flujo de inicialización del firmware.



FIGURA 3.3. Diagrama de flujo de inicialización del firmware.

El firmware comienza realizando la siguiente secuencia de acciones: configuración del hardware del microcontrolador, inicialización de los drivers del hardware externo, creación de los recursos del sistema operativo y finalmente inicialización del scheduler.

Para el control del sistema se crearon cuatro tareas sobre freeRTOS, que se comunican y sincronizan a través de colas y semáforos.

3.3.1. Tarea loop del sistema

La tarea loop comienza iniciando un timer que se encarga controlar el tiempo de repetición del ciclo de la tarea. Luego la tarea se bloquea. Cuando se termina el tiempo del timer, se ejecuta el handler de la interrupción desbloqueando la tarea loop. La tarea envía un evento a la cola de adquisición de datos para realizar la lectura de los sensores y un evento a la cola que maneja la conexión para levantar el servidor. Posteriormente la tarea comprueba si se logró levantar una conexión. Si la conexión existe la tarea manda un evento por la cola del servidor para que se envíen los datos al broker MQTT. Luego la tarea envía un evento a la cola de alarmas para mandar los SMS (Short Message Service) de las alarmas activas del sistema. Finalmente, después de monitorear las alarmas la tarea manda un evento

a la cola de servidor para la desconexión. Al finalizar el ciclo de la tarea, inicia el timer nuevamente y manda al microcontrolador a modo de bajo consumo para ahorrar energía.

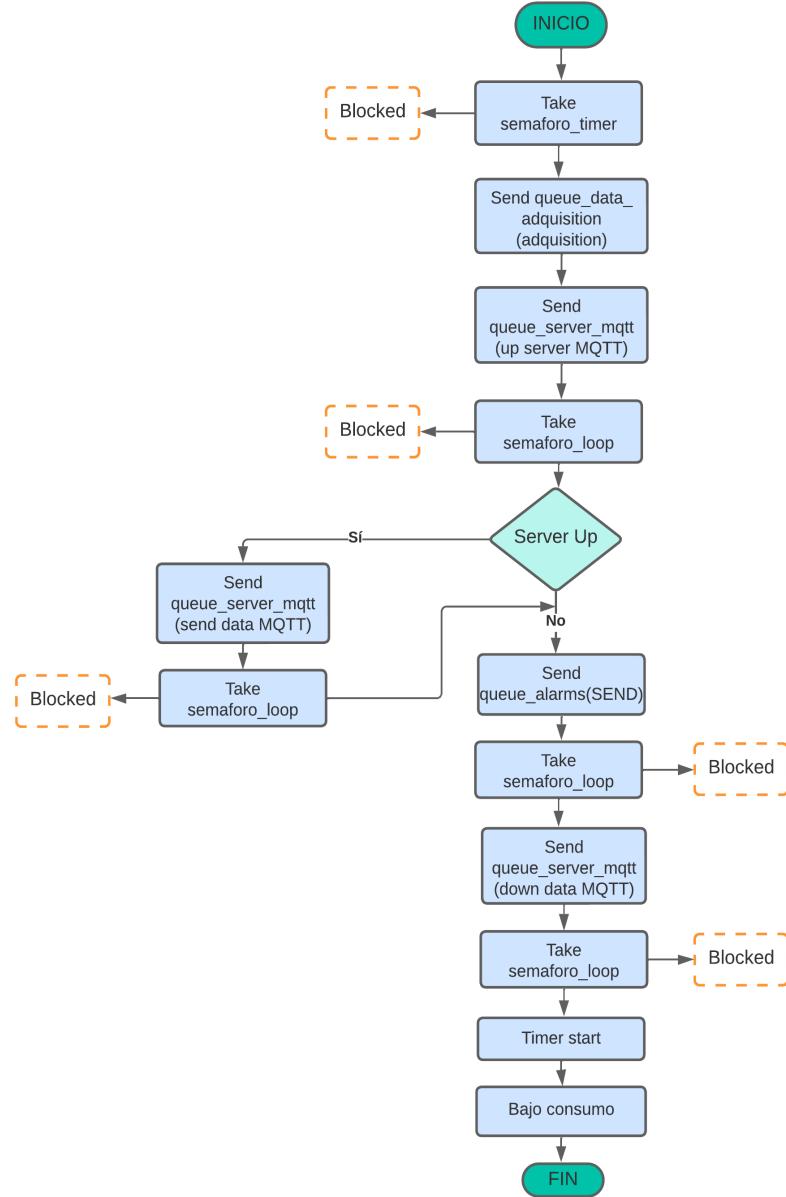


FIGURA 3.4. Diagrama de flujo de la tarea loop.

3.3.2. Tarea adquisición de datos

La figura 3.5 muestra el diagrama de flujo de la tarea de adquisición de datos. La tarea inicia revisando si hay datos en la cola de adquisición. Si existen datos se realiza la lectura de todos los sensores, para posteriormente enviar los valores leídos por la cola de datos y alarmas. Si no hay datos en la cola la tarea se bloquea.

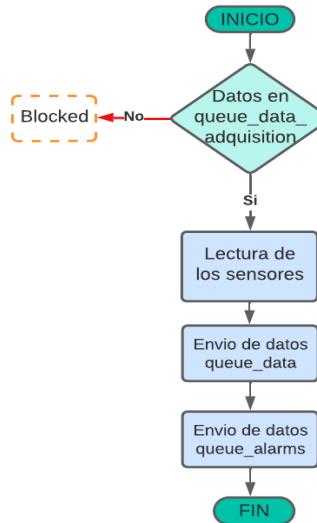


FIGURA 3.5. Diagrama de flujo tarea de adquisición de datos.

3.3.3. Tarea manejador de alarmas

El control de las alarmas se realiza a través de una tarea del sistema operativo, la figura 3.6 muestra el diagrama de flujo de la tarea que maneja las alarmas.

Al ingresar al bucle infinito, la tarea comienza por inspeccionar la cola de alarmas. Si se encuentran datos en dicha cola, procede a analizar el evento que contiene la información recibida.

Existen dos tipos posibles de eventos: "monitorear" y "enviar". En caso de que el evento sea "monitorear", se verifica si el valor del sensor de humedad es menor a 10. En caso afirmativo, se incrementa en uno la variable que lleva el registro de las alarmas activas. Si, por otro lado, el evento es "enviar", se verifica si existen alarmas activas y, de ser así, se envía un mensaje de texto.

En caso de no haber datos en la cola de alarmas, la tarea entra en un estado de bloqueo.

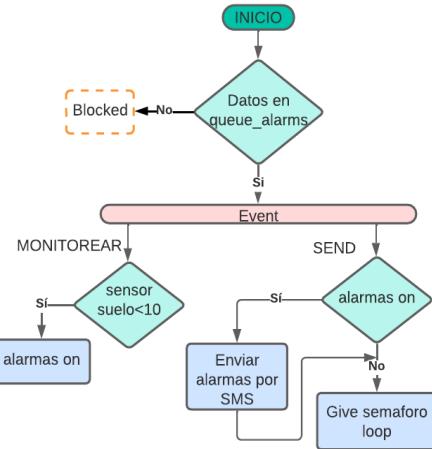


FIGURA 3.6. Diagrama de flujo de la tarea manejador de alarmas.

3.3.4. Tarea manejador del servidor

La tarea comienza esperando datos en la cola del servidor, al llegar datos se analiza el evento que se recibe. Se tiene tres posibles eventos: UP, DOWN y SEND. Si el evento es UP, la tarea ingresa a la máquina de estados que se muestra en la figura 3.9, la que se encarga de establecer una conexión con el servidor. Si el evento es DOWN, la tarea ejecuta la máquina de estados que se ve en la figura 3.8, que se encarga de terminar la conexión con el servidor. Si el evento es SEND, la tarea obtiene los últimos datos leídos por los sensores, arma la trama y publica los datos al broker MQTT.

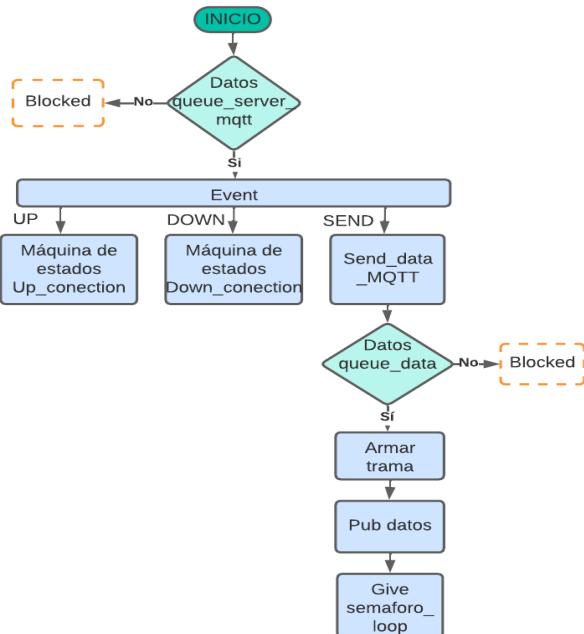


FIGURA 3.7. Diagrama de flujo de la tarea conexión server MQTT.

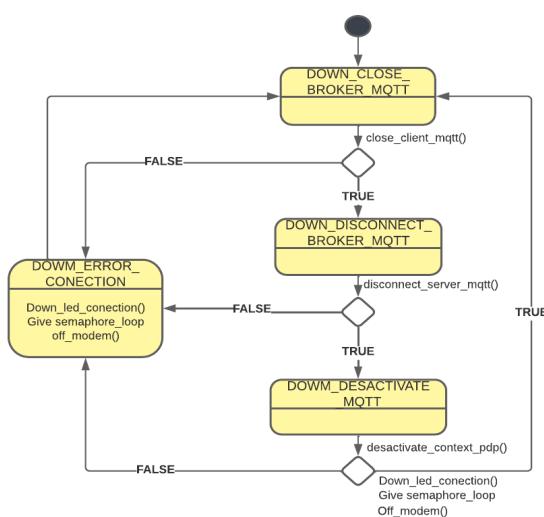


FIGURA 3.8. Máquina de estados down servidor.

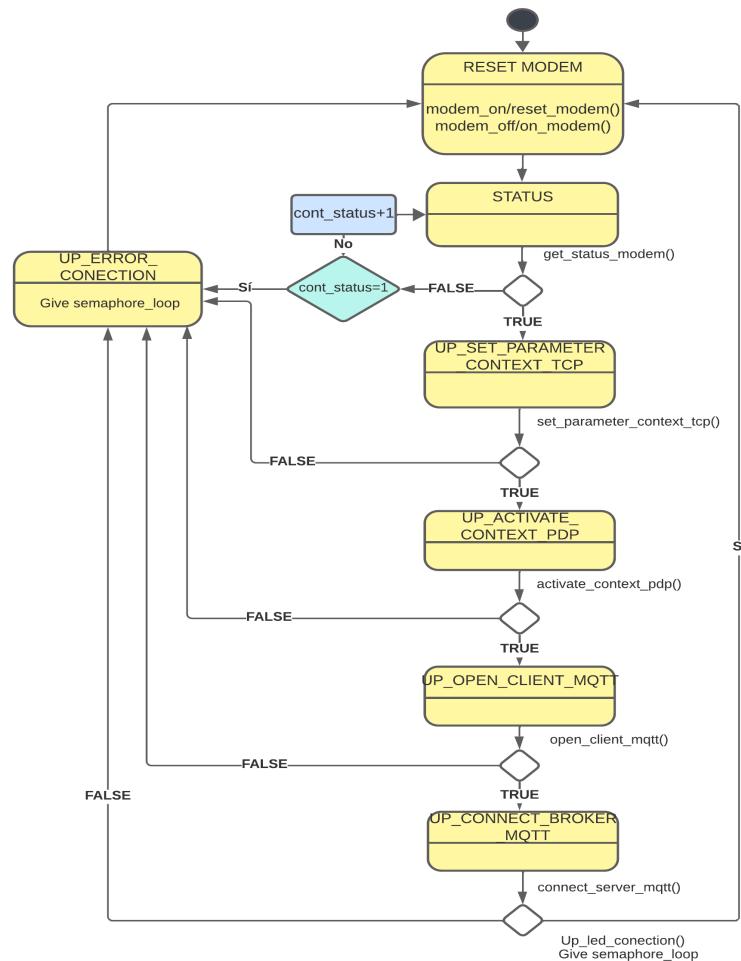


FIGURA 3.9. Máquina de estados up servidor.

3.4. Controladores implementados

Se implementaron dos controladores: para el módulo de comunicación BG96 y para el sensor de humedad y temperatura ambiente AHT10.

3.4.1. Controlador para el sensor AHT10

Para la lectura de humedad y temperatura, se desarrolló en el driver las funciones de *aht10_get_humidity()* y *aht10_get_temperature()* que se muestran en el código 3.1. Las funciones comienzan iniciando una medición en el sensor con la función *aht10_launch_measurement()*, luego realizan la lectura de los registros del sensor, almacenando los valores de retorno en un buffer. Con los bytes obtenidos se realiza un corrimiento para obtener los bytes que contienen la información de humedad y temperatura. Finalmente, se aplican las fórmulas de las líneas 1 y 2 del código 3.1 para convertir los bytes en valores significativos de las variables físicas.

```

1 #define TEMPERATURE(A) ((int8_t) ((A *0.000191)-50)
2 #define HUMIDITY(A) ((uint8_t) (A *0.000095))
3 /**
4  * @brief funcion para obtener el valor de la humedad
5  *
6  * @param obj puntero a la estructura principal del driver

```

```

7  * @param data puntero a la variable donde se retorna la humedad
8  * @return aht10_status_fnc
9  */
10 aht10_status_fnc aht10_get_humidity(aht10_config_t*obj, uint8_t *data)
11 {
12     obj->status_fun=AHT10_ERROR;
13     uint8_t bufferRead[6]={0};
14     uint32_t data_humidity=0;
15     obj->status_fun=aht10_launch_measurement(obj);
16     if (obj->status_fun==AHT10_OK)
17     {
18         obj->status_fun= obj->readI2C(AHT10_ADDRESS,bufferRead,6);
19         if (obj->status_fun==AHT10_OK)
20         {
21             data_humidity=((uint32_t)bufferRead[1]<<16) | ((uint16_t)
22             bufferRead[2]<<8) | (bufferRead[3])>>4;
23             *data= HUMIDITY(data_humidity);
24         }
25     }
26     return obj->status_fun;
27 }
28 /**
29 * @brief funcion para obtener el valor de la temperatura
30 *
31 * @param obj puntero a la estructura principal del driver
32 * @param data puntero a la variable donde se retorna la temperatura
33 * @return aht10_status_fnc
34 */
35 aht10_status_fnc aht10_get_temperature(aht10_config_t*obj, int8_t *data)
36 {
37     uint8_t buffer_read[6]={0};
38     uint32_t data_temperature=0;
39     obj->status_fun=AHT10_ERROR;
40     obj->status_fun=aht10_launch_measurement(obj);
41     if (obj->status_fun==AHT10_OK)
42     {
43         obj->status_fun=obj->readI2C(AHT10_ADDRESS ,buffer_read,6);
44         if (obj->status_fun==AHT10_OK)
45         {
46             data_temperature=((uint32_t)(buffer_read[3] & 0x0F)<<16) | ((
47             uint16_t) buffer_read[4]<<8) | buffer_read[5];
48             *data= TEMPERATURE(data_temperature);
49         }
50     }
51     return obj->status_fun;
52 }
```

CÓDIGO 3.1. Funciones de lectura de humedad y temperatura.

3.4.2. Controlador para el módulo de comunicación BG96

Se han creado diversas funciones destinadas al controlador BG96, las que posibilitan la configuración del módulo, la configuración de la red, y las más cruciales para el funcionamiento: las relacionadas con la conexión, publicación y desconexión del servidor MQTT.

El código 3.2 presenta dos funciones implementadas en el controlador BG96. La función *connect_server_mqtt()* se emplea para establecer la conexión con el servidor MQTT. En su interior, esta función envía el comando visible en la línea 6 y aguarda la respuesta del módulo, que puede ser *OK* o *ERROR*.

La función *publish_message()* es una de las más utilizadas por el firmware, es la encargada de publicar la trama JSON que contiene los datos de los sensores al tópico configurado inicialmente.

```

1 /**
2  * @brief funcion para conectarse al servidor MQTT
3 *
4  * @param self puntero a la estructura principal del driver
5  * @return em_bg96_error_handling
6 */
7 em_bg96_error_handling connect_server_mqtt(st_bg96_config *self)
8 {
9     self->ft_resp=RF_ERROR;
10    char cmd[150]={0};
11    sprintf(cmd, "AT+QMTCNN=%u,\"%s\", \"%s\", \"%s\"\r", self->self_mqtt.
12        identifier_socket_mqtt, self->self_mqtt.mqtt_client_id, self->
13        self_mqtt.username, self->self_mqtt.mqtt_password);
14    self->ft_resp=self->send_data_device(cmd,RS_BG96_CERO, self->
15        buffer_resp,10000);
16    if (self->ft_resp!=RF_OK)
17    {
18        self->last_error=BG96_ERROR_CONNECT_SERVER_MQTT;
19    }
20    return self->ft_resp;
21 }
22 /**
23  * @brief funcion para publicar un mensaje al topico configurado
24 *
25  * @param self puntero a la estructura principal del driver
26  * @param topic topico donde se mandaran los datos
27  * @param data datos que se publicaran en el topico
28  * @return em_bg96_error_handling
29 */
30 em_bg96_error_handling publish_message(st_bg96_config *self ,char *topic ,
31                                         char *data)
32 {
33     self->ft_resp=RF_ERROR;
34     char cmd[50]={0};
35     char buffer_data[220]={0};
36     sprintf(buffer_data, "%\x1a\r",data);
37     sprintf(cmd, "AT+QMTPUB=%u,0,0,\\"%s\"\r", self->self_mqtt.
38         identifier_socket_mqtt,topic);
39     self->ft_resp=self->send_data_device(cmd,RS_SIG, self->buffer_resp
40         ,3000);
41     if (RF_OK==self->ft_resp)
42     {
43         self->ft_resp=self->send_data_device(buffer_data,RS_BG96_CERO, self->
44             buffer_resp,15000);
45         if (self->ft_resp!=RF_OK)
46         {
47             self->last_error=BG96_ERROR_PUBLISH_MESSAGE;
48         }
49     }
50     else self->last_error=BG96_ERROR_PUBLISH_MESSAGE;
51     return self->ft_resp;
52 }
```

CÓDIGO 3.2. Función de conexión y publicación al broker MQTT.

Para el envío de mensajes de texto se desarrolló en el driver la función *send_sms_bg96()* que recibe como parámetros el número al que se desea mandar el mensaje y el

texto del mensaje a enviar. El firmware utiliza esta función para mandar mensajes de alarma al usuario. El código 3.3 muestra la implementación de la función *send_sms_bg96()*

```

1 /**
2 * @brief funcion para enviar mensajes de texto
3 *
4 * @param self puntero a estructura principal del driver
5 * @param number numero de telefono al que se mandara el mensaje
6 * @param message mensaje de texto
7 * @return em_bg96_error_handling
8 */
9 em_bg96_error_handling send_sms(st_bg96_config *self, char*number, char*
10   message)
11 {
12     self->ft_resp=RF_OK;
13     char buffer_message[20];
14     char buffer_number[20];
15     sprintf(buffer_number, "AT+CMGS=\"%s\"\r", number);
16     sprintf(buffer_message, "%s\x1a\r", message);
17     self->ft_resp=self->send_data_device(buffer_number, RS_SIG, self->
18       buffer_resp, 12000);
19     if (RF_OK==self->ft_resp)
20     {
21       self->ft_resp=self->send_data_device(buffer_message, RS_OK, self->
22         buffer_resp, 12000);
23       if (RF_OK!=self->ft_resp)
24     {
25       self->last_error=BG96_ERROR_SEND_SMS;
26     }
27   }
28   return self->ft_resp;
29 }
```

CÓDIGO 3.3. Función para enviar SMS.

3.5. Desarrollo del hardware

Para el diseño del hardware se empleó KiCad 6.0, una herramienta de diseño que se utilizó durante el desarrollo de la especialización.

3.5.1. Esquemático

Dado que se trata de un prototipo, se diseñó una tarjeta que permite la integración y conexión de los componentes utilizados en el trabajo, incluyendo el módulo de comunicación celular, la tarjeta de desarrollo con el microcontrolador y los módulos sensores.

En la figura 3.10 se muestra la página raíz del esquemático del trabajo, está dividida en tres zonas:

- Zona 1: índice de las hojas esquemáticas del trabajo.
- Zona 2: modelo 3D de la tarjeta desarrollada.
- Zona 3: conexiones entre los diferentes hojas esquemáticas .

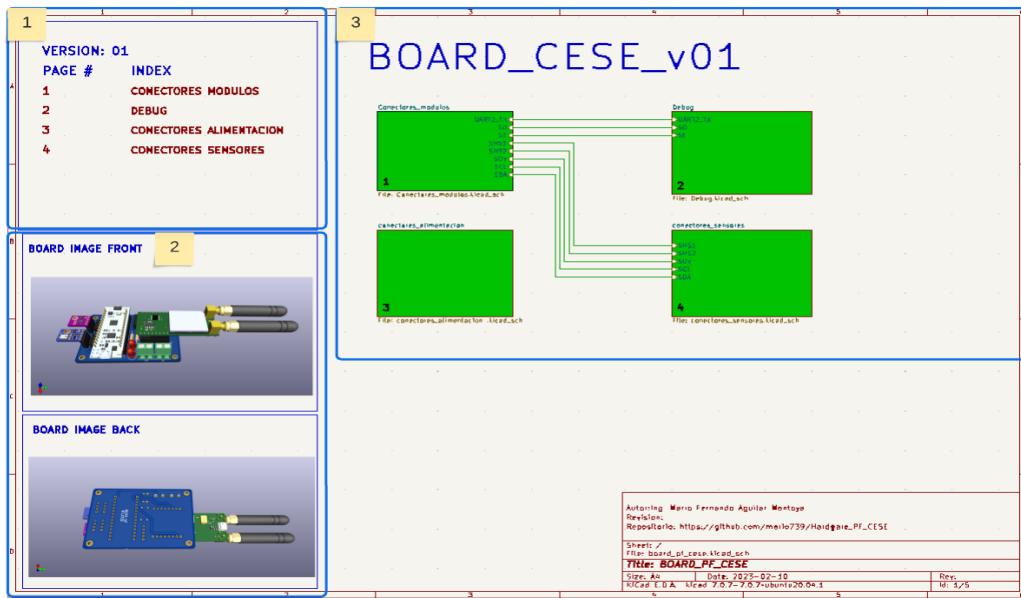


FIGURA 3.10. Esquemático página raíz.

En figura 3.11 se muestran los conectores de los módulos más importantes: el módulo de comunicación BG96, módulo NUCLEO-L432KC y la conexión entre ellos.

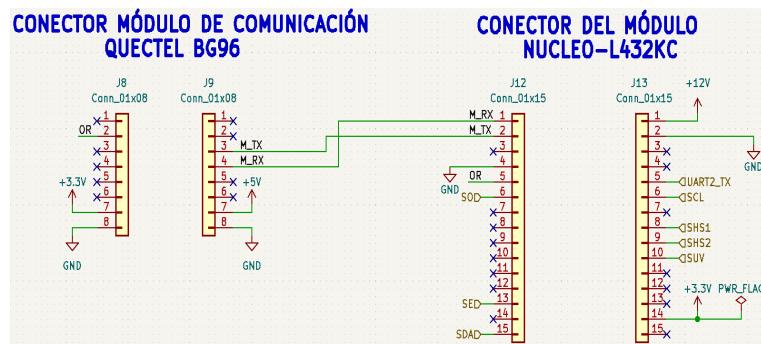


FIGURA 3.11. Conector módulo BG96 y NUCLEO-L432KC.

Se incorporaron dos leds con fines de depuración: uno para indicar si se logró la conexión al servidor MQTT y el otro para señalizar cualquier estado de error del sistema. También se incluyó un conector para un puerto serial a través del cual el módulo transmite las secuencias de comandos enviadas y recibidas del módulo de comunicación. En la 3.12 se puede ver el circuito implementado.

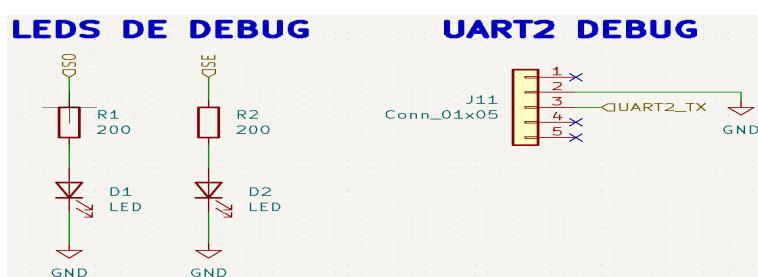


FIGURA 3.12. Esquemático interfaz de debug.

En la figura 3.13 se pueden ver los conectores que se colocaron a la placa para conectar los sensores del nodo: sensor de humedad 1, sensor de humedad 2, sensor de luz UV y el sensor de humedad y temperatura ambiente AHT-10. Se coloca-

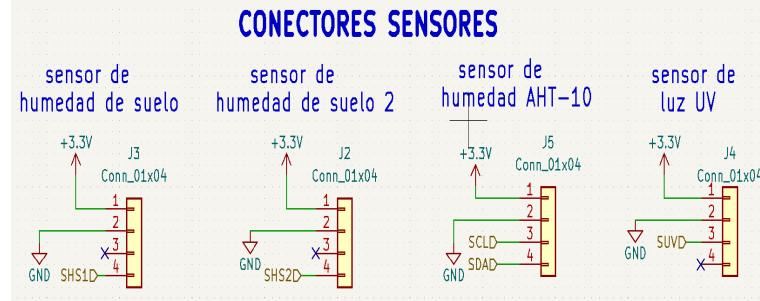


FIGURA 3.13. Esquemático conectores sensores.

ron dos colectores de alimentación como se muestra en la figura 3.14, el conector de 12V para alimentar al módulo del microcontrolador y el otro conector para alimentar al módulo de comunicación.

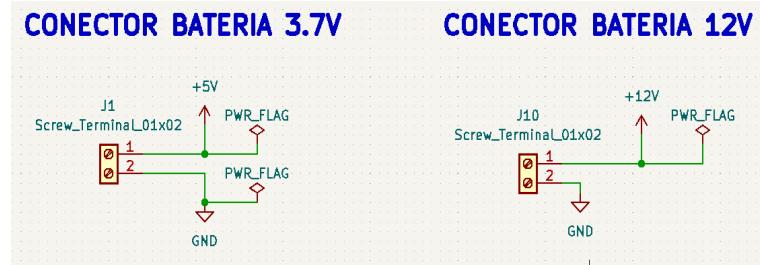


FIGURA 3.14. Esquemático conectores de alimentación.

3.5.2. PCB del hardware

La figura 3.15 muestra el circuito impreso diseñado para el trabajo.

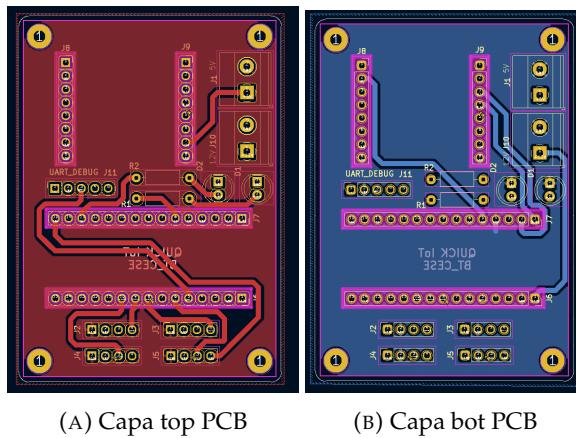


FIGURA 3.15. PCB del trabajo.

En la figura 3.16 se muestra el diseño de la tarjeta del circuito impreso en 3D.

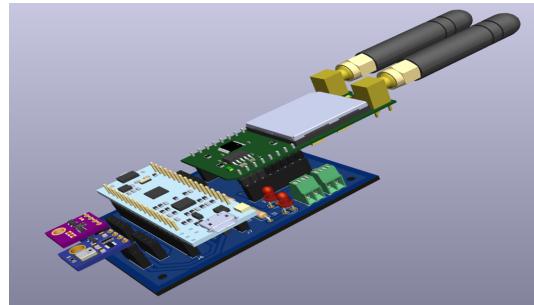


FIGURA 3.16. Modelo 3D de la tarjeta.

3.5.3. Fabricación del circuito impreso

Una vez completado y validado el diseño, se generaron los archivos de fabricación y se mandaron a la empresa JLCPCB para su producción. La figura 3.17 muestra la tarjeta ya ensamblada con los módulos.

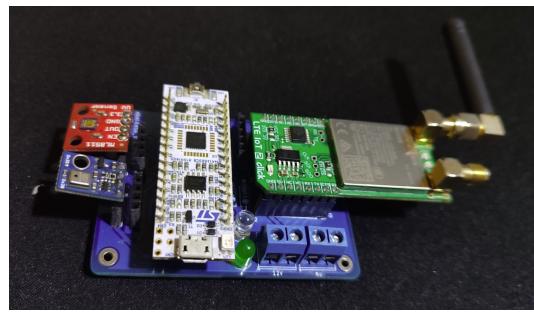


FIGURA 3.17. PCB ensamblado.

3.6. Paneles de visualización

La herramienta de visualización de ThingsBoard es muy versátil para el armado de paneles de visualización escalables y altamente configurables.

Se armó un panel de visualización principal que muestra los nodos sensores monitoreados por el sistema y un panel secundario que muestra las variables monitoreadas por cada nodo sensor a través de gráfica, tablas, etc.

3.6.1. Panel principal

La figura 3.18 muestra el panel principal de la interfaz gráfica. A continuación, se detallarán las distintas áreas que componen este panel. Este se divide en las siguientes zonas:

- Zona 1: listado de todos los nodos sensores implementados y activos, haciendo click en el sensor se navega al panel de visualización secundario.
- Zona 2: gráficas que representan la evolución en el tiempo de los valores de las variables registradas por los sensores.
- Zona 3: mapa con la ubicación de los nodos sensores implementados.

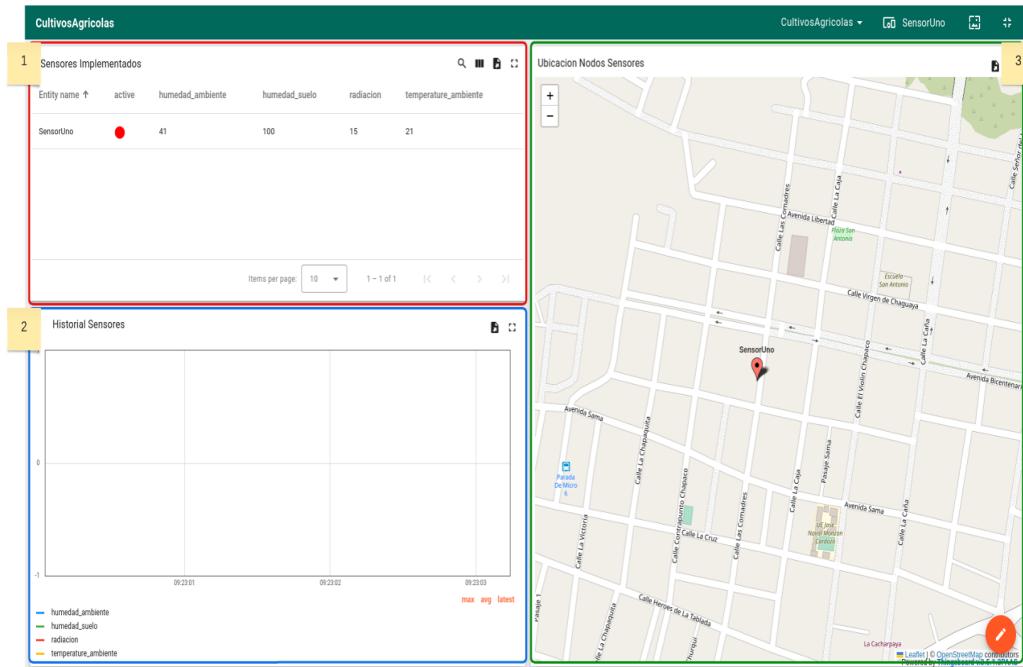


FIGURA 3.18. Panel principal de la interfaz gráfica.

3.6.2. Panel nodo sensor

Para tener un mayor detalle de todos los parámetros de monitoreo de cada nodo sensor se creó un panel secundario que se muestra en la figura 3.19.

El panel está dividido en las siguientes zonas:

- Zona 1: gráficas que muestran los cambios que van teniendo los valores de las variables medidas por los sensores con respecto al tiempo.
- Zona 2: widgets que muestran el último valor obtenido por el nodo sensor de cada variable monitoreada.
- Zona 3: tabla que muestra las alarmas que se activaron.
- Zona 4: se tiene un mapa que muestra la ubicación del nodo sensor implementado.

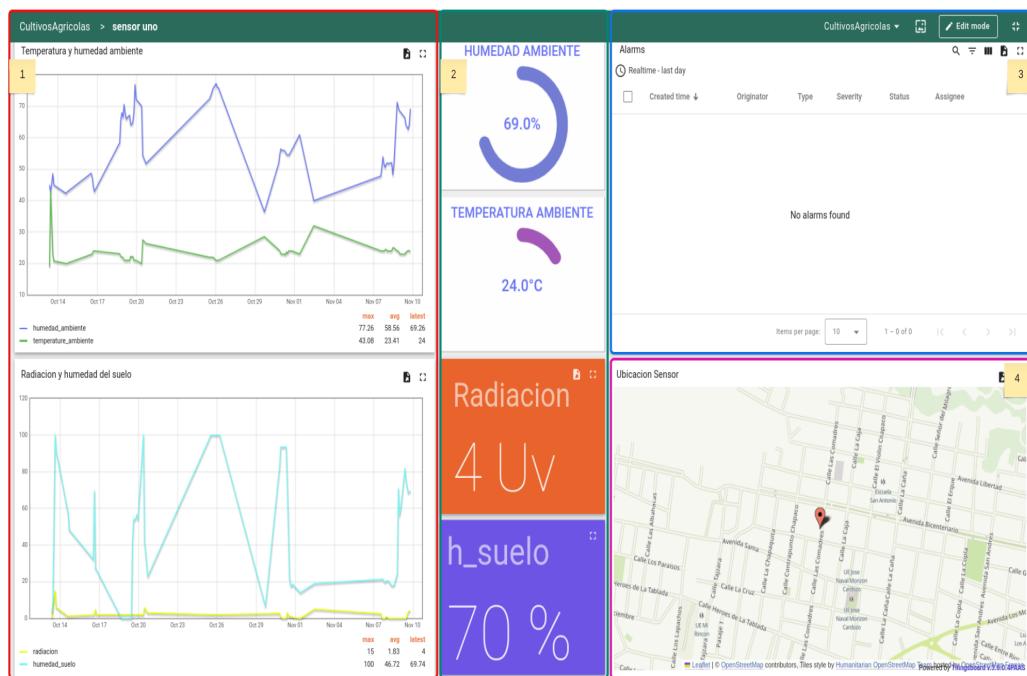


FIGURA 3.19. Panel nodo sensor.

Capítulo 4

Ensayos y resultados

En este capítulo se explican las pruebas realizadas al hardware, firmware, controladores y a la plataforma IoT.

4.1. Pruebas unitarias

Para la implementación de los drivers se utilizó la metodología de desarrollo TDD [19]. Esto implica que se escribieron pruebas unitarias para los drivers. Se utilizó Ceedling como herramienta de desarrollo de pruebas automáticas.

En el fragmento de código 4.1 se muestra la prueba implementada para la función de `aht10_get_status()` del driver del sensor AHT10. Internamente utiliza funciones mock para la lectura y escritura por protocolo I2C.

Se prueban los siguientes casos:

- De la línea 9 a la 12: cuando la lectura del registro de estado del sensor es 0.
- De la línea 14 a la 17: cuando la lectura del registro de estado del sensor es 255.
- De la línea 19 a la 20: cuando la función de lectura devuelve error.

```

1 /**
2 * @test prueba para la funcion de estado del sensor AHT10
3 *
4 */
5 void test_estado_del_sensor(void)
6 {
7     uint8_t buffer[1]={0};
8
9     uint8_t data=0;
10    read_i2c_port_ExpectAndReturn(AHT10_ADDRESS,buffer,1,AHT10_OK);
11    read_i2c_port_ReturnThruPtr_buffer(&data);
12    TEST_ASSERT_EQUAL(SENSOR_IDLE,aht10_get_status(&aht10config));
13
14    data=255;
15    read_i2c_port_ExpectAndReturn(AHT10_ADDRESS,buffer,1,AHT10_OK);
16    read_i2c_port_ReturnThruPtr_buffer(&data);
17    TEST_ASSERT_EQUAL(SENSOR_BUSY,aht10_get_status(&aht10config));
18
19    read_i2c_port_ExpectAndReturn(AHT10_ADDRESS,buffer,1,AHT10_ERROR);
20    TEST_ASSERT_EQUAL(SENSOR_BUSY,aht10_get_status(&aht10config));
21 }
```

CÓDIGO 4.1. Tests del driver del sensor AHT10.

En el fragmento de código 4.2 se muestra la prueba desarrollada para la función `send_sms_bg96()` encargada de mandar los mensajes de texto del driver BG96. Se probaron varios casos, cuando la función de envío de comandos responde con un OK y cuando responde con errores.

```

1 /**
2  * @brief prueba de la funcion de envio de SMS
3 *
4 */
5 void test_send_sms(void)
6 {
7     char bf_resp[30]={0};
8
9     send_data_ExpectAndReturn(
10         "AT+CMGS=\"72950576\"\r",
11         RS_SIG, bf_resp, 12000,
12         RF_OK);
13     send_data_ExpectAndReturn("HOLA\x1a\r", RS_OK, bf_resp, 12000, RF_OK);
14     TEST_ASSERT_EQUAL(RF_OK, send_sms(&module, "72950576", "HOLA"));
15
16     send_data_ExpectAndReturn(
17         "AT+CMGS=\"72950576\"\r",
18         RS_SIG, bf_resp, 12000,
19         RF_OK);
20     send_data_ExpectAndReturn("HOLA\x1a\r", RS_OK, bf_resp, 12000, RF_ERROR);
21     TEST_ASSERT_EQUAL(RF_ERROR, send_sms(&module, "72950576", "HOLA"));
22
23     send_data_ExpectAndReturn(
24         "AT+CMGS=\"72950576\"\r",
25         RS_SIG, bf_resp, 12000,
26         RF_ERROR);
27     TEST_ASSERT_EQUAL(RF_ERROR, send_sms(&module, "72950576", "HOLA"));
28 }
```

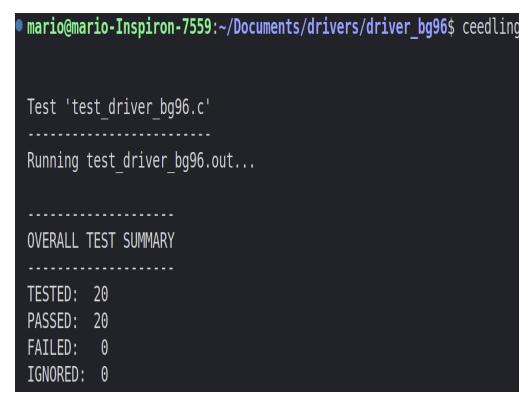
CÓDIGO 4.2. Tests del driver del módulo BG96.

En la figura 4.1 se muestran los resultados de las pruebas realizadas a los drivers. En la figura 4.1a se puede ver que se realizaron 7 pruebas para el driver del sensor AHT10 de los cuales 7 pasaron exitosamente. En la figura 4.1b se puede ver que se realizaron 20 pruebas al driver del módulo BG96 y todas pasaron exitosamente.



```
mario@mario-Inspiron-7559:~/Documents/drivers/aht10$ cedeling
Test 'test_aht10.c'
-----
Running test_aht10.out...
-----
OVERALL TEST SUMMARY
-----
TESTED: 7
PASSED: 7
FAILED: 0
IGNORED: 0
```

(A) Tests realizados al driver AHT10



```
mario@mario-Inspiron-7559:~/Documents/drivers/driver_bg96$ cedeling
Test 'test_driver_bg96.c'
-----
Running test_driver_bg96.out...
-----
OVERALL TEST SUMMARY
-----
TESTED: 20
PASSED: 20
FAILED: 0
IGNORED: 0
```

(B) Tests realizados al driver BG96

FIGURA 4.1. Resultados de los tests realizados a los drivers.

Una forma cuantitativa de evaluar estas pruebas son los informes de cobertura generados por Ceedling.

La figura 4.2 presenta el informe de cobertura del driver AHT10, en el cual se evidencia que las pruebas abarcan el 100 % de las líneas de código implementadas, así como exploran la totalidad de las posibles combinaciones en las estructuras condicionales.

En la figura 4.3, también se puede apreciar el informe de cobertura del driver BG96, el cual abarca un 98,4 % de las líneas de código ejecutadas y explora más del 98,3 % de las combinaciones posibles en el flujo de ejecución.

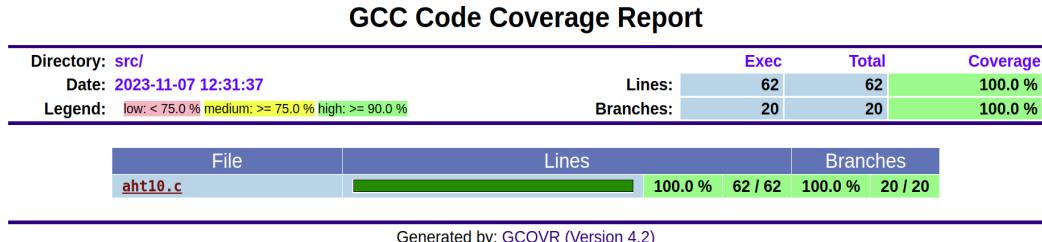


FIGURA 4.2. Informe de cobertura driver AHT10.

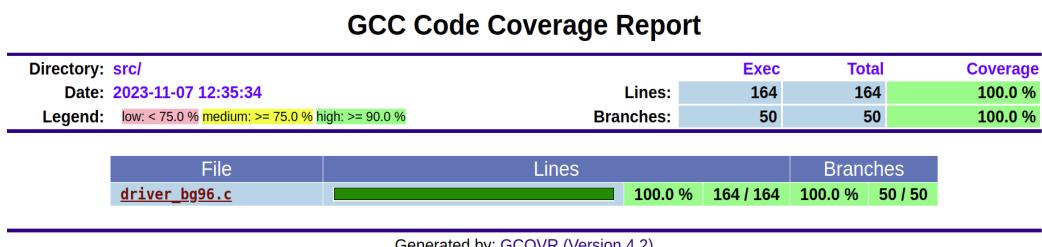


FIGURA 4.3. Informe de cobertura driver BG96.

4.2. Pruebas de la plataforma IoT

4.2.1. Prueba de inyección de mensajes

El objetivo de la prueba de inyección de mensajes a la plataforma IoT, es verificar la llegada de los mensajes mandados por un cliente MQTT al broker de Things-Board. Para realizar el envío de datos al broker, se utilizó el programa mosquitto. Para realizar esta prueba se ejecutó el comando de mosquitto que se muestra en la figura 4.4.

```
mario738@DESKTOP-L5GM2MG:~$ mosquitto_pub -d -q 1 -h "mqtt.thingsboard.cloud" -t "v1/devices/me/telemetry"
-u "ZDqUF9f4VEdj6THx6cAd" -m "{\"humedad_ambiente\":40,\"temperatura_ambiente\":20,\"humedad_suelo\":40,\"radiacion\":2,
\"latitudine\":-21.532614,\"longitude\":-64.762743}"

Client mosq-D8YHs04HRn1B0hfw2 sending CONNECT
Client mosq-D8YHs04HRn1B0hfw2 received CONNACK (0)
Client mosq-D8YHs04HRn1B0hfw2 sending PUBLISH (d0, q1, r0, m1, 'v1/devices/me/telemetry', ... (115 bytes))
Client mosq-D8YHs04HRn1B0hfw2 received PUBACK (Mid: 1, RC:0)
Client mosq-D8YHs04HRn1B0hfw2 sending DISCONNECT
```

FIGURA 4.4. Envio de datos por el cliente MQTT de mosquitto.

Donde:

- Dirección del broker: `-h`
- Tópico: `-t`

- Token: *-u*
- Mensaje en formato JSON: *-m*

Al ejecutar el comando de la figura 4.4, el cliente de mosquitto primeramente se conecta al broker MQTT, luego publica el mensaje en el tópico y finalmente, se desconecta del servidor.

Para comprobar la llegada de los datos al broker de ThingsBoard, se tiene que ir a la sección dispositivos, seleccionar el dispositivo al que se le envió los datos y entrar a la pestaña de última telemetría. En la figura 4.5 se puede ver que los datos enviados llegan correctamente.

Last update time	Key ↑	Value
2023-11-02 11:52:16	humedad_ambiente	40
2023-11-02 11:52:16	humedad_suelo	19
2023-11-02 11:52:16	latitude	-21.532614
2023-11-02 11:52:16	longitude	-64.762743
2023-11-01 09:30:31	position	{"latitude": -21.553789, "longitude": -64.747...}
2023-11-02 11:52:16	radiacion	5
2023-11-02 11:52:16	temperatura_ambiente	32

FIGURA 4.5. Recepción de datos en el broker MQTT.

4.2.2. Prueba del widget de mapa

Dentro de los paneles de visualización, se encuentran mapas que ilustran la ubicación del dispositivo en funcionamiento. En la figura 4.6, se presenta el mapa que indica la posición del prototipo empleado en el trabajo.

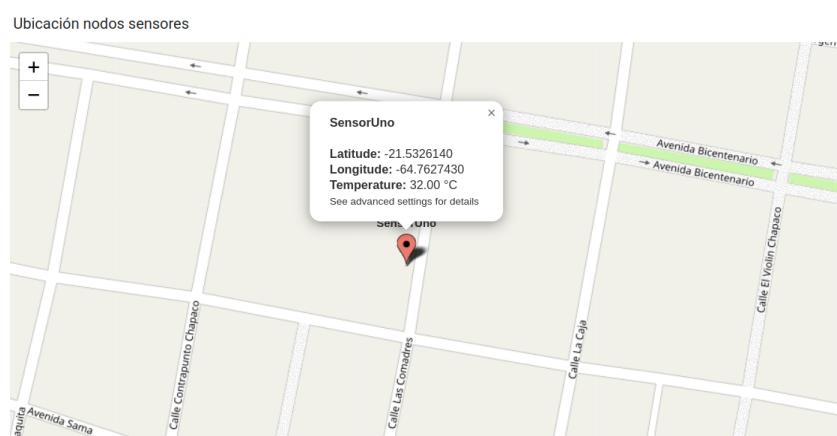


FIGURA 4.6. Ubicación del nodo sensor implementado.

4.2.3. Prueba de la tabla de alarmas del panel visualización

ThingsBoard permite configurar alarmas con respecto a las variables monitoreadas por el sistema. En el panel de visualización de cada sensor se tiene una tabla de alarmas, que muestra las notificaciones de las alarmas que se activaron. En la figura 4.7 se puede ver la notificación de una alarma cuando la temperatura ambiente sobrepasa los 43°C.

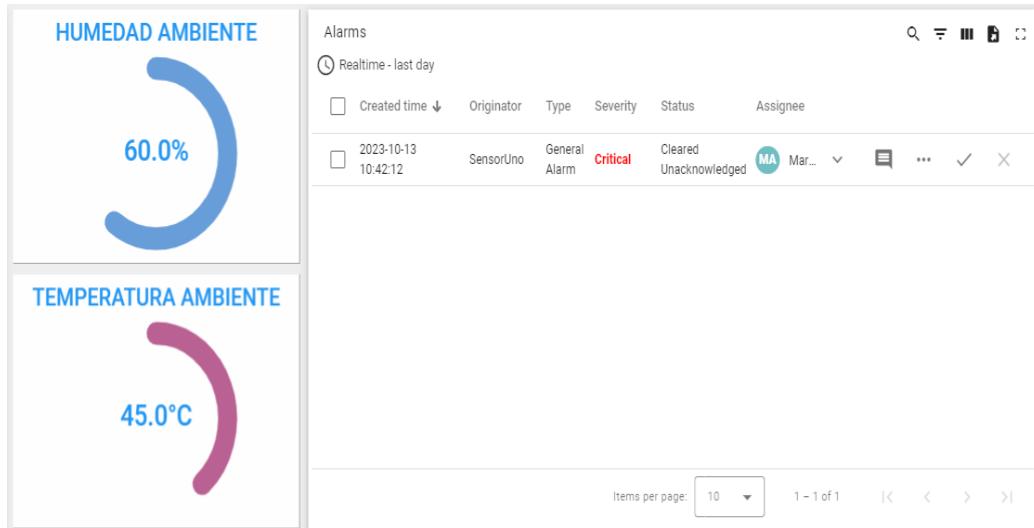


FIGURA 4.7. Tabla de alarmas activas.

4.2.4. Prueba de persistencia de datos

Para realizar la prueba de persistencia de datos, se configuró las gráficas de los paneles de visualización con un entorno de tiempo más amplio. En las gráficas se estableció un rango de tiempo de 7 días. El resultado se muestra en la figura 4.8.

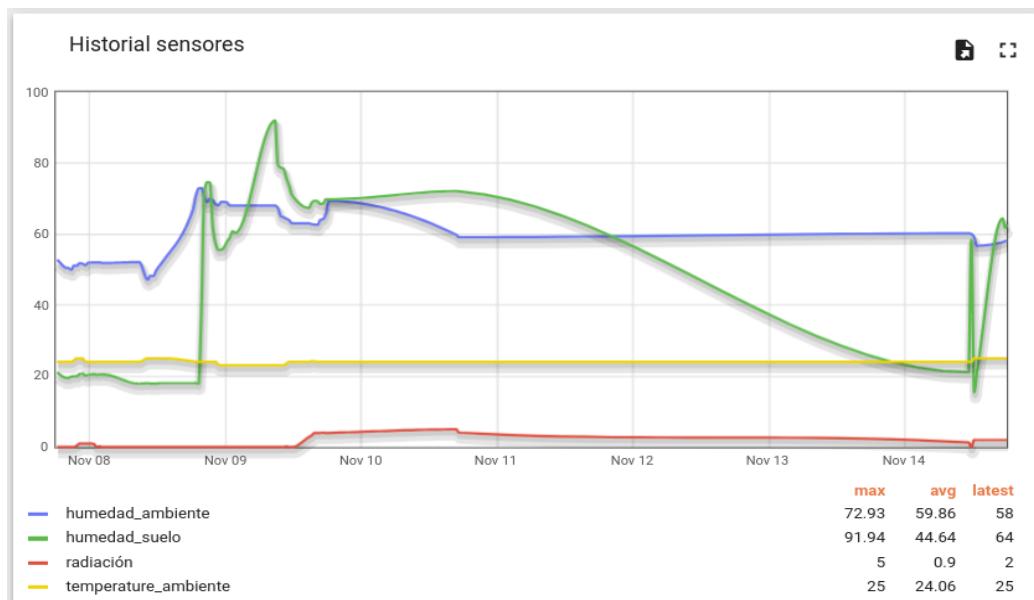


FIGURA 4.8. Persistencia de datos.

4.3. Pruebas de hardware

4.3.1. Prueba de comunicación entre el sensor AHT10 y el microcontrolador

Para comprobar la comunicación por I2C entre el sensor y el microcontrolador se utilizó un analizador lógico. En la figura 4.9 se puede ver la trama capturada por el analizador lógico cuando el firmware escribe en un registro del sensor. En la figura 4.10 se puede apreciar la trama capturada cuando se leen los registros del sensor.

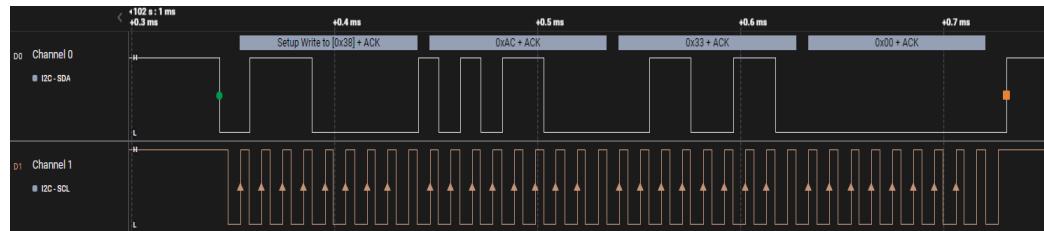


FIGURA 4.9. Trama de escritura al sensor AHT10.

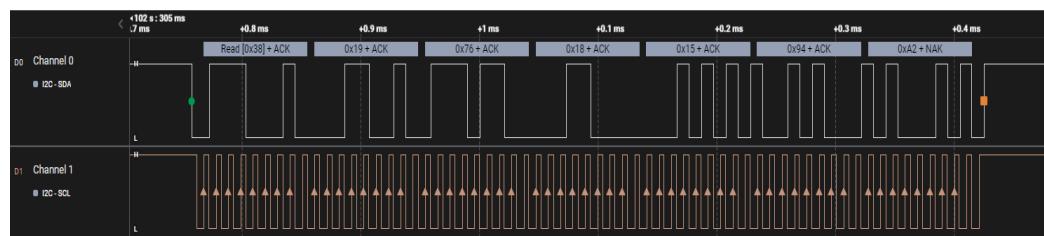


FIGURA 4.10. Trama de lectura del sensor AHT10.

Cuando se realiza la lectura de los registros del sensor se obtienen 6 bytes como se muestra en la figura 4.11. El primer byte contiene información de estado del sensor. De los 5 bytes restantes 20 bits contienen la información de la humedad y 20 bits de la temperatura. Para realizar la conversión de bits a valores representativos para variables físicas, se hace un corrimiento de bits y se aplican las fórmulas que nos proporciona la hoja de datos del sensor.

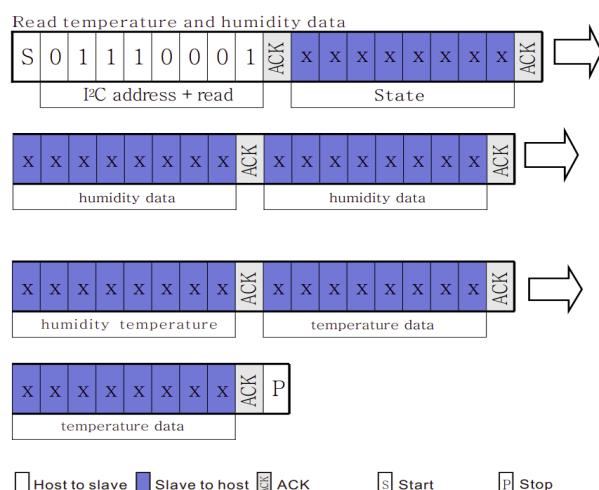


FIGURA 4.11. Bytes de lectura.

4.3.2. Prueba de comunicación entre el módulo BG96 y el microcontrolador

Para probar la comunicación del microcontrolador con el módulo BG96, se utilizó un analizador lógico, que nos permite ver los comandos que envía el microcontrolador y la respuesta del módulo a estos comandos por protocolo serial.

Para la prueba se mandó el siguiente comando:

- Comando: AT+QMTOOPEN=0,"mqtt.thingsboard.cloud",1883\r
- Respuesta: AT+QMTOOPEN=0,"mqtt.thingsboard.cloud",1883\r\r\nOK\r\n\r\n

En la figura 4.12 se puede ver dos canales del analizador lógico: el canal 2 muestra un comando mandado por el microcontrolador al módulo de comunicación y en el canal 3 la respuesta del módulo al comando enviado.

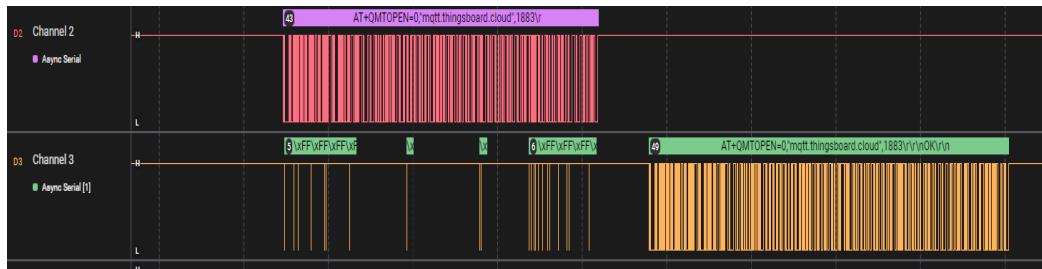


FIGURA 4.12. Envío y recepción de comandos por puerto UART.

4.4. Pruebas funcionales del sistema

Con el propósito de llevar a cabo las pruebas funcionales de todo el sistema, se instaló el prototipo en un campo agrícola, tal como se ilustra en la figura 4.13.



(A) Vista externa.

(B) Vista interna.

FIGURA 4.13. Instalación del prototipo.

4.4.1. Pruebas de lectura de los sensores

Se realizaron varias pruebas al prototipo en diferentes escenarios, a continuación se desarrollarán algunas de las pruebas realizadas.

Caso de uso 1

La prueba consistió en realizar la lectura de los sensores en un momento donde se tenían las siguientes condiciones:

- Tierra con poca humedad
- Temperatura ambiente alta
- Horario de la lectura: 3 p.m

En la figura 4.14 se muestran los resultados de la lectura de los sensores en las condiciones anteriormente mencionadas. Se tiene una lectura de humedad de suelo de 19 %, radiación moderada de 5 y temperatura ambiente del 32°C.

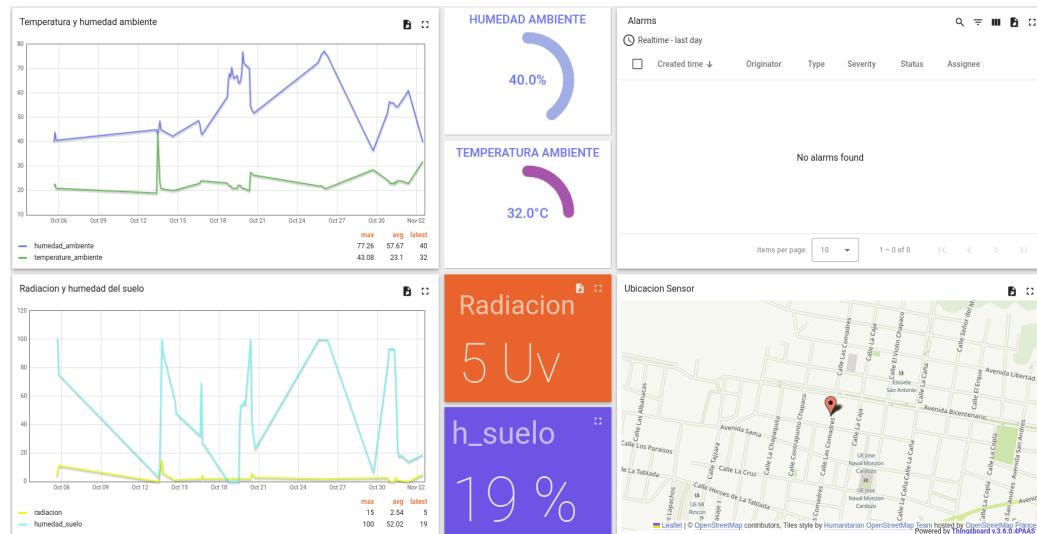


FIGURA 4.14. Panel de visualización con las lecturas obtenidas en el caso de uso 1.

Caso de uso 2

Esta prueba se realiza bajo las siguientes condiciones:

- Tierra con humedad
- Temperatura moderada
- Horario de la lectura: 8 a.m

En la figura 4.15 se muestran los resultados de la lectura. Como resultado tenemos: humedad del suelo del 75 %, nivel de radiación baja de 2, humedad ambiente de 54 % y temperatura ambiente de 23°C.

4.4.2. Prueba del envío de datos al broker MQTT

Una de las tareas más importantes del firmware, es la del manejo de la comunicación con el servidor. Para verificar el funcionamiento de esta tarea, es necesario

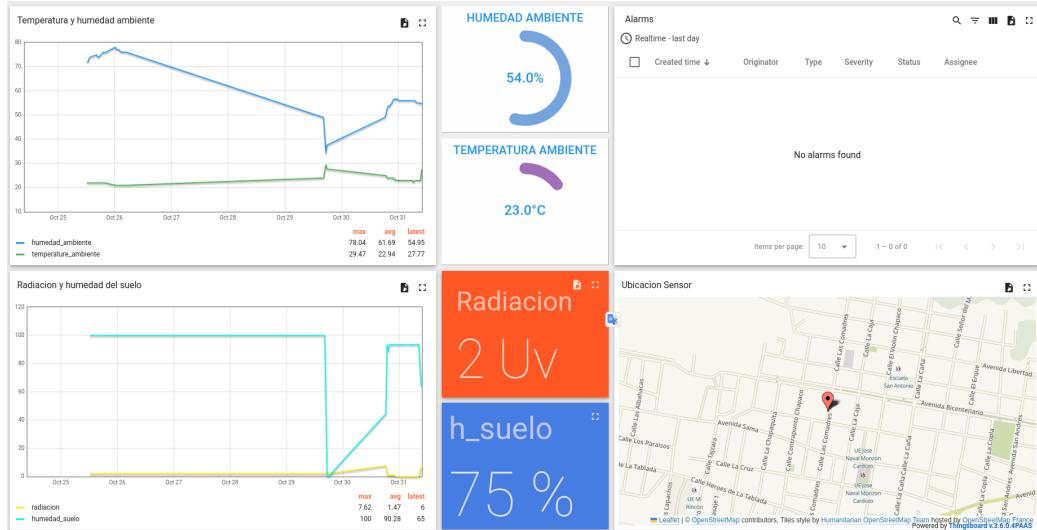


FIGURA 4.15. Panel de visualización con las lecturas obtenidas en el caso de uso 2.

ver la secuencia de comandos que envía el microcontrolador al módulo de comunicación. Para realizar esta prueba se conectó un convertidor UART a USB al conector de depuración que tiene el dispositivo. En la figura 4.16 se puede observar toda la secuencia de comandos que realiza el firmware para realizar el envío de datos a la plataforma IoT. Primeramente, se envía el comando que verifica si el módulo está activo, luego se configura el APN de la red, se conecta al broker MQTT, se publican los datos y finalmente, se realiza el proceso de desconexión del broker.

```

AT                               Comando de estado
OK

AT+QICSGP=1,1,"internet.tigo.bol","","",1
OK                                         Configuración y
AT+QIACT=1                                activación de la red
OK

AT+QMTOOPEN=0,"mqtt.thingsboard.cloud",1883
OK

+QMTOOPEN: 0,0
AT+QMTCOMP=0,"123a56cb9","ZDqUF9f4VEDj6THx6cAd","ZDqUF9f4VEDj6THx6cAd"
OK                                         Conexión broker
                                            MQTT

+QMTCOMP: 0,0,0
AT+QMTPUB=0,0,0,"v1/devices/me/telemetry"
>{"humedad_suelo":21,"humedad_ambiente":48,"radiacion":6,"temperatura_ambiente":24,"position":{""latitude":-21.553789,"longitude":-64.747231}}}}}
OK                                         Publicar datos

+QMTPUB: 0,0,0
AT+QMTCLOSE=0
OK

AT+QMTCLOSE=0
OK                                         Desconexión broker
                                            MQTT
AT+QIDEACT=1
OK
  
```

FIGURA 4.16. Comandos para enviar datos al broker MQTT.

4.4.3. Prueba de envío de alarmas por SMS

El propósito de esta prueba radica en comprobar la eficacia de la alarma encargada de supervisar el nivel de humedad del suelo. Cuando la humedad del suelo

disminuye por debajo del rango aceptable, el firmware envía un mensaje de texto al usuario con la notificación "Humedad de suelo muy baja".

En la figura 4.17 podemos ver los comandos utilizados para enviar un SMS:

- AT+CMGF=1: comando para configurar el módulo en modo texto.
- AT+CMGS="": comando para enviar el mensaje.

En la figura 4.18, se observa que se ha recibido un mensaje de alarma cuando el nivel de humedad descendió por debajo del 10 %.

```
AT+CMGF=1
OK
AT+CMGS="72950576"
> Humedad de suelo muy baja
```

FIGURA 4.17. Comandos para enviar un SMS.

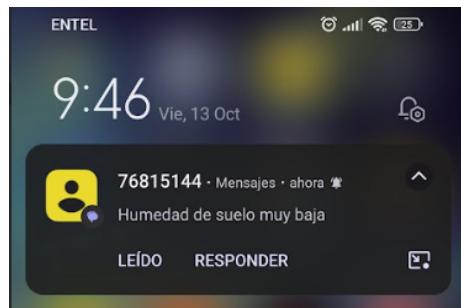


FIGURA 4.18. Recepción del SMS con el mensaje de alarma.

Capítulo 5

Conclusiones

En este capítulo se presentan los resultados obtenidos sobre el trabajo realizado, las herramientas aprendidas durante el transcurso de la carrera que fueron fundamentales para la ejecución y se proponen ideas que permitan mejorar lo desarrollado.

5.1. Resultados obtenidos

En el trabajo realizado se logró implementar de forma exitosa un prototipo correspondiente a un sistema de monitoreo de cultivos agrícolas. A través de las pruebas realizadas se pudo validar y demostrar su correcto funcionamiento. Se pueden nombrar los siguientes logros obtenidos:

- Se fabricó un prototipo funcional y se lo instaló en un cultivo agrícola para la realización de pruebas integrales del sistema, garantizando cumplir todos los requerimientos funcionales.
- Se diseñó el esquemático y el PCB de la tarjeta electrónica para el prototipo.
- Se desarrolló el firmware sobre un sistema operativo de tiempo real, para optimizar los recursos del microcontrolador y hacer un código más modular y escalable.
- Se configuraron paneles de visualización en ThingsBoard. Permitiendo visualizar las variables ambientales adquiridas por el prototipo instalado en el cultivo agrícola.
- Se implementó el envío de alarmas del sistema al usuario mediante SMS.
- Se utilizó Git para control de versiones, ayudó a tener la trazabilidad y flexibilidad necesaria para llevar a cabo el desarrollo de manera ordenada.

Los requerimientos del trabajo fueron cubiertos de acuerdo con la planificación, con la siguiente modificación:

- Se cambió el requerimiento de utilizar HTTP como protocolo de envío de datos a la nube. Se utilizó MQTT porque es un protocolo más rápido, liviano y utiliza el modelo publicador/suscriptor.

En el desarrollo del trabajo se aplicaron muchos de los conocimientos adquiridos a lo largo de la Carrera de Especialización en Sistemas Embebidos. Entre los conocimientos aplicados, se pueden destacar los adquiridos en las siguientes asignaturas:

- Programación de Microcontroladores: se utilizaron conceptos básicos de programación, conceptos de modularización y uso de patrones de diseño.
- Protocolos de Comunicación en Sistemas Embebidos: se utilizaron protocolos de comunicación I2C, UART y MQTT.
- Sistemas Operativos de Tiempo Real: se implementó el firmware sobre freeRTOS. Se utilizaron colas y semáforos, para la comunicación y sincronización entre tareas.
- Testing de Software en Sistemas Embebidos: uso de TDD para el desarrollo de los drivers. Se utilizó Ceedling para el desarrollo de pruebas automáticas.
- Diseño de Circuitos Impresos: se utilizaron buenas prácticas de diseño electrónico para desarrollar el esquemático y el circuito impreso del prototipo. Se utilizó KICAD como herramienta de diseño electrónico.

5.2. Próximos pasos

Si bien se lograron obtener los resultados esperados, a futuro se puede continuar con el desarrollo en varias medidas. A continuación, se detallan los aspectos que sería conveniente tomar en consideración:

- Realizar un nuevo diseño del hardware que integre a todo el sistema y no utilice módulos por separado.
- Implementar un mecanismo de actualización de firmware remoto.
- Incluir soporte para trabajar con energías renovables.
- Aumentar la seguridad al enviar los datos al servidor utilizando SSL.

Bibliografía

- [1] Sara Oleiro Araujo y col. «Characterising the Agriculture 4.0». En: Agronomy 11.4, 2021, pág. 667.
- [2] Marco Centenaro y col. «Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios». En: IEEE Wireless Communications 23.5, 2016, págs. 60-67.
- [3] Ferrovial. *Internet de las cosas(IoT)*. Visitado: 2023-05-27. URL: <https://www.ferrovial.com/es/recursos/internet-de-las-cosas/>.
- [4] Libelium. *Smart Agriculture PRO,TECHNICAL GUIDE*. Visitado: 2023-05-27. URL: <https://development.libelium.com/agriculture-sensor-guide/>.
- [5] WiseConn. *Nodo RF-M1*. Visitado: 2023-05-27. URL: <https://www.wiseconn.cl/dropcontrol/hardware/rf-m1/>.
- [6] ThingsBoard. *Descripción de la plataforma IoT,Definición de ThingsBoard*. Visitado: 2023-05-27. URL: <https://thingsboard.io/>.
- [7] STMicroelectronics. *Especificacion de Producto,STM32 Nucleo-32 boards*. Ultima actualizacion 2019-06-05 V8.0. URL: https://www.st.com/resource/en/data_brief/nucleo-l432kc.pdf.
- [8] MikroElectronic. *Especificacion de Producto,LTE IOT 2 CLICK*. Visitado: 2023-05-27. URL: <https://www.mikroe.com/lte-iot-2-click>.
- [9] SSDIELECT ELECTRONICA SAS. *Especificacion de Producto,AHT10 SENSOR DE TEMPERATURA Y HUMEDAD I2C*. Visitado: 2023-05-27. URL: <https://ssdiselect.com/temperatura/3885-aht10.html>.
- [10] naylampmechatronics. *Especificacion de Producto,MÓDULO SENSOR DE LUZ ULTRAVIOLETA (UV) ML8511*. Visitado: 2023-05-27. URL: <https://ssdiselect.com/temperatura/3885-aht10.html>.
- [11] PANAMAHITEK. *Especificacion de Producto,Módulo HL-69: Un sensor de humedad de suelo*. Visitado: 2023-05-27. URL: <https://panamahitek.com/modulo-hl-69-un-sensor-de-humedad-de-suelo/>.
- [12] STMicroelectronics. *Descripción del producto,Integrated Development Environment for STM32*. Visitado: 2023-05-27. URL: <https://www.st.com/en/development-tools/stm32cubeide.html>.
- [13] FreeRTOS. *Acerca del Sistema Operativo,Descripción General*. Visitado: 2023-05-27. URL: <https://www.freertos.org/RTOS.html>.
- [14] CEDLING. *Descripción del producto,Descripción General*. Visitado: 2023-05-27. URL: <http://www.throwtheswitch.org/ceedling>.
- [15] rohde-schwarz. *Descripción del protocolo,Qué es UART*. Visitado: 2023-05-27. URL: https://www.rohde-schwarz.com/es/productos/test-y-medida/essentials-test-equipment/digital-oscilloscopes/que-es-uart_254524.html.
- [16] sparkfun. *Definicion del Protocolo*. Visitado: 2023-05-27. URL: <https://learn.sparkfun.com/tutorials/i2c/all>.
- [17] MQTT. *Descripción del protocolo,Introducción a MQTT*. Visitado: 2023-05-27. URL: <https://mqtt.org/>.

- [18] connectamericas. *Descripción de la empresa, Definición de UBIDOTS*. Visitado: 2023-05-27. URL: <https://connectamericas.com/es/company/ubidots>.
- [19] BrowserStack. *Definición de la metodología de desarrollo TDD*. Visitado: 2023-11-10. URL: <https://www.browserstack.com/guide/what-is-test-driven-development>.