Índice general

v

Índice general

VI

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].

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].

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.

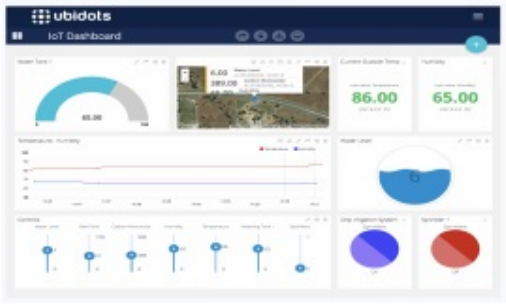

FIGURA 2.8. Ejemplo interfaz gráfica Ubidots[8].

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[9].

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

[9]Imagen tomada de la página https://thingsboard.io/smart-farming/

10 Capítulo 2. Introducción específica

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.

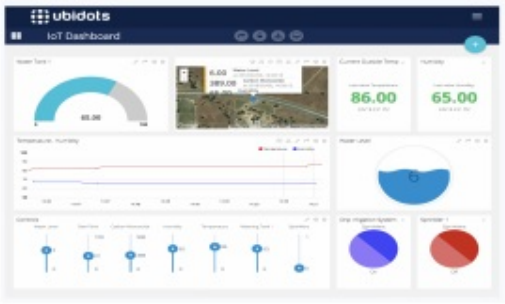

FIGURA 2.8. Ejemplo interfaz gráfica Ubidots[8].

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 ThingsBoard[9].

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

[9]Imagen tomada de la página https://thingsboard.io/smart-farming/

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(Mensaje de texto) de las alarmas activas del sistema. Finalmente, después de monitorear las alarmas la tarea manda un evento

14 Capítulo 3. Diseño e implementación

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.

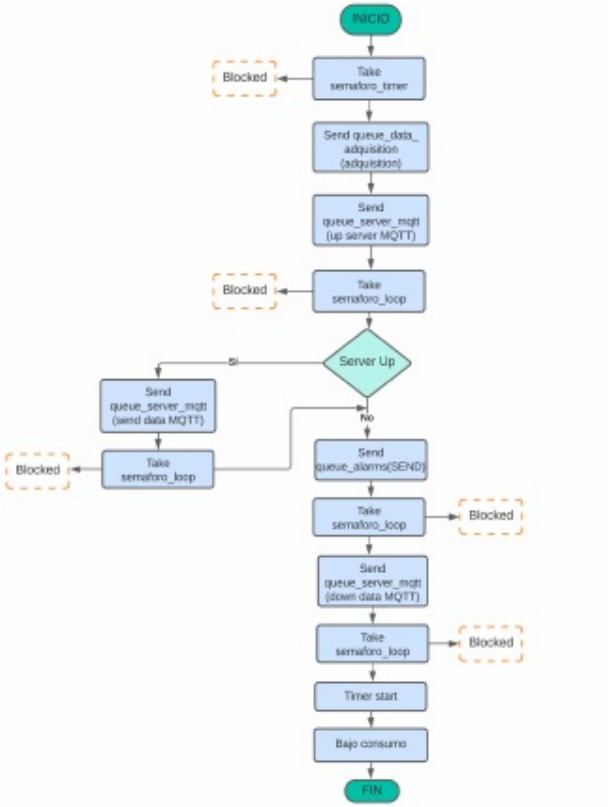

FIGURA 3.4. Diagrama de flujo de la tarea loop.

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.

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.

3.3. Desarrollo del firmware 15

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.

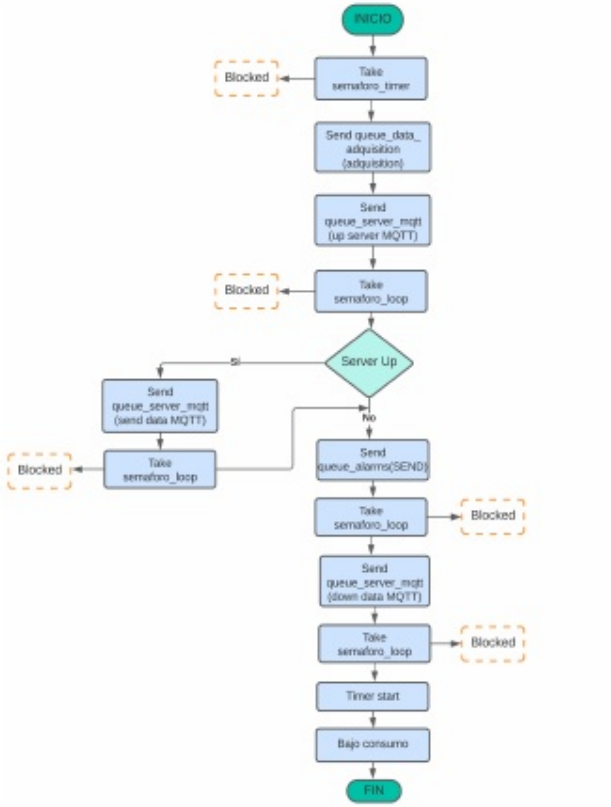

FIGURA 3.4. Diagrama de flujo de la tarea loop.

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.

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 si el valor del sensor de humedad es menor a 10. Aumentando en uno la variable que cuenta las alarmas activas si esto fuera verdadero. 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.

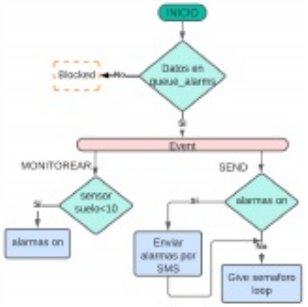

FIGURA 3.6. Diagrama de flujo de la tarea manejador de alarmas.

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 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.

16 Capítulo 3. Diseño e implementación

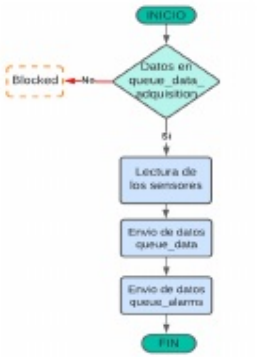

FIGURA 3.5. Diagrama de flujo tarea de adquisición de datos.

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.

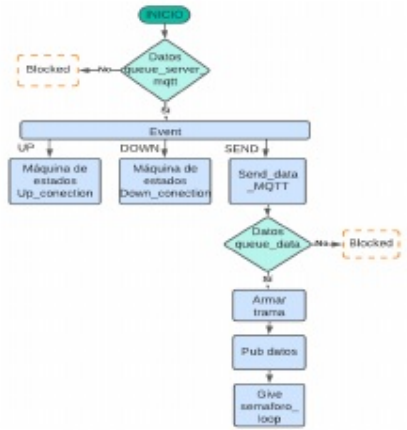

FIGURA 3.7. Diagrama de flujo de la tarea conexion server MQTT.

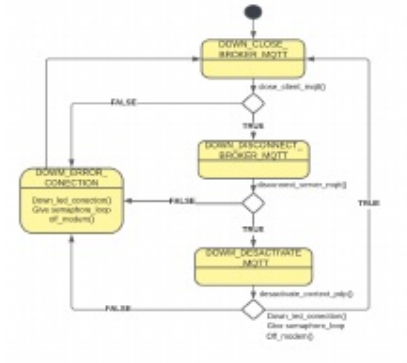

FIGURA 3.8. Máquina de estados down servidor.

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.

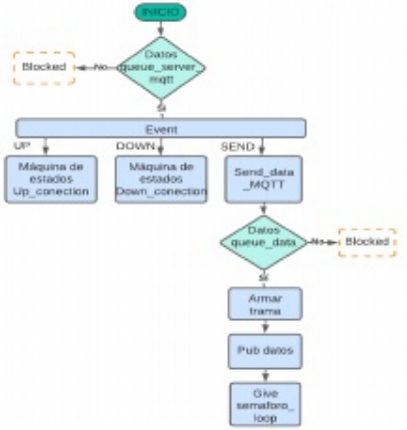

FIGURA 3.7. Diagrama de flujo de la tarea conexion server MQTT.

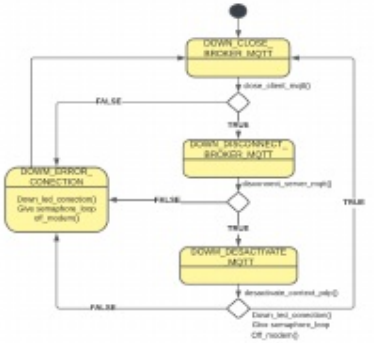

FIGURA 3.8. Máquina de estados down servidor.

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.

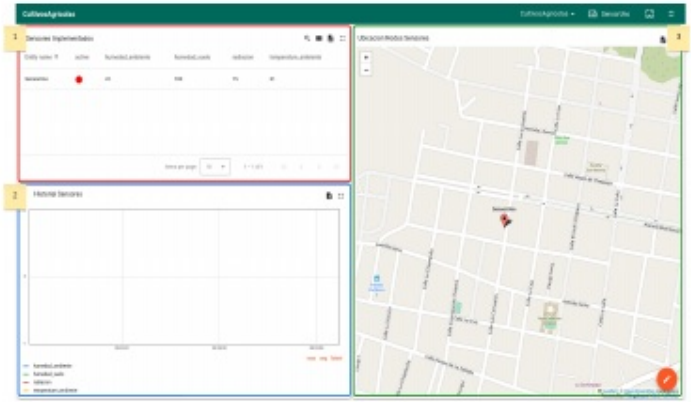

FIGURA 3.18. Panel principal de la interfaz gráfica.

24 Capítulo 3. Diseño e implementación

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.

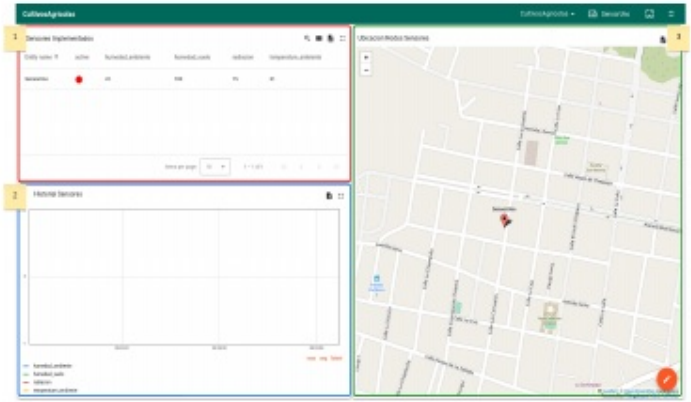

FIGURA 3.18. Panel principal de la interfaz gráfica.

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. Esto implica que se escribieron pruebas unitarias para los drivers. Se utilizó ceedling como herramienta de desarrollo de pruebas automáticas.

En el 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, líneas 6,11 y 15. Se prueban dos casos: cuando la lectura de los registros de estado del sensor es 0 y cuando es 255.

```
1  //Prueba de funcion para obtener el estado del sensor AHT10
2  void test_estado_del_sensor(void)
3  {
4    uint8_t buffer[1]={0};
5    uint8_t data=0;
6    read_I2C_STM32L432_port_ExpectAndReturn(AHT10_ADDRESS_SLAVE, buffer, 1,
       AHT10_OK);
7    read_I2C_STM32L432_port_ReturnThruPtr_buffer(&data);
8    TEST_ASSERT_EQUAL(SENSOR_IDLE, aht10_get_status(&aht10config));
9
10   data=255;
11   read_I2C_STM32L432_port_ExpectAndReturn(AHT10_ADDRESS_SLAVE, buffer, 1,
       AHT10_OK);
12   read_I2C_STM32L432_port_ReturnThruPtr_buffer(&data);
13   TEST_ASSERT_EQUAL(SENSOR_BUSY, aht10_get_status(&aht10config));
14
15   read_I2C_STM32L432_port_ExpectAndReturn(AHT10_ADDRESS_SLAVE, buffer, 1,
       AHT10_ERROR);
16   TEST_ASSERT_EQUAL(SENSOR_BUSY, aht10_get_status(&aht10config));
17 }
```

CÓDIGO 4.1. Tests del driver del sensor AHT10.

En el código **??** 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  //Prueba de la funcion para mandar sms
2  void test_send_sms_bg96(void)
3  {
```

27

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[1]. Esto implica que se escribieron pruebas unitarias para los drivers. Se utilizó Ceedling como herramienta de desarrollo de pruebas automáticas.

En el 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, líneas 6,11 y 15. Se prueban dos casos: cuando la lectura de los registros de estado del sensor es 0 y cuando es 255.

```
1  //Prueba de funcion para obtener el estado del sensor AHT10
2  void test_estado_del_sensor(void)
3  {
4    uint8_t buffer[1]={0};
5    uint8_t data=0;
6    read_I2C_STM32L432_port_ExpectAndReturn(AHT10_ADDRESS_SLAVE, buffer, 1,
       AHT10_OK);
7    read_I2C_STM32L432_port_ReturnThruPtr_buffer(&data);
8    TEST_ASSERT_EQUAL(SENSOR_IDLE, aht10_get_status(&aht10config));
9
10   data=255;
11   read_I2C_STM32L432_port_ExpectAndReturn(AHT10_ADDRESS_SLAVE, buffer, 1,
       AHT10_OK);
12   read_I2C_STM32L432_port_ReturnThruPtr_buffer(&data);
13   TEST_ASSERT_EQUAL(SENSOR_BUSY, aht10_get_status(&aht10config));
14
15   read_I2C_STM32L432_port_ExpectAndReturn(AHT10_ADDRESS_SLAVE, buffer, 1,
       AHT10_ERROR);
16   TEST_ASSERT_EQUAL(SENSOR_BUSY, aht10_get_status(&aht10config));
17 }
```

CÓDIGO 4.1. Tests del driver del sensor AHT10.

En el 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]Definición de la abreviatura https://www.browserstack.com/guide/what-is-test-driven-development

```
    char buffer_resp[30]={0};
    send_data_ExpectAndReturn("AT+CMGS=\"72950576\"\r",RS_BG96_SIGNAL,
      buffer_resp,12000,FT_BG96_OK);
    send_data_ExpectAndReturn("HOLA\x1a\r",RS_BG96_OK,buffer_resp,12000,
      FT_BG96_OK);
    TEST_ASSERT_EQUAL(FT_BG96_OK,send_sms_bg96(&config_module,"72950576","
      HOLA"));

    send_data_ExpectAndReturn("AT+CMGS=\"72950576\"\r",RS_BG96_SIGNAL,
      buffer_resp,12000,FT_BG96_ERROR);
    TEST_ASSERT_EQUAL(FT_BG96_ERROR,send_sms_bg96(&config_module,"72950576
      ","HOLA"));

    send_data_ExpectAndReturn("AT+CMGS=\"72950576\"\r",RS_BG96_SIGNAL,
      buffer_resp,12000,FT_BG96_OK);
    send_data_ExpectAndReturn("HOLA\x1a\r",RS_BG96_OK,buffer_resp,12000,
      FT_BG96_ERROR);
    TEST_ASSERT_EQUAL(FT_BG96_ERROR,send_sms_bg96(&config_module,"72950576
      ","HOLA"));
}
```

CÓDIGO 4.2. Tests del driver del modulo bg96.

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

En la figura 4.1 se puede observar el informe de cobertura del driver aht10, donde se puede apreciar que las pruebas ejecutan el 100 % de las líneas de código escritas y explora el 100 % de las combinaciones en los saltos de condicionales.

En la figura 4.2 también se observa el informe de cobertura del driver de bg96, con 98.4 % de líneas ejecutadas y explora más del 98.3 % de combinaciones posibles.

FIGURA 4.1. Informe de cobertura driver AHT10.

FIGURA 4.2. Informe de cobertura driver BG96.


```
//Prueba de la funcion para mandar sms
void test_send_sms_bg96(void)
{
    char buffer_resp[30]={0};
    send_data_ExpectAndReturn("AT+CMGS=\"72950576\"\r",RS_BG96_SIGNAL,
      buffer_resp,12000,FT_BG96_OK);
    send_data_ExpectAndReturn("HOLA\x1a\r",RS_BG96_OK,buffer_resp,12000,
      FT_BG96_OK);
    TEST_ASSERT_EQUAL(FT_BG96_OK,send_sms_bg96(&config_module,"72950576","
      HOLA"));

    send_data_ExpectAndReturn("AT+CMGS=\"72950576\"\r",RS_BG96_SIGNAL,
      buffer_resp,12000,FT_BG96_ERROR);
    TEST_ASSERT_EQUAL(FT_BG96_ERROR,send_sms_bg96(&config_module,"72950576
      ","HOLA"));

    send_data_ExpectAndReturn("AT+CMGS=\"72950576\"\r",RS_BG96_SIGNAL,
      buffer_resp,12000,FT_BG96_OK);
    send_data_ExpectAndReturn("HOLA\x1a\r",RS_BG96_OK,buffer_resp,12000,
      FT_BG96_ERROR);
    TEST_ASSERT_EQUAL(FT_BG96_ERROR,send_sms_bg96(&config_module,"72950576
      ","HOLA"));
}
```

CÓDIGO 4.2. Tests del driver del módulo BG96.

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

La figura 4.1 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 4.2, 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.

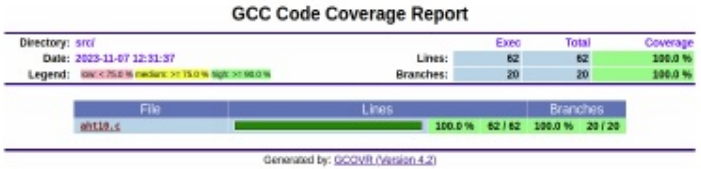

FIGURA 4.1. Informe de cobertura driver AHT10.

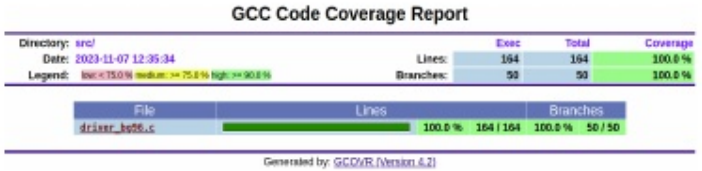

FIGURA 4.2. Informe de cobertura driver BG96.

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 ThingsBoard. 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 4.3.

FIGURA 4.3. Envio de datos por el cliente MQTT de mosquitto.

Donde:

- -h dirección del broker
- -t tópico
- -u token
- -m mensaje en formato json

Al ejecutar el comando de la figura 4.3, 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 envio los datos y entrar a la pestaña de última telemetría. En la figura 4.4 podemos ver que los datos enviados llegan correctamente.

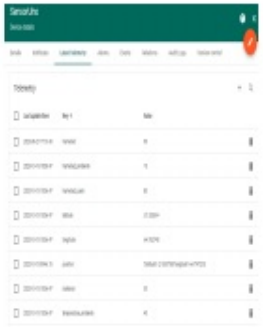

FIGURA 4.4. Recepción de datos en el broker MQTT.

4.2. Pruebas de la plataforma IoT 29

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 ThingsBoard. 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 4.3.

FIGURA 4.3. Envio de datos por el cliente MQTT de mosquitto.

Donde:

- -h dirección del broker
- -t tópico
- -u token
- -m mensaje en formato json

Al ejecutar el comando de la figura 4.3, 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 envio los datos y entrar a la pestaña de última telemetría. En la figura 4.4 se puede ver que los datos enviados llegan correctamente.

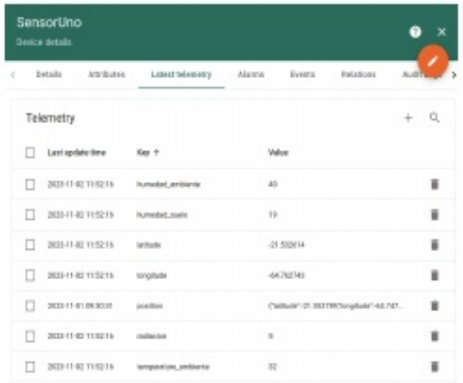

FIGURA 4.4. Recepción de datos en el broker MQTT.

4.2.2. 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 4.5 podemos ver la notificación de una alarma cuando la temperatura ambiente sobrepasa los 43°C.

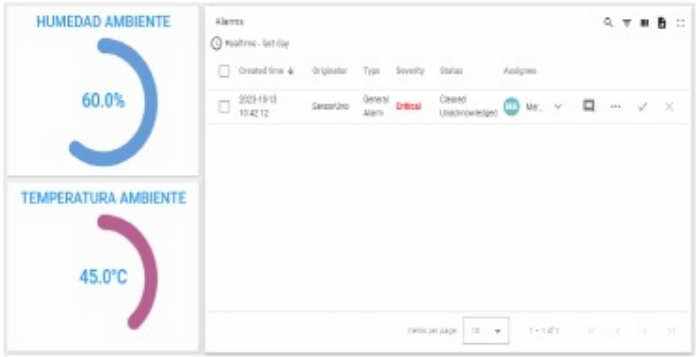

FIGURA 4.5. Tabla de alarmas activas.

4.2.3. Prueba del widget de mapa

En los paneles de visualización tenemos mapas que muestran la ubicación del dispositivo implementado. En la figura 4.6 podemos ver la ubicación del prototipo implementado para el trabajo.

FIGURA 4.6. Ubicación del nodo sensor implementado.

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

4.2.2. 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 4.5 se puede ver la notificación de una alarma cuando la temperatura ambiente sobrepasa los 43°C.

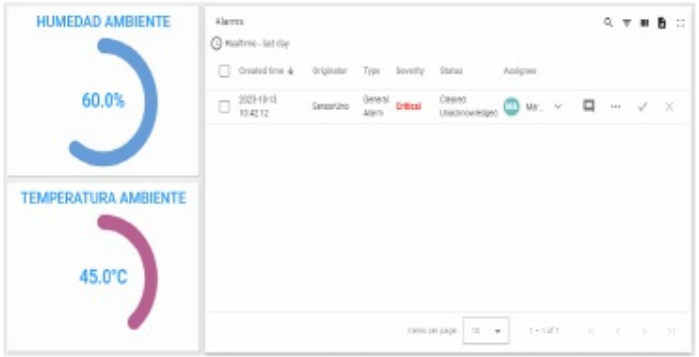

FIGURA 4.5. Tabla de alarmas activas.

4.2.3. 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.

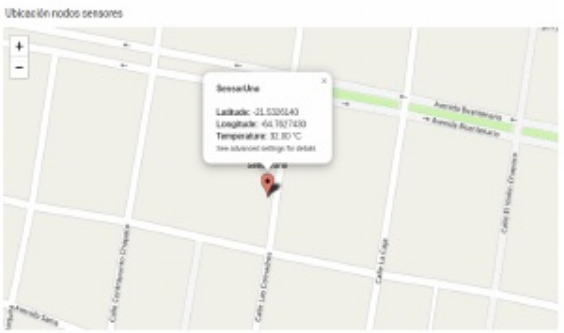

FIGURA 4.6. Ubicación del nodo sensor implementado.

4.2.4. 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 4.7.

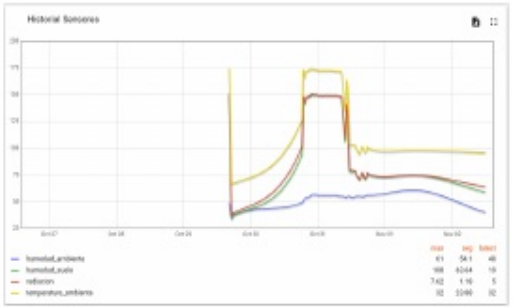

FIGURA 4.7. Persistencia de datos.

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 4.8 podemos ver la trama capturada por el analizador lógico cuando el firmware escribe en un registro del sensor.

En la figura 4.9 tenemos la trama capturada cuando se leen los registros del sensor. Se obtienen 6 bytes en los que se encuentra la información de la humedad y la temperatura.

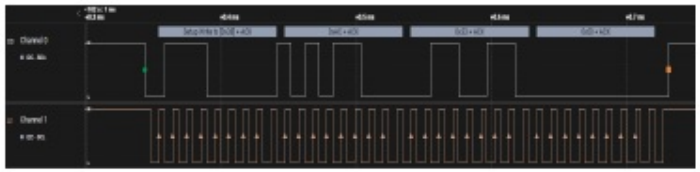

FIGURA 4.8. Trama de escritura al sensor AHT10.

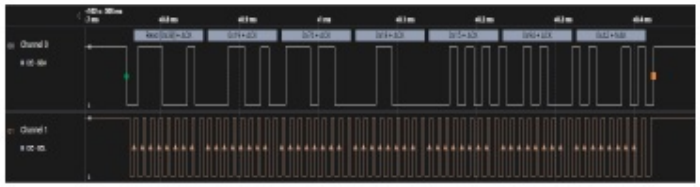

FIGURA 4.9. Trama de lectura del sensor AHT10.

4.3. Pruebas de hardware 31

estableció un rango de tiempo de 7 días. El resultado se muestra en la figura 4.7.

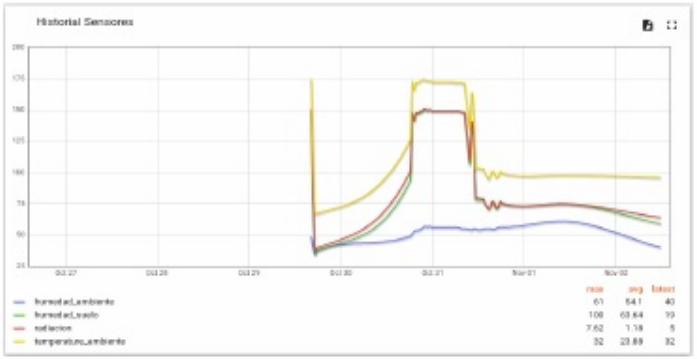

FIGURA 4.7. Persistencia de datos.

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 4.8 se puede ver la trama capturada por el analizador lógico cuando el firmware escribe en un registro del sensor.

En la figura 4.9 se puede apreciar la trama capturada cuando se leen los registros del sensor. Se obtienen 6 bytes en los que se encuentra la información de la humedad y la temperatura.

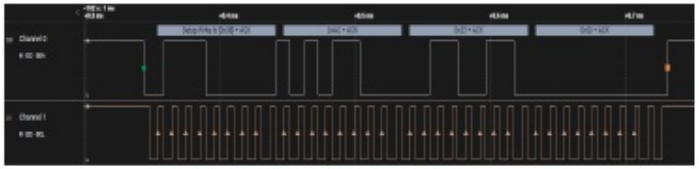

FIGURA 4.8. Trama de escritura al sensor AHT10.

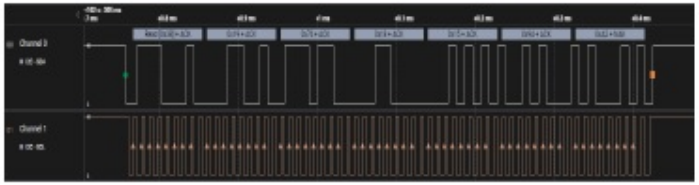

FIGURA 4.9. Trama de lectura del sensor AHT10.

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 respuestas del módulo a estos comandos por protocolo serial. En la figura 4.10 vemos 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 4.10. Envío y recepción de comandos por puerto UART.

4.4. Pruebas funcionales del sistema

Para realizar las pruebas funcionales de todo el sistema, se implementó el prototipo en un cultivo agrícola como se muestra en la figura 4.11.

FIGURA 4.11. Implementación del prototipo.

4.4.1. Pruebas de lectura de los sensores

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

Prueba uno

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

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. En la figura 4.10 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 4.10. 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 4.11.

FIGURA 4.11. Instalación del prototipo.

4.4.1. Pruebas de lectura de los sensores

Se realizaron varias pruebas al prototipo en diferentes escenarios, a continuación se desarrollaran 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 4.12 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 4.12. Panel de visualización con las lecturas obtenidas en la prueba 1.

Prueba dos

Para la prueba dos se tenian las siguiente condiciones:

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

En la figura 4.13 se muestran los resultado 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.

FIGURA 4.13. Panel de visualización con las lecturas obtenidas en la prueba 2.

- Horario de la lectura: 3 p.m

En la figura 4.12 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 4.12. 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 4.13 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.

FIGURA 4.13. Panel de visualización con las lecturas obtenidas en el caso de uso 2.

4.4.2. Prueba del envío de datos al broker MQTT

Una de las tareas más importante del firmware, es la del manejo de la comunicación con el servidor. Para verificar el funcionamiento de esta tarea, tenemos que ver la secuencia de comandos que manda el microcontrolador al módulo de comunicación. Para realizar esta prueba se conectó un convertidor UART a USB al conector de depuración que tiene nuestro dispositivo. En la figura 4.14 podemos 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.

FIGURA 4.14. Comandos para enviar datos al broker MQTT.

4.4.3. Prueba de envio de alarmas por SMS

El objetivo de esta prueba es verificar el funcionamiento de la alarma que monitorea la humedad del suelo. Cuando la humedad del suelo baja por debajo del rango permitido. El firmware manda un SMS al usuario con el mensaje de "Humedad de suelo muy baja". En la figura 4.15 vemos que se recibió un SMS de alarma cuando la humedad bajó de 10 %.

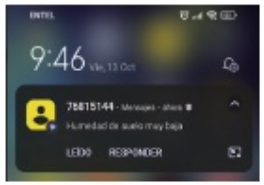

FIGURA 4.15. Recepción del SMS con el mensaje de alarma.

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 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 4.14 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.

FIGURA 4.14. Comandos para enviar datos al broker MQTT.

4.4.3. Prueba de envio 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 4.15, se observa que se ha recibido un mensaje de alarma cuando el nivel de humedad descendió por debajo del 10 %.

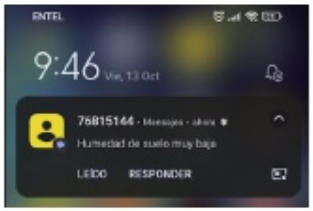

FIGURA 4.15. Recepción del SMS con el mensaje de alarma.

Capítulo 5

Conclusiones

5.1. 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 puedo seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos 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?

5.2. Próximos pasos

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

35

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 todas 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.
- Envío de alarmas del sistema por SMS al usuario.
- 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 listan los puntos que sería deseable considerar:

- 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] connectamericas. *Descripción de la empresa,Definición de UBIDOTS*. Visitado: 2023-05-27. URL: https://connectamericas.com/es/company/ubidots.

[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://ssdielect.com/temperatura/3885-aht10.html.

[10] naylampmechatronics. *Especificacion de Producto,MÓDULO SENSOR DE LUZ ULTRAVIOLETA (UV) ML8511*. Visitado: 2023-05-27. URL: https://ssdielect.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] 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.

[16] sparkfun. *Definicion del Protocolo*. Visitado: 2023-05-27. URL: hhttps://learn.sparkfun.com/tutorials/i2c/all.

[17] MQTT. *Descripción del protocolo,Introduccion a MQTT*. Visitado: 2023-05-27. URL: https://mqtt.org/.

37

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://ssdielect.com/temperatura/3885-aht10.html.

[10] naylampmechatronics. *Especificacion de Producto,MÓDULO SENSOR DE LUZ ULTRAVIOLETA (UV) ML8511*. Visitado: 2023-05-27. URL: https://ssdielect.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] 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.

[16] sparkfun. *Definicion del Protocolo*. Visitado: 2023-05-27. URL: hhttps://learn.sparkfun.com/tutorials/i2c/all.

[17] MQTT. *Descripción del protocolo,Introduccion 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/.

[18] connectamericas. *Descripción de la empresa,Definición de UBIDOTS*. Visitado: 2023-05-27. URL: https://connectamericas.com/es/company/ubidots.