

# Índice general

Resumen	I
<b>1. Introducción general</b>	
1.1. Introducción . . . . .	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
<b>2. Introducción específica</b>	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. CEDDLING . . . . .	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
<b>3. Diseño e implementación</b>	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 . . . . .	16

# Índice general

Resumen	I
<b>1. Introducción general</b>	
1.1. IoT 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
<b>2. Introducción específica</b>	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. CEDDLING . . . . .	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
<b>3. Diseño e implementación</b>	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 . . . . .	16

3.4. Controladores implementados . . . . .	18
3.4.1. Controlador sensor AHT10 . . . . .	18
3.4.2. Controlador módulo de comunicación BG96 . . . . .	19
3.5. Desarrollo del hardware . . . . .	23
3.5.1. Esquematico . . . . .	23
3.5.2. PCB del hardware . . . . .	25
3.5.3. Fabricacion circuito impreso . . . . .	25
3.6. Paneles de visualizacion . . . . .	26
3.6.1. Panel principal . . . . .	26
3.6.2. Panel nodo sensor . . . . .	27
<b>4. Ensayos y resultados</b>	<b>29</b>
<b>5. Conclusiones</b>	<b>31</b>
5.1. Conclusiones generales . . . . .	31
5.2. Próximos pasos . . . . .	31
<b>Bibliografía</b>	<b>33</b>

3.4. Controladores implementados . . . . .	18
3.4.1. Controlador sensor AHT10 . . . . .	18
3.4.2. Controlador 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 circuito impreso . . . . .	23
3.6. Paneles de visualización . . . . .	24
3.6.1. Panel principal . . . . .	24
3.6.2. Panel nodo sensor . . . . .	25
<b>4. Conclusiones</b>	<b>27</b>
4.1. Conclusiones generales . . . . .	27
4.2. Próximos pasos . . . . .	27
<b>Bibliografía</b>	<b>29</b>

## 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.	6
2.2. Módulo LTE IOT 2 CLICK.	6
2.3. Sensor AHT10.	7
2.4. Módulo Sensor ML8511.	7
2.5. Módulo Sensor HL-69.	8
2.6. Logo FreeRTOS.	8
2.7. Arquitectura de publicación/suscripción de MQTT.	9
2.8. Ejemplo interfaz gráfica Ubidots.	10
2.9. Ejemplo interfaz gráfica ThingsBoard.	10
3.1. Diagrama general del sistema IoT.	11
3.2. Capas del firmware.	12
3.3. Diagrama de flujo inicio firmware.	14
3.4. Diagrama de flujo tarea loop.	15
3.5. Diagrama de flujo tarea de adquisición de datos.	16
3.6. Diagrama de flujo tarea manejador de alarmas.	16
3.7. Diagrama de flujo 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.	23
3.11. Esquemático conectores modulos.	23
3.12. Esquemático interfaz de debug.	24
3.13. Esquemático conectores sensores.	24
3.14. Esquemático conectores alimentación.	24
3.15. PCB del proyecto.	25
3.16. 3D del tarjeta desarrollada.	25
3.17. PCB ensamblado.	25
3.18. Panel principal de la interfaz gráfica.	26
3.19. Panel nodo sensor.	27

## 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 <sup>1</sup> .	6
2.2. Módulo LTE IOT 2 CLICK <sup>2</sup> .	6
2.3. Sensor AHT10 <sup>3</sup> .	7
2.4. Módulo Sensor ML8511.	7
2.5. Módulo sensor HL-69 <sup>4</sup> .	8
2.6. Logo FreeRTOS <sup>5</sup> .	8
2.7. Arquitectura de publicación/suscripción de MQTT <sup>6</sup> .	9
2.8. Ejemplo interfaz gráfica Ubidots <sup>7</sup> .	10
2.9. Ejemplo interfaz gráfica ThingsBoard <sup>8</sup> .	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.	21
3.11. Conector módulo BG96 y NUCLEO-L432KC.	22
3.12. Esquemático interfaz de debug.	22
3.13. Esquemático conectores sensores.	22
3.14. Esquemático conectores alimentación.	23
3.15. PCB del proyecto.	23
3.16. Modelo 3D de la tarjeta.	23
3.17. PCB ensamblado.	24
3.18. Panel principal de la interfaz gráfica.	24
3.19. Panel nodo sensor.	25

## 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. Introducción

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.

## 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. IoT 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 proyecto 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].

- 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.



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 [Ubidots](#).

### 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 [Ubidots](#) [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 [6].

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.

## 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-L432KC.

### 2.1.2. Módulo De Comunicación LTE IOT 2 CLICK

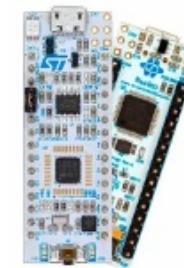
LTE IoT 2 click que se muestra en la 2.2 está equipado con el módulo BG96 LTE de Quectel Wireless Solutions, que admite tecnologías LTE CAT M1 y NB1, desarrolladas para aplicaciones IoT. Además, admite EGPRS a 850/900/1800/1900 MHz, lo que significa que se puede usar globalmente; no está restringido a ninguna región [7].

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 CLICK.

FIGURA 2.1. Plataforma de desarrollo NUCLEO-L432KC<sup>1</sup>.

### 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 CLICK<sup>2</sup>.

<sup>1</sup>Datasheet [https://www.st.com/resource/en/user\\_manual/um1956-stm32-nucleo32-boards-mb1180-stmicroelectronics.pdf](https://www.st.com/resource/en/user_manual/um1956-stm32-nucleo32-boards-mb1180-stmicroelectronics.pdf)

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

**2.1. Componentes principales de hardware**

7

**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 [8].

FIGURA 2.3. Sensor AHT10.<sup>3</sup>**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 ( $mW/cm^2$ ) [9].



FIGURA 2.4. Módulo Sensor ML8511.

**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 a agua, líquidos y humedad [10].

**2.1. Componentes principales de hardware**

7

**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 AHT10.<sup>3</sup>**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.<sup>4</sup>**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 a agua, líquidos y humedad [11].

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

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

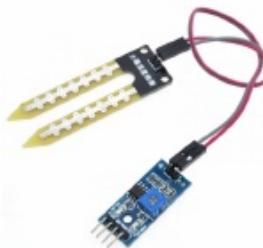


FIGURA 2.5. Módulo Sensor HL-69.

## 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++ avanzada con funciones de configuración de periféricos, generación de código, compilación de código y depuración para microcontroladores y microporcesadores 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 [11].

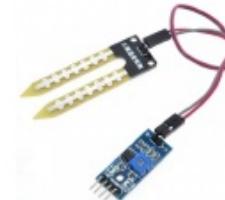
Integra las funcionalidades de configuración y creación de proyectos de STM32 de STM32CubeMX para ofrecer una experiencia de herramienta todo en uno y ahorrar tiempo de instalación y desarrollo [11].

### 2.2.2. FreeRTOS

FreeRTOS es un sistema operativo en tiempo real (RTOS) líder en el mercado para microcontroladores y pequeños microporcesadores. 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 [12]. El logo de FreeRTOS se muestra en la figura 2.6.



FIGURA 2.6. Logo FreeRTOS.

FIGURA 2.5. Módulo sensor HL-69<sup>5</sup>.

## 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++ avanzada con funciones de configuración de periféricos, generación de código, compilación de código y depuración para 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 microporcesadores. 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 FreeRTOS<sup>6</sup>.

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

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

## 2.3. Protocolos de Comunicación

9

### 2.2.3. CEDDLING

Ceddling 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. Ceddling está dirigido principalmente al desarrollo basado en pruebas en C y está diseñado para reunir CMock, Unity y CException [13].

## 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 [14].

### 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 [15].

### 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 [16]. En la figura 2.7 se muestra la arquitectura del protocolo MQTT.



FIGURA 2.7. Arquitectura de publicación/suscripción de MQTT.

## 2.3. Protocolos de Comunicación

9

### 2.2.3. CEDDLING

Ceddling 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. Ceddling 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 MQTT.<sup>7</sup>

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

## 2.4. Plataformas IoT

### 2.4.1. Ubidots

Ubidots una plataforma de IoT que habilita la toma de decisiones a empresas de integración de sistemas a nivel global. Este producto 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 [17]. En la figura 2.8 se muestra un ejemplo de interfaz gráfica en Ubidots.

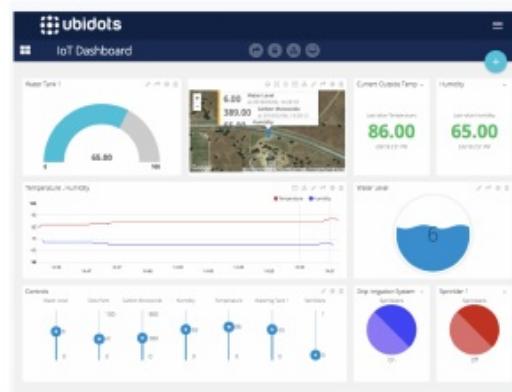


FIGURA 2.8. Ejemplo interfaz gráfica Ubidots.

### 2.4.2. ThingsBoard

ThingsBoard es una plataforma IoT de código abierto para la recopilación, el procesamiento, la visualización y la gestión de dispositivos de datos. Permite la conectividad de dispositivos a través de protocolos IoT estándar de la industria: MQTT, CoAP y HTTP, y admite implementaciones en la nube y locales. ThingsBoard combina escalabilidad, tolerancia a fallas y rendimiento [18]. En la figura 2.9 se muestra un ejemplo de una interfaz gráfica desarrollada en ThingsBoard.

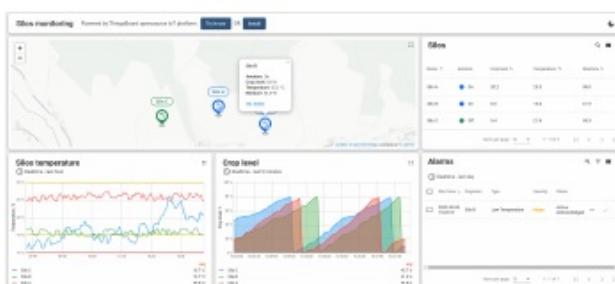


FIGURA 2.9. Ejemplo interfaz gráfica ThingsBoard.

## 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 [6]. En la figura 2.8 se muestra un ejemplo de interfaz gráfica en Ubidots.

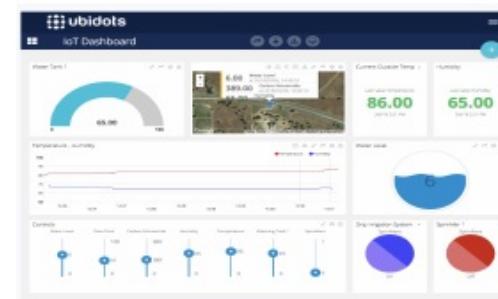


FIGURA 2.8. Ejemplo interfaz gráfica Ubidots<sup>8</sup>.

### 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 [18]. En la figura 2.9 se muestra un ejemplo de una interfaz gráfica desarrollada en ThingsBoard.

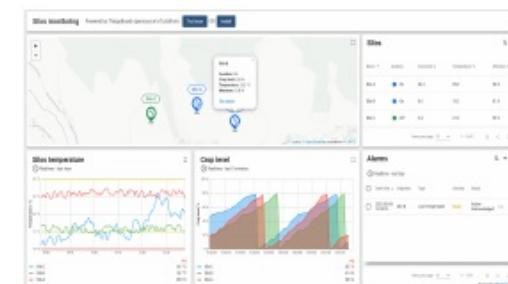


FIGURA 2.9. Ejemplo interfaz gráfica ThingsBoard<sup>9</sup>.

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

<sup>9</sup>Imagen 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 proyecto que consta de tres capas: percepción, red y aplicación.

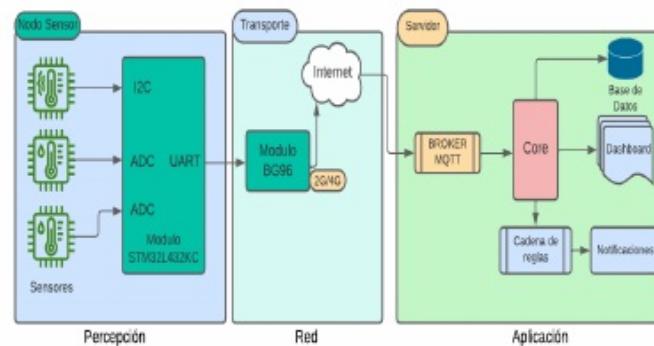


FIGURA 3.1. Diagrama general del sistema IoT.

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 preprocessamiento 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, 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.

## 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.

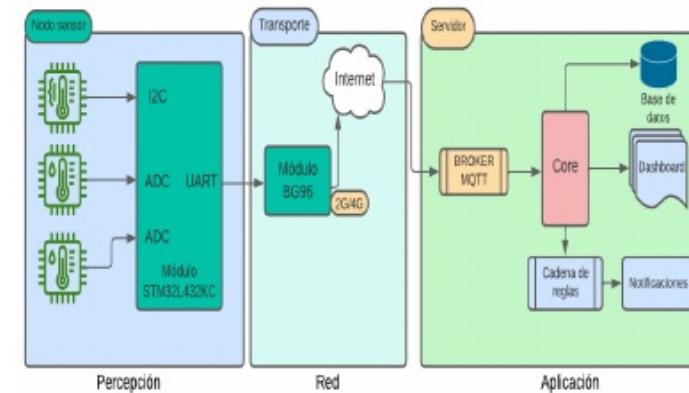


FIGURA 3.1. Diagrama general del sistema IoT.

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 preprocessamiento 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 cuanto a la red se utilizó un módulo quectel BG96 que puede conectarse a la red 2G, 4G, NB-IoT automáticamente dependiendo del nivel de la red en el lugar de la implementación del nodo sensor, se comunica con el microcontrolador por comandos AT por puerto UART.
- **Capa de aplicación:** En la capa de aplicación, se utilizó ThingsBoard como plataforma IoT que nos brinda los microservicios de broker MQTT como puerta de entrada al servidor, base de datos para el almacenamiento, nos brinda la interfaz gráfica para la visualización de los datos y nos permite gestionar las alarmas del sistema.

### 3.2. Arquitectura de firmware

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



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 de microcontroladora 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.

- **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.

### 3.2. Arquitectura de firmware

13

- 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 de 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 drivers para el modulo de comunicacion BG96, AHT10.

- Driver BG96: **Las** funciones más importantes que proporciona el **driver**:
  - Estado del módulo.
  - Descripción del módulo.
  - Configuración APN de la red.
  - Conexión TCP.
  - Conexión al broker MQTT.
- Driver AHT10: **Se** desarrollo utilizando la hoja de datos del sensor, proporciona **las** funciones mas importante de **inicializacion** 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 **nos** permite hacer un código más escalable.

Se implementaron cuatro tareas.

- Loop del sistema: **Esta** tarea es la que **nos** da 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.

### 3.2. Arquitectura de firmware

13

- 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** desarrollo 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.

- 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, interrupciones.

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



FIGURA 3.3. Diagrama de flujo inicio firmware.

Lo primero que el firmware realiza es la configuración del hardware del microcontrolador, luego se inicializan los drivers del modulo de comunicación celular BG96 y el driver del sensor de humedad y temperatura AHT10, posteriormente se crean las tareas, colas y los semáforos del sistema y finalmente se da el control del sistema al scheduler del sistema operativo.

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 secuencialidad del sistema es manejada por la tarea loop, la figura 3.4 muestra el diagrama de flujo de la tarea loop. La tarea comienza iniciando el timer del sistema el cual es el encargado de manejar el tiempo en que se repetira el ciclo de la tarea loop, luego la tarea se bloquea en el un semáforo que sera desbloqueada en el momento de la ejecución del handler de la interrupción del timer, luego se envia datos por la cola de adquisición de datos y server mqtt para levantar el servidor y la tarea se vuelve a bloquear en el semáforo, cuando se desbloquea pregunta si se logró levantar el servidor, si se logró se manda un mensaje por la cola de server mqtt con el evento de enviar datos y se bloquea nuevamente pero

### 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 cola de adquisición de datos para realizar la lectura de los sensores y un evento a una 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 de las alarmas activas del sistema. Finalmente después de monitorear las alarmas la tarea manda un evento a la cola de servidor

## 3.3. Desarrollo del firmware

15

si no se logró levantar el servidor no se enviaran datos, luego se envía un evento a la cola de alarmas y se bloquea nuevamente esperando que envíen las alarmas, cuando se desbloquea se manda un evento a la cola de server mqtt para desconectar el servidor y se bloquea la tarea en el semáforo esperando la desconexión finalmente se inicia nuevamente el timer y mandamos al sistema a modo de bajo consumo.

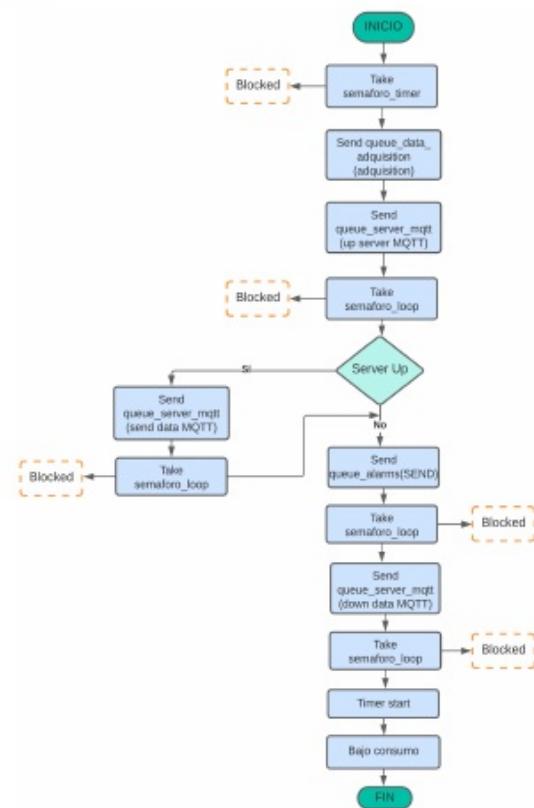


FIGURA 3.4. Diagrama de flujo tarea loop.

## 3.3.2. Tarea adquisición de datos

La figura 3.5 muestra la secuencia repetitiva que realiza la tarea de adquisición de datos, la tarea inicia creando algunas variables locales que se utilizará en la tarea, luego revisa si hay algún dato en la cola de adquisición de datos si no hay datos la tarea se bloquea pero si existen datos se realiza la lectura de todos los sensores para posteriormente enviar los datos recolectados por la cola de data y alarmas.

## 3.3. Desarrollo del firmware

15

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.

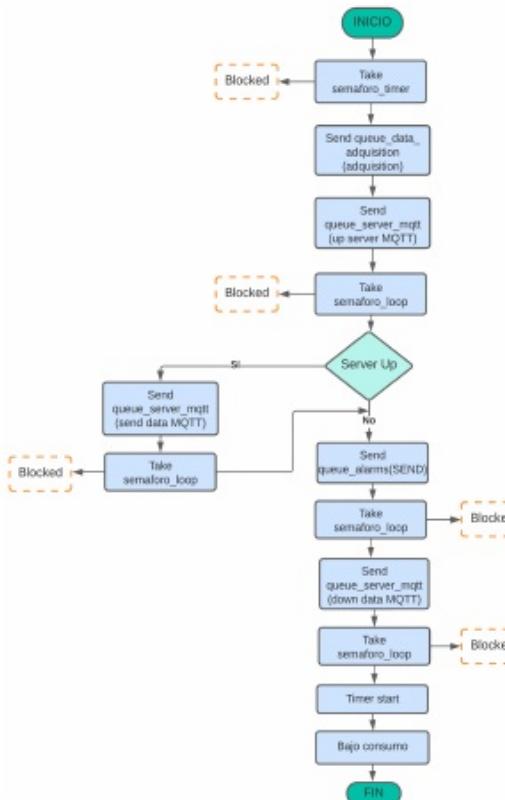


FIGURA 3.4. Diagrama de flujo de la tarea loop.

## 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.

## 3.3.3. Tarea manejador de alarmas

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

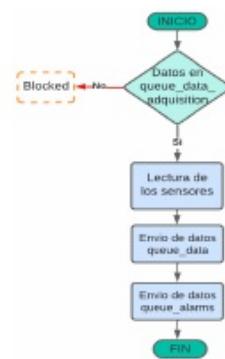


FIGURA 3.5. Diagrama de flujo tarea de adquisicion de datos.

### 3.3.3. Tarea manejador de alarmas

El control de las alarmas se realiza a través de la tarea de manejador de alarmas la figura 3.6 describe la secuencia de la tarea, al entrar al bucle infinito lo primero que se realiza es revisar si existe algún dato en la cola de alarmas si no hay datos la tarea se bloquea hasta que alguien envíe un dato a la cola, si hay datos se pregunta qué evento contiene el dato recibido, si el evento es de MONITOREAR lo que se hace es ver si el dato del sensor de humedad es menor al valor mínimo aceptado si es así se pone una variable en alto advirtiendo al sistema que hay una alarma, si se recibió el evento de SEND se revisa si hay alarmas activas si hay se envía un mensaje de alarma por mensaje de texto al número configurado al inicio del sistema.

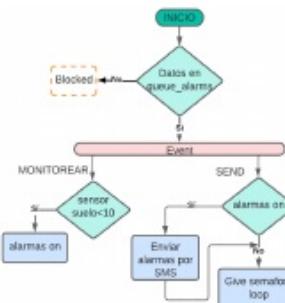


FIGURA 3.6. Diagrama de flujo tarea manejador de alarmas.

### 3.3.4. Tarea manejador del servidor

La última tarea que se implementó es la tarea que maneja la conexión del servidor que se describe en la figura 3.7. La tarea comienza esperando datos en la cola del servidor mqtt si no hay datos la tarea se bloquea, si hay datos se revisa que

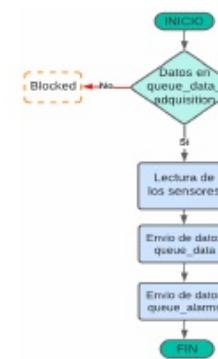


FIGURA 3.5. Diagrama de flujo tarea de adquisicion de datos.

Al entrar al bucle infinito lo primero que realiza la tarea es revisar la cola de alarmas. Si existen datos en la cola, se analiza el evento que contiene el dato recibido. Se tiene dos posibles eventos: monitorear y enviar. Si el evento es monitorear se compara el valor del sensor de humedad con un valor mínimo permitido, si es menor se activa una alarma advierte al sistema que ocurrió esto a través de una variable. Si el evento es enviar se revisa si hay alarmas activas, enviando un mensaje de texto si así fuera. Si no hay datos en la cola de alarmas la tarea se bloquea.

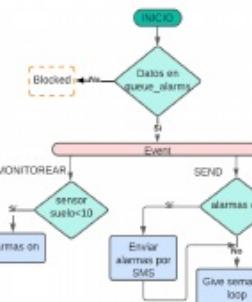


FIGURA 3.6. Diagrama de flujo de la tarea manejador de alarmas.

### 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 entra a la máquina de estados que se muestra en la figura 3.9 encargada de levantar 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, armar la trama y publica los datos al broker MQTT.

### *3.3. Desarrollo del firmware*

12

evento es el que se recibió tenemos tres posibles eventos UP, DOWN y SEND, si recibimos el evento de UP la tarea entra a una máquina de estados para levantar el servidor la figura 3.9 muestra la máquina de estados del evento UP, si el evento es de DOWN la tarea ejecuta la máquina de estados que se ve en la figura 3.8 y si el evento es SEND la tarea espera que existan datos en la cola de data para armar la trama y publicar los datos al broker MQTT.

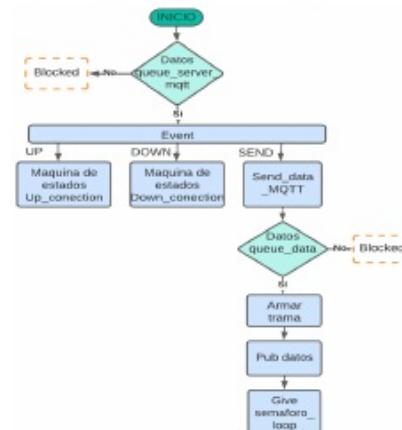


FIGURA 3.7. Diagrama de flujo tarea conexión server MQTT

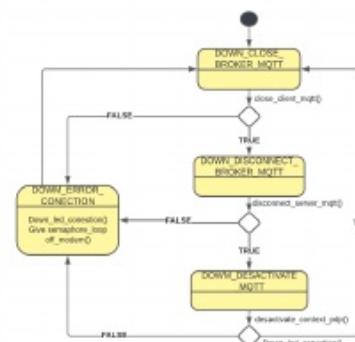


FIGURA 3.8. Máquina de estados down servidor

### *3.3. Desarrollo del firmware*

17

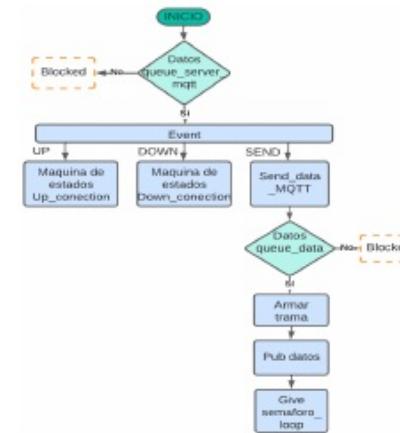


FIGURA 3.7. Diagrama de flujo de la tarea conexión server MQTT.

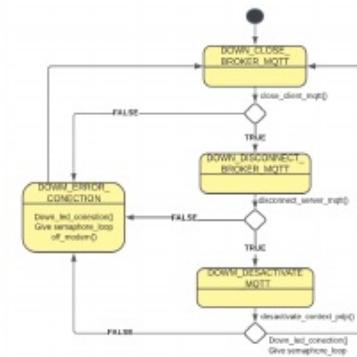


FIGURA 3.8. Máquina de estados down servidor.

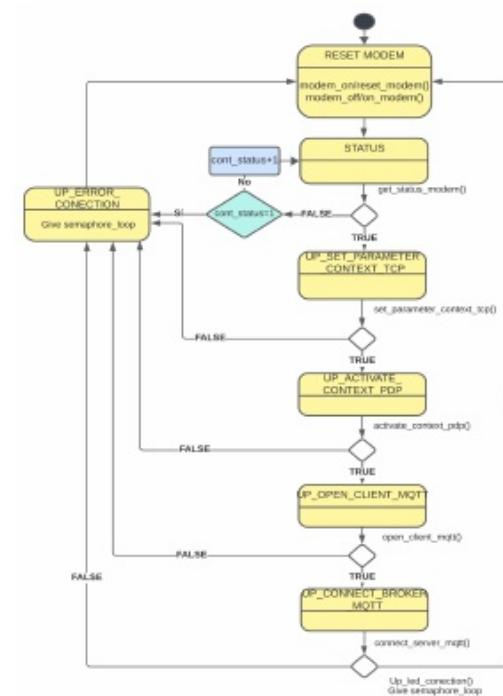


FIGURA 3.9. Máquina de estados up servidor.

### 3.4. Controladores implementados

Se implementaron dos controladores para el modulo de comunicacion BG96 y para el sensor de humedad y temperatura ambiente AHT-10.

#### 3.4.1. Controlador sensor AHT10

En el código 3.1 se ven las funciones más utilizadas del driver en el firmware:

```

1 //Estructura para manejar los datos del driver del sensor
2 typedef struct
3 {
4     aht10WriteFcn_t writeI2C;
5     aht10ReadFcn_t readI2C;
6     delay1ms_t      delay_ms_I2C;
7     aht10_status_fnc status_fun;
8 } aht10_config_t;
9
10 //Funcion para inicializar el driver
  
```

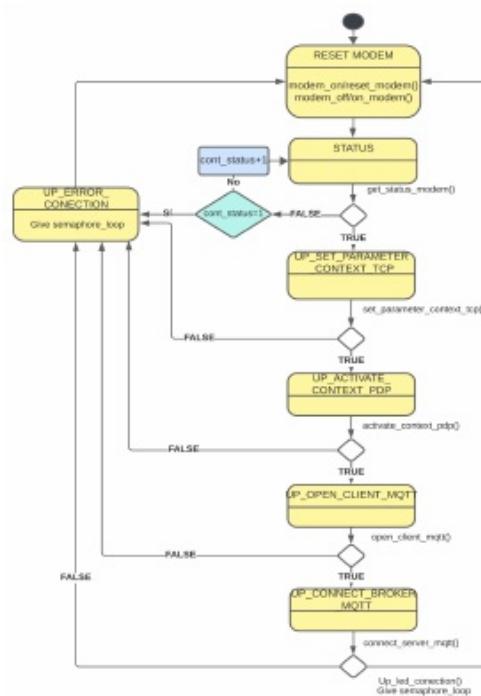


FIGURA 3.9. Máquina de estados up servidor.

### 3.4. Controladores implementados

Se implementaron dos controladores para el modulo de comunicacion BG96 y para el sensor de humedad y temperatura ambiente AHT10.

#### 3.4.1. Controlador 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.

### 3.4. Controladores implementados

19

```

11 void aht10Init(aht10_config_t *obj, aht10WriteFnc_t fncWritePort,
12                 aht10ReadFnc_t fncReadPort, delayMs_t fncDelayPort)
13 {
14     obj->writeI2C=fncWritePort;
15     obj->readI2C=fncReadPort;
16     obj->delay_ms_I2C=fncDelayPort;
17 }
18 //Funcion para obtener el valor de la humedad
19 aht10_status_fnc aht10_get_humidity(aht10_config_t*obj, uint8_t *data)
20 {
21     if (obj== NULL)
22     {
23         return AHT10_ERROR;
24     }
25     obj->status_fun=AHT10_ERROR;
26     uint8_t bufferRead[6]={0};
27     uint32_t data_humidity=0;
28     obj->status_fun=aht10_launch_measurement(obj);
29     if (obj->status_fun==AHT10_OK)
30     {
31         obj->status_fun= obj->readI2C(AHT10_ADDRESS_SLAVE,bufferRead ,6);
32         if (obj->status_fun==AHT10_OK)
33         {
34             data_humidity =(( uint32_t)bufferRead[1]<<16) + (( uint16_t)
35             bufferRead[2]<<8) | (bufferRead [3])>>4;
36             *data= HUMEDITY(data_humidity);
37         }
38     }
39 //Funcion para obtener el valor de la temperatura
40 aht10_status_fnc aht10_get_temperature(aht10_config_t*obj, int8_t *data)
41 {
42     if (obj== NULL)
43     {
44         return AHT10_ERROR;
45     }
46     uint8_t buffer_read[6]={0};
47     uint32_t data_temperature=0;
48     obj->status_fun=AHT10_ERROR;
49     obj->status_fun=aht10_launch_measurement(obj);
50     if (obj->status_fun==AHT10_OK)
51     {
52         obj->status_fun=obj->readI2C(AHT10_ADDRESS_SLAVE ,buffer_read ,6);
53         if (obj->status_fun==AHT10_OK)
54         {
55             data_temperature=((uint32_t)(buffer_read [3] & 0x0F)<<16) + ((
56             uint16_t) buffer_read[4]<<8) | buffer_read [5];
57             *data= TEMPERATURE(data_temperature);
58         }
59     }
60     return obj->status_fun;
61 }
```

CÓDIGO 3.1. Funciones principales del driver del sensor AHT10.

#### 3.4.2. Controlador módulo de comunicación BG96

En el código 3.2 se muestran las funciones mas utilizadas por el firmware del driver que son las de configuración y activación del APN, conexión y desconexión al servidor MQTT.

### 3.4. Controladores implementados

19

```

1 #define TEMPERATURE(A) ((int8_t) ((A *0.000191)-50)
2 #define HUMEDITY(A) ((uint8_t) (A *0.000095)
3 //Funcion para obtener el valor de la humedad
4 aht10_status_fnc aht10_get_humidity(aht10_config_t*obj, uint8_t *data)
5 {
6     if (obj== NULL)
7     {
8         return AHT10_ERROR;
9     }
10    obj->status_fun=AHT10_ERROR;
11    uint8_t bufferRead [6]={0};
12    uint32_t data_humidity=0;
13    obj->status_fun=aht10_launch_measurement(obj);
14    if (obj->status_fun==AHT10_OK)
15    {
16        obj->status_fun= obj->readI2C(AHT10_ADDRESS_SLAVE,bufferRead ,6);
17        if (obj->status_fun==AHT10_OK)
18        {
19            data_humidity =((( uint32_t)bufferRead[1]<<16) + (( uint16_t)
20            bufferRead[2]<<8) | (bufferRead [3])>>4;
21            *data= HUMEDITY(data_humidity);
22        }
23    }
24 }
25 //Funcion para obtener el valor de la temperatura
26 aht10_status_fnc aht10_get_temperature(aht10_config_t*obj, int8_t *data)
27 {
28     if (obj== NULL)
29     {
30         return AHT10_ERROR;
31     }
32     uint8_t buffer_read[6]={0};
33     uint32_t data_temperature=0;
34     obj->status_fun=AHT10_ERROR;
35     obj->status_fun=aht10_launch_measurement(obj);
36     if (obj->status_fun==AHT10_OK)
37     {
38         obj->status_fun=obj->readI2C(AHT10_ADDRESS_SLAVE ,buffer_read ,6);
39         if (obj->status_fun==AHT10_OK)
40         {
41             data_temperature=((uint32_t)(buffer_read [3] & 0x0F)<<16) + ((
42             uint16_t) buffer_read[4]<<8) | buffer_read [5];
43             *data= TEMPERATURE(data_temperature);
44         }
45     }
46 }
```

CÓDIGO 3.1. Funciones de lectura de humedad y temperatura.

#### 3.4.2. Controlador módulo de comunicación BG96

Se desarrollaron varias funciones para el driver BG96, que permiten configurar el módulo, configurar la red y las más importantes para el trabajo las de conexión, publicación y desconexión al servidor MQTT.

El código 3.2 muestra dos funciones desarrolladas en el driver BG96. La función `connect_server_mqtt()` utilizada para conectarse al servidor MQTT, internamente lo que hace es mandar el comando que se ve en la línea 6 y esperar la respuesta del módulo, que puede ser un OK o ERROR. La función `publish_message()` es una

```

1 //Estructura para manejar los datos del driver
2 typedef struct
3 {
4     em_status_modem status_modem;
5     em_state_server_mqtt_conection status_mqtt_server;
6     pf_send_data send_data_device;
7     pf_reset_modem f_reset_modem;
8     st_config_parameters_mqtt self_mqtt;
9     st_config_context_tcp self_tcp;
10    st_info_product info_product;
11    uint8_t last_error;
12    char buffer_resp [100];
13    char *current_cmd;
14    em_bg96_error_handling ft_resp;
15 }st_bg96_config;
16
17 //Funcion de inicio del driver
18 em_bg96_error_handling init_driver(st_bg96_config *self ,pf_send_data
19                                     ft_send_data_device ,pf_reset_modem ft_reset_modem)
20 {
21     if (ft_send_data_device!=NULL) {
22         self->send_data_device=ft_send_data_device;
23     }
24     if (ft_reset_modem!=NULL) {
25         self->f_reset_modem=ft_reset_modem;
26     }
27     self->status_modem=OFF;
28     self->ft_resp=FT_BG96_OK;
29     self->last_error=BG96_NO_ERROR;
30     self->self_tcp.context_id=1;
31     self->self_tcp.context_type=1;
32     self->self_tcp.method_authentication=1;
33     self->self_tcp.tcp_password="";
34     self->self_tcp.tcp_username="";
35     self->self_mqtt.identifier_socket_mqtt=0;
36     self->self_mqtt.quality_service=0;
37     self->self_mqtt.port=1883;
38     self->self_mqtt.mqtt_client_id="123a56cb9";
39     self->status_mqtt_server=SERVER_MQTT_DOWN;
40     self->self_mqtt.host_name="\\"mqtt.thingsboard.cloud\"";
41     self->self_mqtt.username="";
42     self->self_mqtt.mqtt_password="";
43     self->self_tcp.tcp_apn="internet.tigo.bo";
44     return self->ft_resp;
45 }
46 //Funcion para obtener el estado del modulo
47 em_bg96_error_handling get_status_modem(st_bg96_config* self)
48 {
49     self->ft_resp=FT_BG96_OK;
50     self->current_cmd="AT\r";
51     self->ft_resp=self->send_data_device (self->current_cmd ,RS_BG96_OK ,self
52                                         ->buffer_resp ,1000);
53     if (self->ft_resp!=FT_BG96_OK)
54     {
55         self->last_error=BG96_ERROR_STATUS_MODEM;
56     }
57     return self->ft_resp;
58 }
59 //Funcion para enviar mensaje de texto
60 em_bg96_error_handling send_sms_bg96(st_bg96_config *self ,char*number ,
61                                       char*message)
62 {
63     self->ft_resp=FT_BG96_OK;

```

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 //Funcion para conectarse al servidor MQTT
2 em_bg96_error_handling connect_server_mqtt(st_bg96_config *self)
3 {
4     self->ft_resp=FT_BG96_ERROR;
5     char cmd[150]={0};
6     sprintf(cmd,"AT+QMICONN=%u,%*s,%*s,%*s\n",self->self_mqtt.
7     identifier_socket_mqtt ,self->self_mqtt.mqtt_client_id ,self->
8     self_mqtt.mqtt_username ,self->self_mqtt.mqtt_password );
9     self->ft_resp=self->send_data_device(cmd,RS_BG96_CERO ,self->
10    buffer_resp ,10000);
11    if (self->ft_resp!=FT_BG96_OK)
12    {
13        self->last_error=BG96_ERROR_CONNECT_SERVER_MQTT;
14    }
15    return self->ft_resp;
16 }
17 //Funcion para publicar un mensaje al topico configurado
18 em_bg96_error_handling publish_message(st_bg96_config *self ,char *
19 topic ,char *data)
20 {
21     self->ft_resp=FT_BG96_ERROR;
22     char cmd[50]={0};
23     char buffer_data[220]={0};
24     sprintf(buffer_data,"%*xla\n",data);
25     sprintf(cmd,"AT+QMIPUB=%u,0,0,0,%*s\n",self->self_mqtt.
26     identifier_socket_mqtt ,topic );
27     self->ft_resp=self->send_data_device(cmd,RS_BG96_SIGNAL ,self->
28     buffer_resp ,3000);
29     if (FT_BG96_OK==self->ft_resp)
30     {
31         self->ft_resp=self->send_data_device(buffer_data ,RS_BG96_CERO ,self
32             ->buffer_resp ,15000);
33         if (self->ft_resp!=FT_BG96_OK)
34         {
35             self->last_error=BG96_ERROR_PUBLISH_MESSAGE;
36         }
37     }
38     else self->last_error=BG96_ERROR_PUBLISH_MESSAGE;
39 }
40

```

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 //Funcion para enviar mensaje de texto
2 em_bg96_error_handling send_sms_bg96(st_bg96_config *self ,char*number ,
3                                       char*message)
4 {
5     self->ft_resp=FT_BG96_OK;
6     char buffer_message[256];
7     char buffer_number[20];
8     sprintf(buffer_number,"AT+CMGS=%*s\n",number);
9     sprintf(buffer_message,"%*xla\n",message);

```

**3.4. Controladores implementados**

21

```

51 char buffer_message[256];
52 char buffer_number[20];
53 sprintf(buffer_number, "AT+CMGS=%s\r\n", number);
54 sprintf(buffer_message, "%s\x1a\x1a", message);

55 self->ft_resp=self->send_data_device(buffer_number, RS_BG96_SIGNAL, self
56 ->buffer_resp,12000);
57 if (FT_BG96_OK==self->ft_resp)
58 {
59   self->ft_resp=self->send_data_device(buffer_message,RS_BG96_OK, self
60 ->buffer_resp,12000);
61   if (FT_BG96_OK!=self->ft_resp)
62   {
63     self->last_error=BG96_ERROR_SEND_SMS;
64   }
65 }
66 return self->ft_resp;
}

//Funcion para configurar el contexto de la red
em_bg96_error_handling set_parameter_context_tcp(st_bg96_config *self)
{
  self->ft_resp=FT_BG96_OK;
  char cmd[100];
  sprintf(cmd, "AT+QICSGR=%u,%u,\"%s\", \"%s\", \"%s\", \"%s\", \"%s\", self->self_tcp
  .context_id, self->self_tcp.context_type, self->self_tcp.tcp_apn, self
  ->self_tcp.tcp_username, self->self_tcp.tcp_password, self->self_tcp;
  method_authentication);
  self->ft_resp=self->send_data_device(cmd,RS_BG96_OK, self->buffer_resp
  ,3000);
  if (self->ft_resp!=FT_BG96_OK)
  {
    self->last_error=BG96_ERROR_SET_PARAMETER_CONTEXT_TCP;
  }
  return self->ft_resp;
}

//Funcion para activar el contexto pdp
em_bg96_error_handling activate_context_pdp(st_bg96_config *self)
{
  self->ft_resp=FT_BG96_OK;
  char cmd[30];
  sprintf(cmd, "AT+QIACT=%u\r\n", self->self_tcp.context_id);
  self->ft_resp=self->send_data_device(cmd,RS_BG96_OK, self->buffer_resp
  ,15000);
  if (self->ft_resp!=FT_BG96_OK)
  {
    self->last_error=BG96_ERROR_ACTIVATE_CONTEXT_PDP;
  }
  return self->ft_resp;
}

//Funcion para abrir un cliente MQTT
em_bg96_error_handling open_client_mqtt(st_bg96_config *self)
{
  self->ft_resp=FT_BG96_ERROR;
  char cmd[100];
  sprintf(cmd, "AT+QMIOHEN=%u,%s,%s\r\n", self->self_mqtt.
  identifier_socket_mqtt, self->self_mqtt.host_name, self->self_mqtt.
  port);
  self->ft_resp=self->send_data_device(cmd,RS_BG96_CERO, self->
  buffer_resp,75000);
  if (self->ft_resp!=FT_BG96_OK)
  {
    self->last_error=BG96_ERROR_OPEN_CLIENT_MQTT;
  }
}

```

**3.5. Desarrollo del hardware**

21

```

10 self->ft_resp=self->send_data_device(buffer_number,RS_BG96_SIGNAL, self
11 ->buffer_resp,12000);
12 if (FT_BG96_OK==self->ft_resp)
13 {
14   self->ft_resp=self->send_data_device(buffer_message,RS_BG96_OK, self
15 ->buffer_resp,12000);
16   if (FT_BG96_OK!=self->ft_resp)
17   {
18     self->last_error=BG96_ERROR_SEND_SMS;
19   }
20 }
21 return self->ft_resp;
}

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**

Al ser un prototipo lo que se hizo fue desarrollar un tarjeta donde se puedan ensamblar y conectar los módulos utilizados en el trabajo: módulo de comunicación celular, 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 .

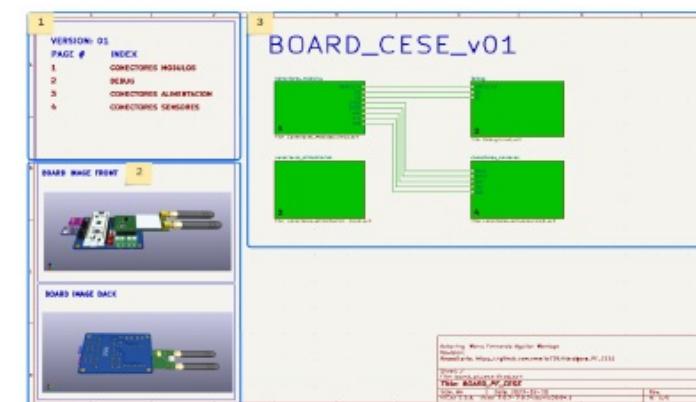


FIGURA 3.10. Esquemático página raíz.

```

134     return self->ft_resp;
135 }
136 //Funcion para conectarse al servidor MQTT
137 em_bg96_error_handling connect_server_mqtt(st_bg96_config *self)
138 {
139     self->ft_resp=FT_BG96_ERROR;
140     char cmd[150]={0};
141     sprintf(cmd, "AT+QMICONN=%u,%s,%s,%s,%s\r", self->self_mqtt.
142         identifier_socket_mqtt, self->self_mqtt.mqtt_client_id, self->
143         self_mqtt.username, self->self_mqtt.password);
144     self->ft_resp=self->send_data_device(cmd, RS_BG96_CERO, self->
145         buffer_resp,10000);
146     if (self->ft_resp!=FT_BG96_OK)
147     {
148         self->last_error=BG96_ERROR_CONNECT_SERVER_MQTT;
149     }
150     return self->ft_resp;
151 }
152 //Funcion para publicar un mensaje al topico configurado
153 em_bg96_error_handling publish_message(st_bg96_config *self, char *topic,
154     char *data)
155 {
156     self->ft_resp=FT_BG96_ERROR;
157     char cmd[50]={0};
158     char buffer_data[220]={0};
159     sprintf(buffer_data, "%x1a\r",data);
160     sprintf(cmd, "AT+QMIPUB=%u,0,0,%s\r", self->self_mqtt.
161         identifier_socket_mqtt,topic);
162     self->ft_resp=self->send_data_device(cmd, RS_BG96_SIGNAL, self->
163         buffer_resp,.3000);
164     if (FT_BG96_OK==self->ft_resp)
165     {
166         self->ft_resp=self->send_data_device(buffer_data ,RS_BG96_CERO, self->
167             buffer_resp,15000);
168         if (self->ft_resp!=FT_BG96_OK)
169         {
170             self->last_error=BG96_ERROR_PUBLISH_MESSAGE;
171         }
172     }
173     else self->last_error=BG96_ERROR_PUBLISH_MESSAGE;
174 }
175 }
```

CÓDIGO 3.2. Funciones principales del driver del sensor AHT10.

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.

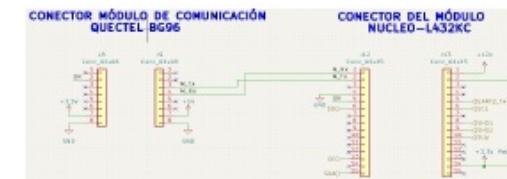


FIGURA 3.11. Conector módulo BG96 y NUCLEO-L432KC.

Se colocaron dos leds para debug, uno para señalizar si se logró conectar al servidor MQTT y el otro led para señalizar si el sistema entra en un estado de error. Se colocó un conector para una puerto serial por donde el módulo manda la secuencias de comandos que manda y recibe del módulo de comunicación. En la figura 3.12 se puede ver el circuito.



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.

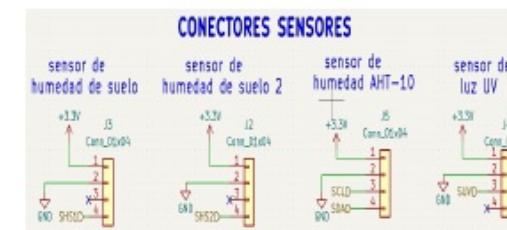


FIGURA 3.13. Esquemático conectores sensores.

Se colocaron 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.

### 3.5. Desarrollo del hardware

Para diseño del hardware se utilizó KICAD 6.0 herramienta de diseño aprendida en el transcurso de la especialización.

#### 3.5.1. Esquemático

Al ser un prototipo lo que se hizo fue desarrollar un tarjeta donde se puedan ensamblar y conectar nuestros módulos utilizados en el proyecto: módulo de comunicación celular, tarjeta de desarrollo con el microcontrolador y los módulos sensores.

En la figura 3.10 se muestra la página raíz del esquemático del proyecto, en la parte izquierda superior se ven la número de hojas en las que se dividió el esquemático, en la parte izquierda inferior se ve el 3d del la tarjeta desarrollada y en parte derecha se ve la conexión entre las diferentes hojas del proyecto.

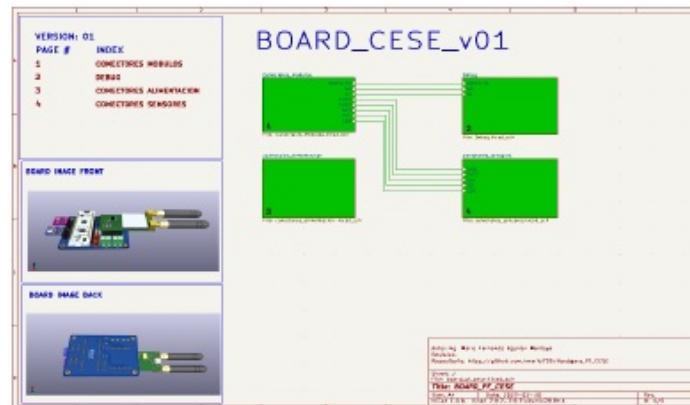


FIGURA 3.10. Esquemático página raíz.

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

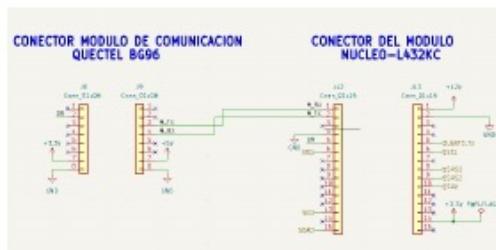


FIGURA 3.11. Esquemático conectores módulos.

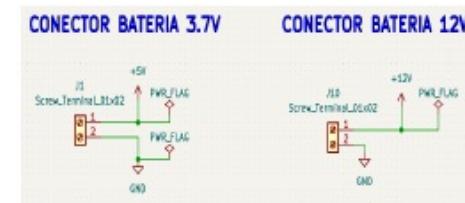
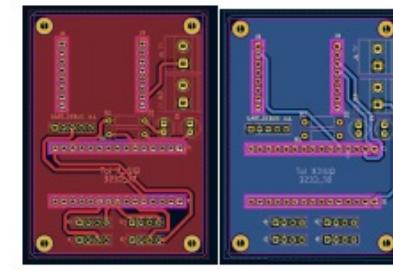


FIGURA 3.14. Esquemático conectores alimentación.

#### 3.5.2. PCB del hardware

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



(A) Capa top PCB      (B) Capa bot PCB

FIGURA 3.15. PCB del proyecto.

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

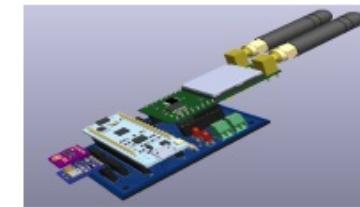


FIGURA 3.16. Modelo 3D de la tarjeta.

#### 3.5.3. Fabricación 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.

Se colocaron dos leds para debug, uno para señalizar si se logró conectar al servidor MQTT y el otro led para señalizar si el sistema entra en un estado de error, se colocó un conector para una puerto serial por donde el módulo manda la secuencias de comandos que manda y recibe del módulo de comunicación. En la figura 3.12 se puede ver el circuito.

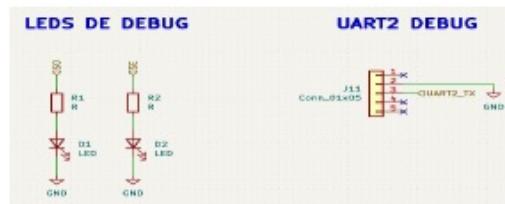


FIGURA 3.12. Esquemático interfaz de debug.

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.

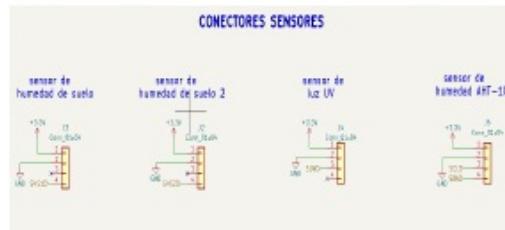


FIGURA 3.13. Esquemático conectores sensores.

Se colocaron 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.

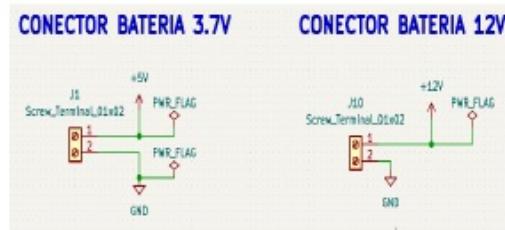


FIGURA 3.14. Esquemático conectores alimentación.



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 gráficas, tablas, etc.

#### 3.6.1. Panel principal

En la figura 3.18 se aprecia el panel principal de la interfaz gráfica. A continuación se describirán cada zona del panel principal. El panel está dividido 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 muestran los cambios que van teniendo los valores de las variables medidas por los sensores con respecto al tiempo.
- Zona 3: mapa con la ubicación de los nodos sensores implementados.

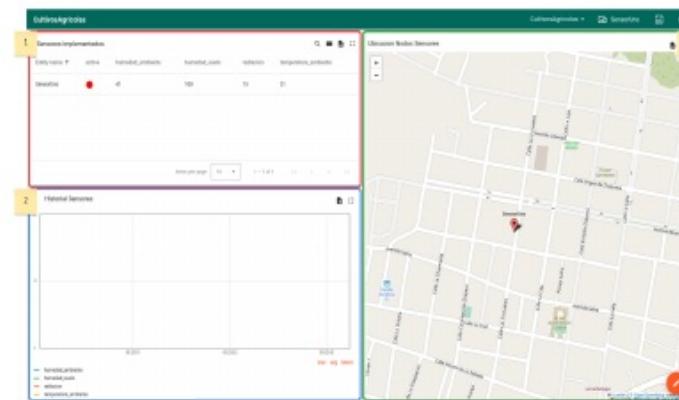


FIGURA 3.18. Panel principal de la interfaz gráfica.

### 3.5. Desarrollo del hardware

25

#### 3.5.2. PCB del hardware

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

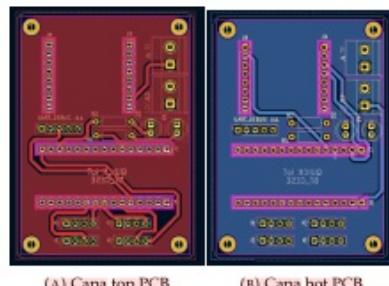


FIGURA 3.15. PCB del proyecto.

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

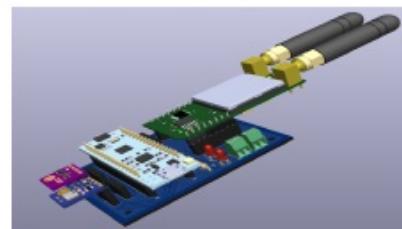


FIGURA 3.16. 3D del tarjeta desarrollada.

#### 3.5.3. Fabricacion circuito impreso

Una vez completado y validado el diseño, se generaron los archivos de fabricación para su posterior producción por la empresa JLCPCB figura 3.17.

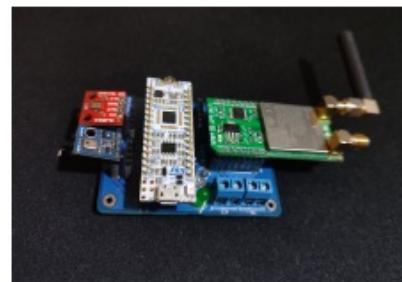


FIGURA 3.17. PCB ensamblado.

### 3.6. Paneles de visualización

25

#### 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.

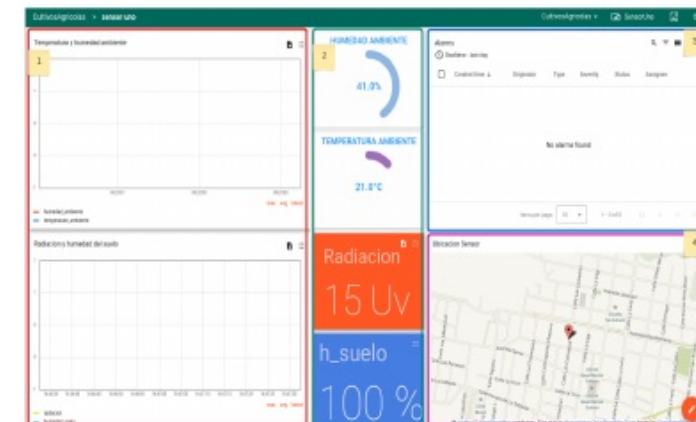


FIGURA 3.19. Panel nodo sensor.

### 3.6. Paneles de visualización

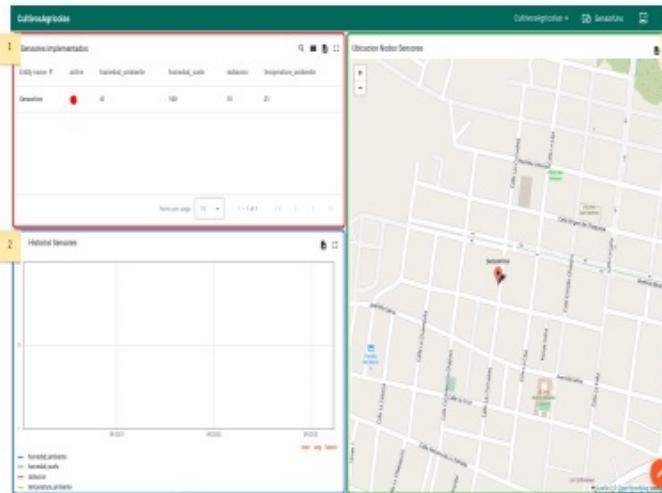
La herramienta de visualización de ThingsBoard es muy versátil para el armado de paneles de visualización escalables y altamente configurables.

Se armaron dos paneles de visualización un panel principal que muestra todos los nodos sensores implementados y un panel secundario o panel de nodo sensor que muestra de forma detallada las variables monitoreadas por el nodo sensor.

#### 3.6.1. Panel principal

En la figura 3.18 se aprecia el panel principal de la interfaz gráfica. A continuación se describirán cada zona del panel principal. El panel está dividido en las siguientes zonas:

- Zona 1: Listado de todos los nodos sensores implementados y activos, asiendo click en el dispositivo podemos navegar al panel de visualización del nodo donde podemos ver con mas detalle las variables medidas.
- Zona 2: Gráficas que muestra los cambios que van teniendo los valores de las variables medidas por los sensores con respecto al tiempo.
- Zona 3: Mapa con la ubicación de los nodo sensores implementados.



### 3.6. Paneles de visualización

27

#### 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.

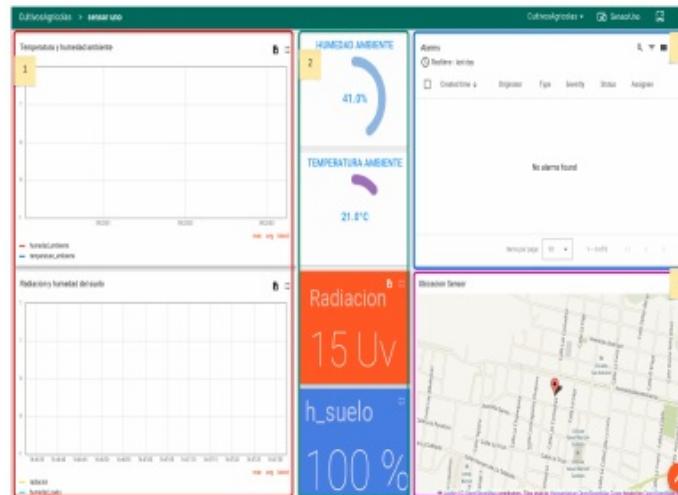


FIGURA 3.19. Panel nodo sensor.

27

## Capítulo 4

# Conclusiones

#### 4.1. Conclusiones generales

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

Algunas preguntas que pueden servir para completar este capítulo:

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puede seguir la planificación original (cronograma incluido)?
- ¿Se manifestó alguno de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

#### 4.2. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.

## Capítulo 4

### Ensayos y resultados

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

## 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] connectamericas. *Descripción de la empresa,Definición de UBIDOTS*. Visitado: 2023-05-27. URL: <https://connectamericas.com/es/company/ubidots>.
- [7] STMicroelectronics. *Especificación de Producto,STM32 Nucleo-32 boards*. Ultima actualización 2019-06-05 V8.0. URL: [https://www.st.com/resource/en/data\\_brief/nucleo-l432kc.pdf](https://www.st.com/resource/en/data_brief/nucleo-l432kc.pdf).
- [8] MikroElectronic. *Especificación de Producto,LTE IOT 2 CLICK*. Visitado: 2023-05-27. URL: <https://www.mikroe.com/lte-iot-2-click>.
- [9] SSDIELECT ELECTRONICA SAS. *Especificación de Producto,AHT10 SENSOR DE TEMPERATURA Y HUMEDAD I2C*. Visitado: 2023-05-27. URL: <https://ssdiselect.com/temperatura/3885-aht10.html>.
- [10] naylampmechatronics. *Especificación de Producto,MÓDULO SENSOR DE LUZ ULTRAVIOLETA (UV) ML8511*. Visitado: 2023-05-27. URL: <https://ssdiselect.com/temperatura/3885-aht10.html>.
- [11] PANAMAHITEK. *Especificación 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] Ceedling. *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](https://www.rohde-schwarz.com/es/productos/test-y-medida/essentials-test-equipment/digital-oscilloscopes/que-es-uart_254524.html).
- [16] sparkfun. *Definición 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] ThingsBoard. *Descripción de la plataforma IoT, Definición de ThingsBoard.*  
Visitado: 2023-05-27. URL: <https://thingsboard.io/>.