.UBAfiuba
FACULTAD DE INGENIERÍA

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

# Sistema de monitoreo de cultivos agrícolas

**Autor:**
**Ing. Mario Fernando Aguilar Montoya**

Director:
Esp. Ing. Julian Bustamante Narvaez (TECREA SAS)

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

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

.UBAfiuba
FACULTAD DE INGENIERÍA

CARRERA DE ESPECIALIZACIÓN EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

Sistema de monitoreo de cultivos agrícolas

**Autor:**
**Ing. Mario Fernando Aguilar Montoya**

Director:
Esp. Ing. Julian Bustamante Narvaez (TECREA SAS)

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

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

# 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

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 [19]. 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  //Prueba de la funcion para mandar sms
2  void test_send_sms_bg96(void)
3  {
```

```
//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.

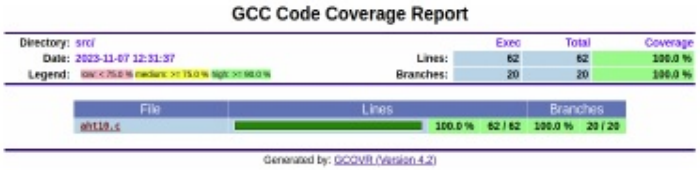


FIGURA 4.1. Informe de cobertura driver AHT10.

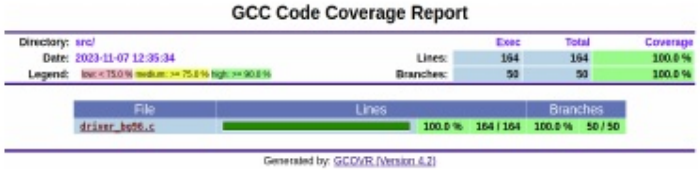


FIGURA 4.2. Informe de cobertura driver BG96.

28 Capítulo 4. Ensayos y resultados

```
  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.

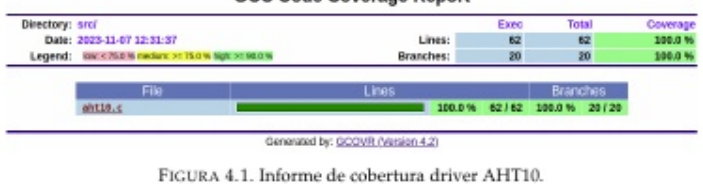


FIGURA 4.1. Informe de cobertura driver AHT10.

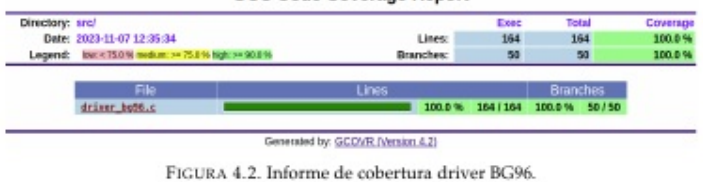


FIGURA 4.2. Informe de cobertura driver BG96.

### 4.3.2. 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

32 Capítulo 4. Ensayos y resultados

### 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 desarrollarán algunas de las pruebas realizadas.

**Caso de uso 1**

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

- Tierra con poca humedad
- Temperatura ambiente alta

# 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:

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

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

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

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

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

## 5.2. Próximos pasos

Si bien se lograron obtener los resultados esperados, a futuro se puede continuar con el desarrollo en varias medidas. A continuación se 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.

36 Capítulo 5. Conclusiones

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

## 5.2. Próximos pasos

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

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

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

38 *Bibliografía*

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