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.

Resumen

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

Agradecimientos

En primer lugar a mi familia y amigos que siempre me brindan su apoyo y confianza.
A mi director por su paciencia, guía y consejo en todo momento.
A mis compañeros y maestros de la CESE, por los aportes y enseñanzas que me dieron a los largo de la especialización.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción general

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

1.1. 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.

- 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].

FIGURA 1.1. Modulo 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. Modulo 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.

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.

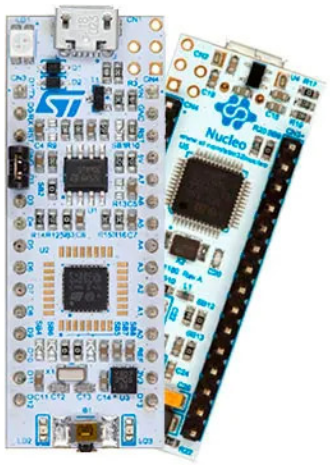


FIGURA 2.1. Plataforma de desarrollo NUCLEO-L432KC.

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.

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].

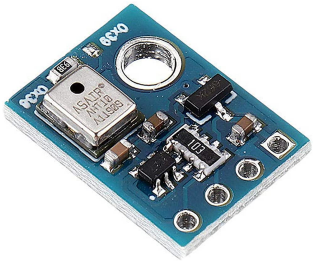


FIGURA 2.3. Sensor AHT10.

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].

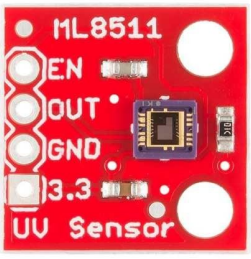


FIGURA 2.4. Módulo Sensor ML8511.

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].

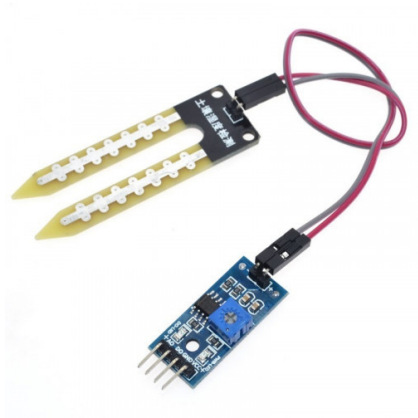


FIGURA 2.5. Módulo Sensor HL-69.

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 microprocesadores 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 microprocesadores. Distribuido libremente bajo la licencia de código abierto del MIT, FreeRTOS incluye un núcleo y un conjunto creciente de bibliotecas adecuadas para su uso en todos los sectores de la industria. FreeRTOS está diseñado con énfasis en la confiabilidad, la accesibilidad y la facilidad de uso [12]. El logo de FreeRTOS se muestra en la figura 2.6.

FIGURA 2.6. Logo FreeRTOS.

2.2.3. CEEDLING

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

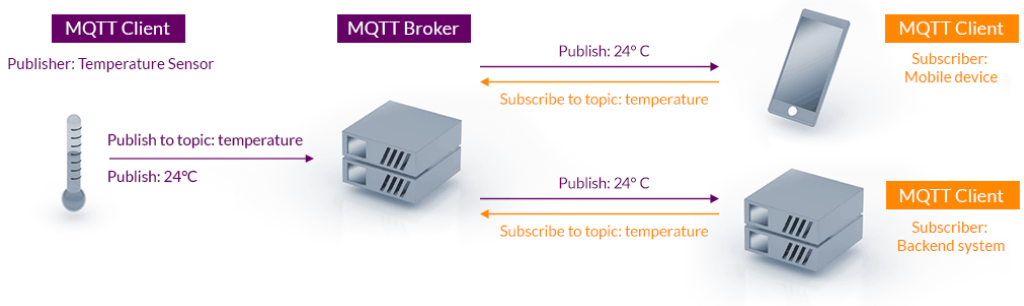


FIGURA 2.7. Arquitectura de publicación/suscripción de MQTT.

2.4. 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 grafica desarrollada en ThingsBoard.

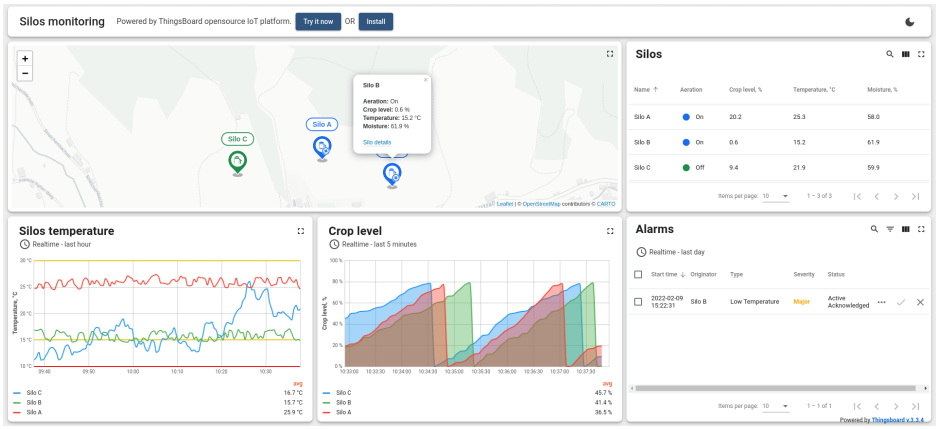


FIGURA 2.9. Ejemplo interfaz gráfica ThingsBoard.

Capítulo 3

Diseño e implementación

En este capitulo se abordara la descripcion de la arquitectura general del sistema, arquitectura del firmware, detalles de los controladores desarrollados para el modulo de comunicaicon y los sensores, desarrollo del hardware y la configuracion 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 implementado donde se describe la arquitectura IoT la cual consta de tres capas: percepción, red y aplicación.

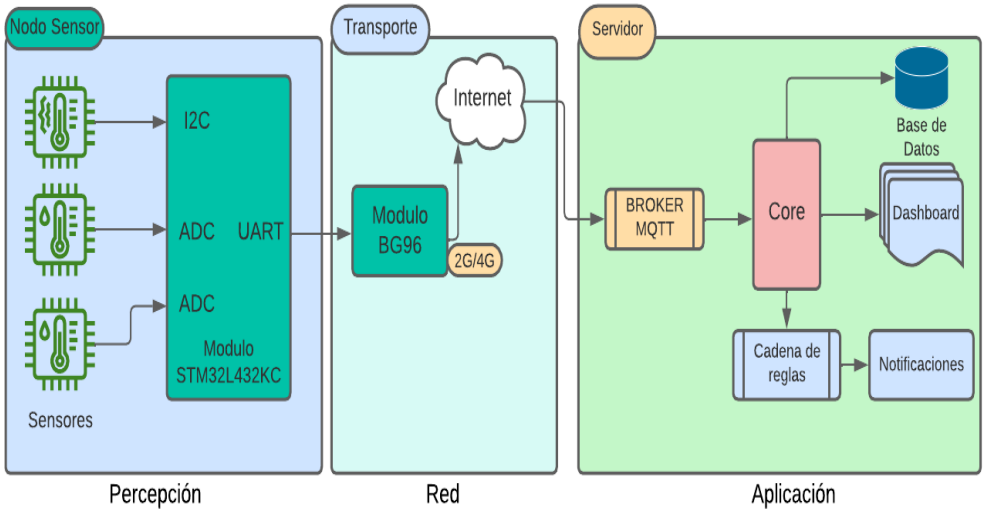


FIGURA 3.1. Diagrama general del sistema IoT.

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

- Capa de percepción. En la capa de percepción, los nodos sensores son los encargados de medir variables ambientales, hacer un prepocesamiento y enviarlas a la capa de red. Para su desarrollo se utilizo la placa STM32L432KC que contiene el firmware del sistema,tambien consta de un sensor de humedad y temperatrua ambiente AHT10 que se comunica con la placa de desarrollo mediante el protocolo I2C, sensor de humedad de suelo HL-69

y el sensor de luz UV ML-18 que se comunicacion con el modulo mediante entradas analogicas.

- Capa de red. En cuanto a la red, se utilizo un modulo quectel BG96 que puede conectarse a la red 2G, 4G, NB-IoT automaticamente dependiendo del nivel de la red en el lugar de la implementacion del modulo sensor, se comunica con el microcontrolador por comandos AT por puerto UART.
- Capa de aplicación. En la capa de aplicacion, se utilizo 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 grafica para la visualizacion de los datos y nos permite gestionar la alarmas del sistema.

3.2. Arquitectura de firmware

El desarrollo del firmware fue la tarea mas compleja del proyecto debido a que uno de los objetivos fue lograr El firmware fue estructurado en capas como se muestra en la figura 3.2 se dividio el firmware en capas para facilitar el desarrollo y reducir la complejidad del codigo escritolo separo en capas

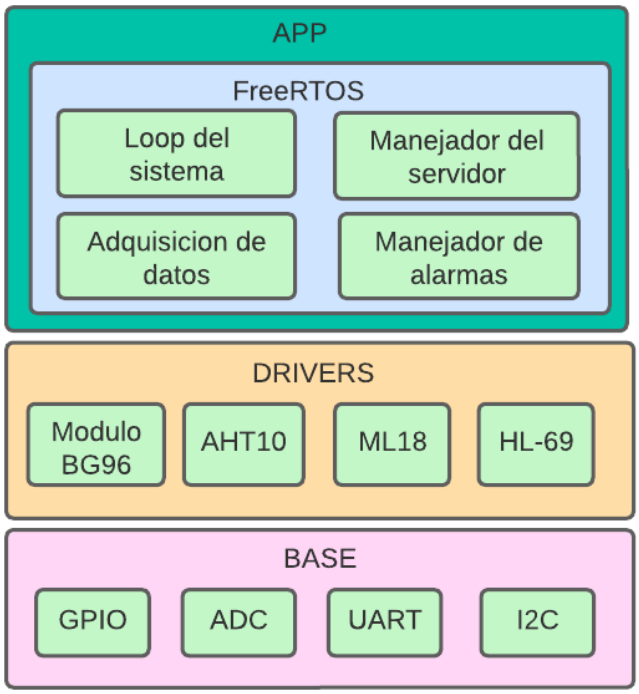


FIGURA 3.2. Capas del firmware.

La capa Base es la mas baja del sistema y esta compuesta por la libreria HAL de STM, proporciona a las capas superiores la capacidad de interactuar con los perifericos del microcontrolador STM32L432KC atraves de funciones en lenguaje C

- GPIO: La API proporciona funciones para definir el estado de los pines del microcontrolador.Fueron utilizadas por la capa de APP para el control de

los leds de debug del sistema y para el encendido y reset del modulo de comunicacion. Las funciones de esta libreria fue utilizada por la capa de APP para gestionar las entradas y salidas del microcontrolador.

- ADC: Proporcionan funciones para la configuracion, lectura y escritura de los pines de microcontroladora senales analogicas. Se utilizaron estas funciones para hacer la lectura de los sensores de humedad de suelo y el sensor de luz UV.
- UART: Brinda funciones para la lectura y estritura por el puerto UART del microcontrolador.El firmware utiliza estas funciones para la comunicacion con el modulo 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.Se utilizaron estas funciones para la lectura de los datos del sensor de AHT10.

La capa DRIVERS esta compuesta por los modulos que se desarrollaron para hardware externo al microcontrolador que permiten al microcontrolador enteractuar con hardware externo.Se desarrollaron drivers para el modulo de comunicacion BG96,AHT10,ML18 y HL69

- Modulo BG96:Se desarrollo el driver para el modulo BG96, para establecer la comunicacion de este elemento con el microcontrolador a traves del puerto UART.Las funciones mas importantes que proporcionan son
  - Estado del modulo.
  - Descripcion del modulo.
  - Configuracion APN de la red.
  - Conexion TCP.
  - Conexion a broker MQTT.
- AHT10: Se desarrollo utilizando la hoja de datos del sensor, proporciona las funciones mas importante inicializacion y lectura de los valores obtenidos por el sensor.
- ML18: Se escribio una funcion que permite convertir los datos obtenidos de forma analogica a valores significativos de humedad.

La capa de APP es la de mayor nivel gerarquico.Se la desarrollo sobre freeRTOS que nos permite hacer un codigo mas escalable. Se implementaron cuatro tareas

- Loop del sistema: Esta tarea es la que nos da la secuencialidad del sistema.
- Conexion servidor: Se encarga de manejar la conexion a la red y al broker MQTT.
- Adquisicion de datos: Se encarga de hacer la lectura de los sensores.
- Manejador de eventos: Esta tarea se encarga de manejar todos lo eventos del sistema.

3.3. Desarrollo del firmware

Para el desarrollo del firmware se utilizo el IDE oficial de STMicroelectronic(STM32CubeIDE).

El firmware fue desarrollado sobre freeRTOS, se utilizaron algunas de sus funcionanlidades como queue, semaforos, tareas, interrupciones.

El firmware tiene una etapa de inicializacion en la figura 3.3 podemos ver la secuancia de inicializacion del sistema.

Luego el firmware da el control del sistema al sistema operativo y permanece ejecutando de forma continua las tareas creadas en la etapa de inicializacion del sistema

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

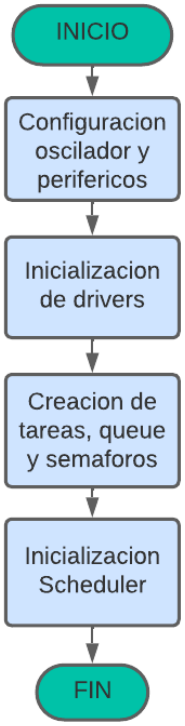


FIGURA 3.3. Diagrama de flujo inicio firmware.

Descripcion de los bloques del diagrama.

Lo primero que el firmware realiza es la configuracion del hardware del microcontrolador, luego se inicializan los drivers del modulo de comunicacion celular BG96 y el driver del sensor de humedad AHT10, ejecutan las funciones de inicialization de los sensores y del modulo de comunicacion celular,posteriormente se crean las tareas, queues y los semaforos del sistema y finalmente se da el contro del sistema al scheduler del sistema operativo.

Para el control del sistema se crearon cuatro tareas sobre freeRTOS, que se comunican y sincronizan atraves de colas y semaforos en la figura la comunicacion entre tareas.

A continuacion se desarrollara a implementacion de cada tarea del sistema.

La secuencialidad del sistema es manejado por la tarea de loop que nos permite tener secuencia repetitiva, la figura muestra el diagrma de flujo de la tarea loop La tarea comienza creando variables que se utilizaran localmente se inicia el timer del sistema, luego la tarea se blokea en el un semaforo esperando ser deblokeado por el un give que sera dado en el momento de la ejecucion del handler de la interrupcion del timer, luego se envia datos por las colas de adquisicion de datos y server mqtt para levantar el servidor y la tarea se vuelve a bloquear en el semaforos, cuando se desbloquea pregunta si se logro levantar el servidor si se logro se manda un mensaje por la cola de server mqtt con el evento de enviar datos y se bloquea nuevamente pero si no se logro levantar el servidor no se enviaran datos, luego se envia un evento a la cola de alarmas se bloquea nuevamente esperando que envien las alarmas, cuando se desbloquea el semaforo se manda un evento a la cola de server mqtt para desconectar el servidor y se bloquea la tarea en el semanforo esperando la desconexion finalmente se inicia nuevamente el timer y mandamos al sistema a modo de bajo consumo.

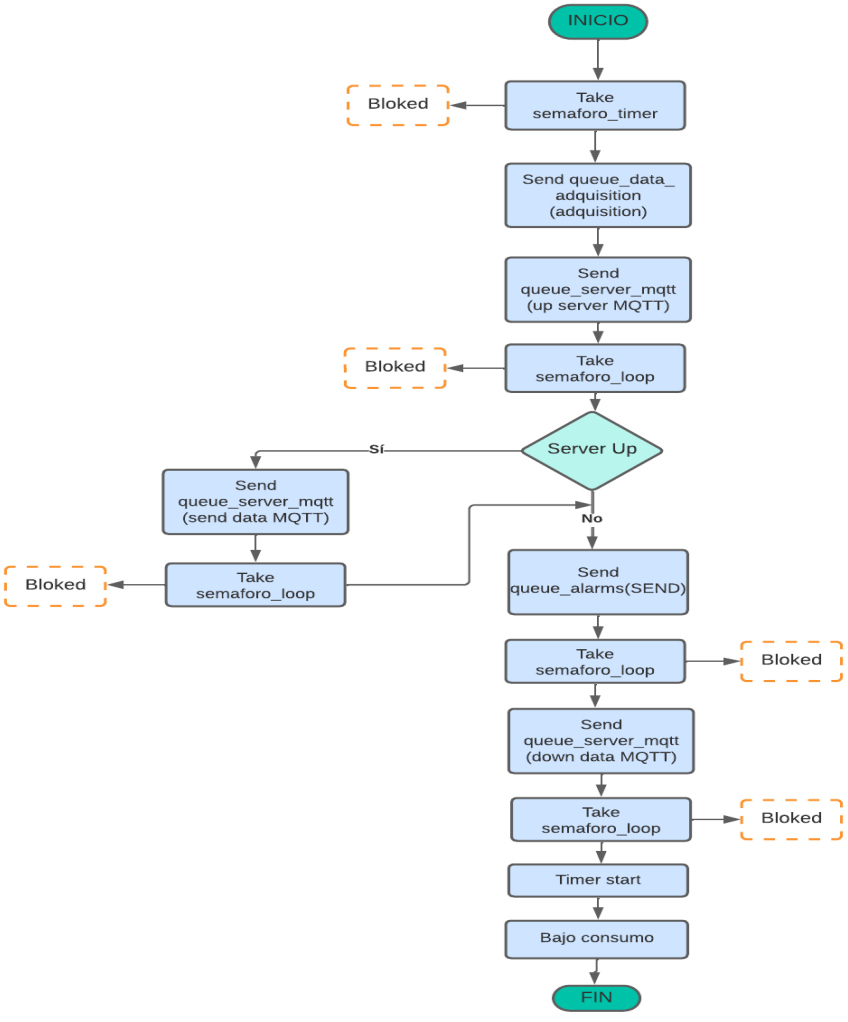


FIGURA 3.4. Diagrama de flujo tarea loop.

La figura 3.5 muestra la secuencia repetiva que realiza la tarea de adquisicion de datos, la tarea inicia creando algunas variables locales que se utilizara en la tarea, luego entra al bucle infinito donde lo que promero que hace es ver si hay algun dato en la cola de adquisicion de datos si no hay se bloquea la tarea pero si hay realiza la lectura de todos los sensores, con los datos recolectados los envia a las colas de data y alarmas y termina el ciclo.

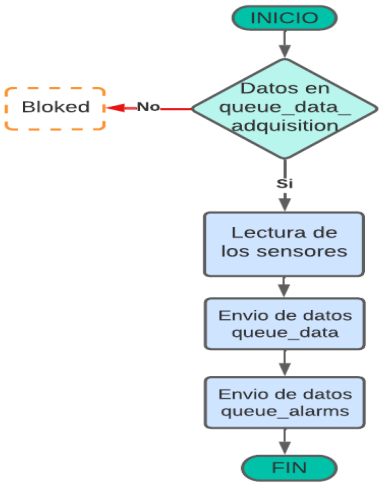


FIGURA 3.5. Diagrama de flujo tarea de adquisicion de datos.

El manejo de las alarmas se realiza atraves de la tarea 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 algun dato en la cola de alarmas si no hay datos la tarea se bloquea hasta que alguien envie un dato a la cola, si hay dato se pregunta que evento contiene el dato resivido por la cola si es de monitorear lo que se hace es ver si el dato del sensor de humedad es menor al rango esperado si es asi se una variable en alto advirtiendo al sistema que hay una alarma, si se recicivio el evento de enviar se revisa si hay alarmas activas si hay se envia un mensaje de texto al

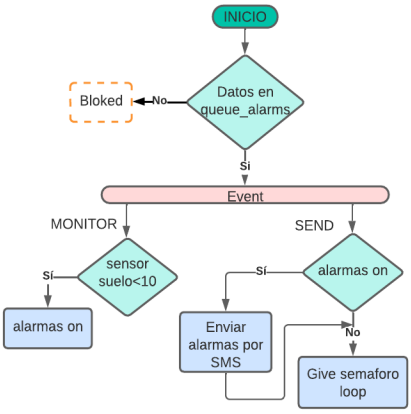


FIGURA 3.6. Diagrama de flujo tarea alarmas.

La ultima tarea que implento es la tarea que maneja la conexion del servidor que se describe en la figura La tarea comienza esperando datos en la cola de servidor

mqtt si no hay datos la tarea se bloquea, si hay datos se revisa que evento es el que se recibio tenemos tres posibles eventos UP, DOWN y SEND , si recivimos el evento de UP la tarea entra a una maquina de estados pra levantar el servidor la figura muestra la maquina de estados del evento UP, si el evento es de DOWN la tarea ejecuta la maquina de estados que se ve en la figura y el evento es SEND la tarea espera que existan datos en la cola de data para haci armar la trama y publicar los datos al borker MQTT.

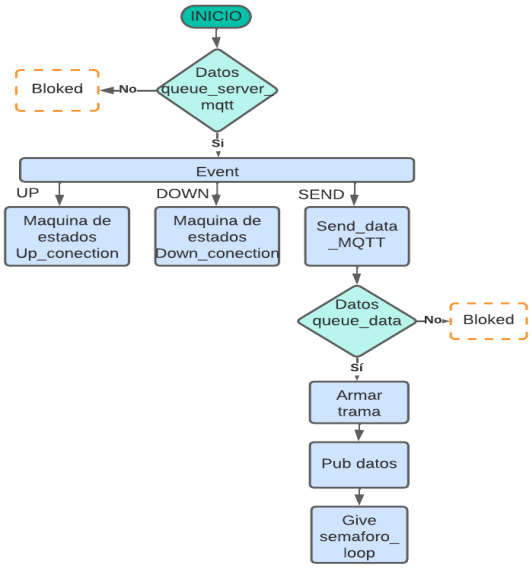


FIGURA 3.7. Diagrama de flujo tarea general conexion.

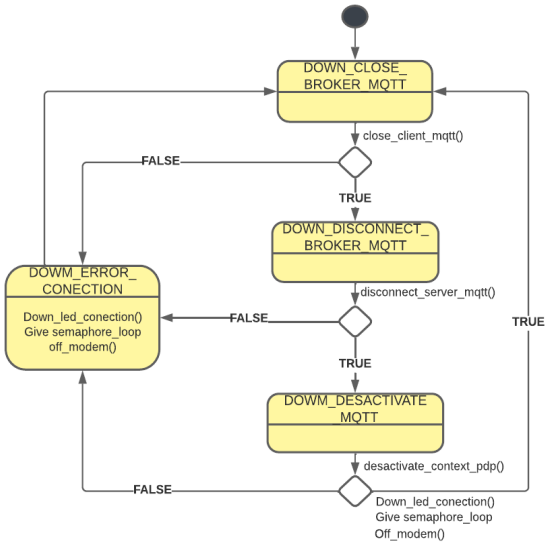


FIGURA 3.8. Maquina de estados dowm servidor.

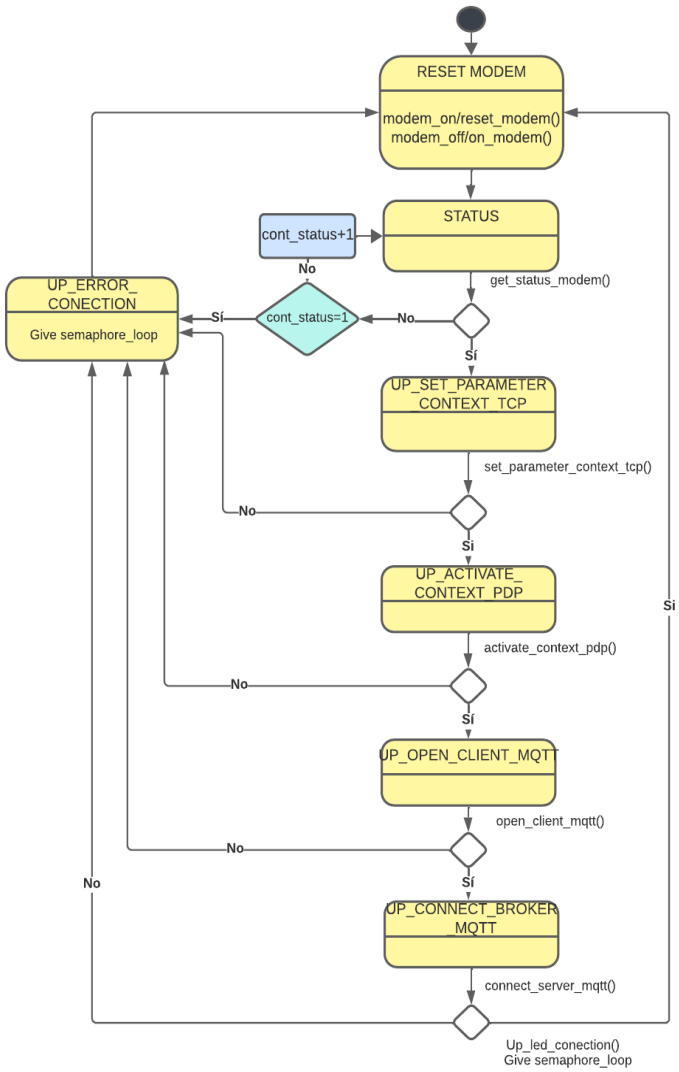


FIGURA 3.9. Maquina de estados up servidor.

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 En el codigo se ven las funciones mas itilizadas 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
11 void aht10Init(aht10_config_t *obj, aht10WriteFcn_t fncWritePort,
       aht10ReadFcn_t fncReadPort, delay1ms_t fncDelayPort)
12 {
13   obj->writeI2C=fncWritePort;
14   obj->readI2C=fncReadPort;
15   obj->delay_ms_I2C=fncDelayPort;
16 }
17 //!< Funcion para lanzar una medicion
18 aht10_status_fnc aht10_launch_measurement(aht10_config_t *obj)
19 {
20   uint8_t cmd[3] = {AHT10_CMD_TRIGGER_MEASUREMENT,AHT10_CMD_DATO_0,
       AHT10_CMD_DATO_1};
21   obj->status_fun=AHT10_ERROR;
22   obj->status_fun = obj->writeI2C(AHT10_ADDRESS_SLAVE ,cmd,3);
23   if (obj->status_fun==AHT10_OK)
24   {
25     obj->delay_ms_I2C(AHT10_DELAY_LAUNCH_MEASUREMENT);
26     if (aht10_get_status(obj)==SENSOR_IDLE)
27     {
28       obj->status_fun=obj->writeI2C(AHT10_ADDRESS_SLAVE,cmd,3);
29       obj->delay_ms_I2C(AHT10_DELAY_LAUNCH_MEASUREMENT);
30       return obj->status_fun;
31     }
32     else obj->status_fun=AHT10_ERROR;
33   }
34   return  obj->status_fun;
35 }
36 //!< Funcion para obtener el valor de la humedad del sensor
37 aht10_status_fnc aht10_get_humedity(aht10_config_t*obj, uint8_t *data)
38 {
39
40   if (obj== NULL)
41   {
42     return AHT10_ERROR;
43   }
44   obj->status_fun=AHT10_ERROR;
45   uint8_t bufferRead[6]={0};
46   uint32_t data_humedity=0;
47   obj->status_fun=aht10_launch_measurement(obj);
48   if (obj->status_fun==AHT10_OK)
49   {
50     obj->status_fun= obj->readI2C(AHT10_ADDRESS_SLAVE,bufferRead,6);
51     if (obj->status_fun==AHT10_OK)
52     {
53       data_humedity=(((uint32_t)bufferRead[1]<<16) | ((uint16_t)
       bufferRead[2]<<8) | (bufferRead[3]))>>4;
```

```
54        *data= HUMEDITY(data_humedity);
55      }
56    }
57    return obj->status_fun;
58  }
59  //Funcion para obtener la temperatura obtenida por el sensor
60  aht10_status_fnc aht10_get_temperature(aht10_config_t*obj, int8_t *data)
61  {
62    if (obj== NULL)
63    {
64      return AHT10_ERROR;
65    }
66    uint8_t buffer_read[6]={0};
67    uint32_t data_temperature=0;
68    obj->status_fun=AHT10_ERROR;
69    obj->status_fun=aht10_launch_measurement(obj);
70    if (obj->status_fun==AHT10_OK)
71    {
72      obj->status_fun=obj->readI2C(AHT10_ADDRESS_SLAVE ,buffer_read,6);
73      if (obj->status_fun==AHT10_OK)
74      {
75        data_temperature=((uint32_t)(buffer_read[3] & 0x0F)<<16) | ((
      uint16_t) buffer_read[4]<<8)| buffer_read[5];
76        *data= TEMPERATURE(data_temperature);
77      }
78    }
79    return obj->status_fun;
80  }
```

CÓDIGO 3.1. Funciones principales del driver del sensor AHT10.

Se implemento el driver del modulo de comunicacion.En el codigo se muestran las funciones mas utilizadas por el firmware del driver que son las de configuracion y activacion del APN, conexion y desconexion al servidor MQTT.

```
1  typedef struct
2  {
3    em_status_modem  status_modem;
4    em_state_server_mqtt_conection status_mqtt_server;
5    pf_send_data     send_data_device;
6    pf_reset_modem   f_reset_modem;
7    st_config_parameters_mqtt self_mqtt;
8    st_config_context_tcp self_tcp;
9    st_info_product info_product;
10   uint8_t           last_error;
11   char buffer_resp [100];
12   char *current_cmd;
13   em_bg96_error_handling ft_resp;
14 }st_bg96_config;
15
16 em_bg96_error_handling init_driver(st_bg96_config *self,pf_send_data
       ft_send_data_device,pf_reset_modem ft_reset_modem)
17 {
18   if (ft_send_data_device!=NULL) {
19     self->send_data_device=ft_send_data_device;
20 }
21   if (ft_reset_modem!=NULL) {
22     self->f_reset_modem=ft_reset_modem;
23 }
24     self->status_modem=OFF;
25     self->ft_resp=FT_BG96_OK;
26     self->last_error=BG96_NO_ERROR;
27     self->self_tcp.context_id=1;
```

```
28      self->self_tcp.context_type=1;
29      self->self_tcp.method_authentication=1;
30      self->self_tcp.tcp_password="";
31      self->self_tcp.tcp_username="";
32      self->self_mqtt.identifier_socket_mqtt=0;
33      self->self_mqtt.quality_service=0;
34      self->self_mqtt.port=1883;
35      self->self_mqtt.mqtt_client_id="123a56cb9";
36      self->status_mqtt_server=SERVER_MQTT_DOWN;
37      #ifdef UBIDOTS
38      self->self_mqtt.host_name="\"industrial.api.ubidots.com\"";
39      self->self_mqtt.mqtt_username="";
40      self->self_mqtt.mqtt_password="";
41      #endif
42      #ifdef THINGS_BOARD
43      self->self_mqtt.host_name="\"mqtt.thingsboard.cloud\"";
44      self->self_mqtt.mqtt_username="";
45      self->self_mqtt.mqtt_password="";
46      #endif
47      #ifdef MOSQUITTO
48      self->self_mqtt.host_name="\"test.mosquitto.org\"";
49      self->self_mqtt.mqtt_username="";
50      self->self_mqtt.mqtt_password="";
51      #endif
52      #ifdef ENTEL
53      self->self_tcp.tcp_apn="4g.entel";
54      #endif
55      #ifdef TIGO
56      self->self_tcp.tcp_apn="internet.tigo.bol";
57      #endif
58
59      return self->ft_resp;
60  }
61  em_bg96_error_handling get_status_modem(st_bg96_config* self)
62  {
63      self->ft_resp=FT_BG96_OK;
64      self->current_cmd="AT\r";
65      self->ft_resp=self->send_data_device(self->current_cmd,RS_BG96_OK,
    self->buffer_resp,1000);
66      if (self->ft_resp!=FT_BG96_OK)
67      {
68          self->last_error=BG96_ERROR_STATUS_MODEM;
69      }
70      return self->ft_resp;
71  }
72
73
74
75  em_bg96_error_handling send_sms_bg96(st_bg96_config *self,char*number,
    char*message)
76  {
77      self->ft_resp=FT_BG96_OK;
78      char buffer_message[256];
79      char buffer_number[20];
80      sprintf(buffer_number,"AT+CMGS=\"%s\"\r",number);
81      sprintf(buffer_message,"%s\x1a\r",message);
82
83      self->ft_resp=self->send_data_device(buffer_number,RS_BG96_SIGNAL,
    self->buffer_resp,12000);
84      if (FT_BG96_OK==self->ft_resp)
85      {
86          self->ft_resp=self->send_data_device(buffer_message,RS_BG96_OK
    ,self->buffer_resp,12000);
```

```
87          if (FT_BG96_OK!=self->ft_resp)
88          {
89              self->last_error=BG96_ERROR_SEND_SMS;
90          }
91      }
92      return self->ft_resp;
93  }
94
95  em_bg96_error_handling set_parameter_context_tcp(st_bg96_config *self)
96  {
97      self->ft_resp=FT_BG96_OK;
98      char cmd[100];
99      sprintf(cmd,"AT+QICSGP=%u,%u,\"%s\",\"%s\",\"%s\",%u\r",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);
100     self->ft_resp=self->send_data_device(cmd,RS_BG96_OK,self->
    buffer_resp,3000);
101     if (self->ft_resp!=FT_BG96_OK)
102     {
103         self->last_error=BG96_ERROR_SET_PARAMETER_CONTEXT_TCP;
104     }
105     return self->ft_resp;
106 }
107
108 em_bg96_error_handling activate_context_pdp(st_bg96_config *self)
109 {
110     self->ft_resp=FT_BG96_OK;
111     char cmd[30];
112     sprintf(cmd,"AT+QIACT=%u\r",self->self_tcp.context_id);
113     self->ft_resp=self->send_data_device(cmd,RS_BG96_OK,self->
    buffer_resp,15000);
114     if (self->ft_resp!=FT_BG96_OK)
115     {
116         self->last_error=BG96_ERROR_ACTIVATE_CONTEXT_PDP;
117     }
118     return self->ft_resp;
119 }
120
121 em_bg96_error_handling desactivate_context_pdp(st_bg96_config *self)
122 {
123     self->ft_resp=FT_BG96_OK;
124     char cmd[15];
125     sprintf(cmd,"AT+QIDEACT=%u\r",self->self_tcp.context_id);
126     self->ft_resp=self->send_data_device(cmd,RS_BG96_OK,self->
    buffer_resp,4000);
127     if (self->ft_resp!=FT_BG96_OK)
128     {
129         self->last_error=BG96_ERROR_DESACTIVATE_CONTEXT_PDP;
130     }
131     return self->ft_resp;
132 }
133 em_bg96_error_handling set_parameters_mqtt(st_bg96_config *self)
134 {
135     self->ft_resp=FT_BG96_ERROR;
136     char cmd_pdpid[30];
137     sprintf(cmd_pdpid,"AT+QMTCFG=\"pdpid\",%u,%u\r",self->self_mqtt.
    identifier_socket_mqtt,self->self_tcp.context_id);
138     self->ft_resp=self->send_data_device(cmd_pdpid,RS_BG96_OK,self->
    buffer_resp,300);
139     if (self->ft_resp!=FT_BG96_OK)
140     {
141         self->last_error=BG96_ERROR_SET_PARAMETER_MQTT;
```

```
142         }
143         return self->ft_resp;
144     }
145
146     em_bg96_error_handling open_client_mqtt(st_bg96_config *self)
147     {
148         self->ft_resp=FT_BG96_ERROR;
149         char cmd[100];
150         sprintf(cmd,"AT+QMTOPEN=%u,%s,%u\r",self->self_mqtt.
        identifier_socket_mqtt,self->self_mqtt.host_name,self->self_mqtt.
        port);
151         self->ft_resp=self->send_data_device(cmd,RS_BG96_CERO,self->
        buffer_resp,75000);
152         if (self->ft_resp!=FT_BG96_OK)
153         {
154             self->last_error=BG96_ERROR_OPEN_CLIENT_MQTT;
155
156         }
157
158         return self->ft_resp;
159     }
160     em_bg96_error_handling close_client_mqtt(st_bg96_config *self)
161     {
162         self->ft_resp=FT_BG96_ERROR;
163         char cmd[50];
164         sprintf(cmd,"AT+QMTCLOSE=%u\r",self->self_mqtt.
        identifier_socket_mqtt);
165         self->ft_resp=self->send_data_device(cmd,RS_BG96_OK,self->
        buffer_resp,3000);
166         if (self->ft_resp!=FT_BG96_OK)
167         {
168             self->last_error=BG96_ERROR_CLOSE_CLIENT_MQTT;
169         }
170         return self->ft_resp;
171     }
172
173     em_bg96_error_handling connect_server_mqtt(st_bg96_config *self)
174     {
175         self->ft_resp=FT_BG96_ERROR;
176         char cmd[150]={0};
177         sprintf(cmd,"AT+QMTCONN=%u,\"%s\",\"%s\",\"%s\"\r",self->self_mqtt
        .identifier_socket_mqtt,self->self_mqtt.mqtt_client_id,self->
        self_mqtt.mqtt_username,self->self_mqtt.mqtt_password);
178         self->ft_resp=self->send_data_device(cmd,RS_BG96_CERO,self->
        buffer_resp,10000);
179         if (self->ft_resp!=FT_BG96_OK)
180         {
181             self->last_error=BG96_ERROR_CONNECT_SERVER_MQTT;
182         }
183        return self->ft_resp;
184     }
185
186     em_bg96_error_handling disconnect_server_mqtt(st_bg96_config *self)
187     {
188         self->ft_resp=FT_BG96_ERROR;
189         char cmd[50]={0};
190         sprintf(cmd,"AT+QMTDISC=%u\r",self->self_mqtt.
        identifier_socket_mqtt);
191         self->ft_resp=self->send_data_device(cmd,RS_BG96_OK,self->
        buffer_resp,5000);
192         if (self->ft_resp!=FT_BG96_OK)
193         {
194             self->last_error=BG96_ERROR_DISCONNECT_SERVER_MQTT;
```

```
195         }
196         return self->ft_resp;
197     }
198
199
200     em_bg96_error_handling publish_message(st_bg96_config *self,char *
        topic,char *data)
201     {
202         self->ft_resp=FT_BG96_ERROR;
203         char cmd[50]={0};
204         char buffer_data[220]={0};
205         sprintf(buffer_data,"%s\x1a\r",data);
206         sprintf(cmd,"AT+QMTPUB=%u,0,0,0,\"%s\"\r",self->self_mqtt.
        identifier_socket_mqtt,topic);
207         self->ft_resp=self->send_data_device(cmd,RS_BG96_SIGNAL,self->
        buffer_resp,3000);
208         if (FT_BG96_OK==self->ft_resp)
209         {
210             self->ft_resp=self->send_data_device(buffer_data,RS_BG96_CERO,
        self->buffer_resp,15000);
211             if (self->ft_resp!=FT_BG96_OK)
212             {
213                 self->last_error=BG96_ERROR_PUBLISH_MESSAGE;
214             }
215         }
216         else self->last_error=BG96_ERROR_PUBLISH_MESSAGE;
217         return self->ft_resp;
218     }
```

CÓDIGO 3.2. Funciones principales del driver del sensor AHT10.

3.5. Desarrollo del hardware

Para el desarrollo del diseño del hardware se decidio utilizar como herramienta de diseño KICAD 6.0 por que es la herramienta que se aprendio en la especializacion.

3.5.1. Esquematico

Lo que se hizo fue desarrollar un tarjeta donde se puedan conectar todos los modulos que se utilizaron para el prototipo Al ser un prototipo no se realizon el diseño de un tarjeta con los compenentes mismo sino que se utlizaron modulo de desarrollo, lo que se hizo fue desarrollar un tarjeta donde se puedan conectar nuestros modulos utilizados como se el modulo celular, tarjeta de desarrollo STM32L432KC, los modulos sensores.

En la figura 3.10 se muestra la pagina raiz del esquematico del proyecto donde se puede ver la division del esquematico en 4 hojas, en la parte izquierda superior se ve el titulo de cada hoja en la parte derecha se ve la conexion entre las diferentes hojas, y en la parte izquierda inferior se ve el 3d del la tarjeta.

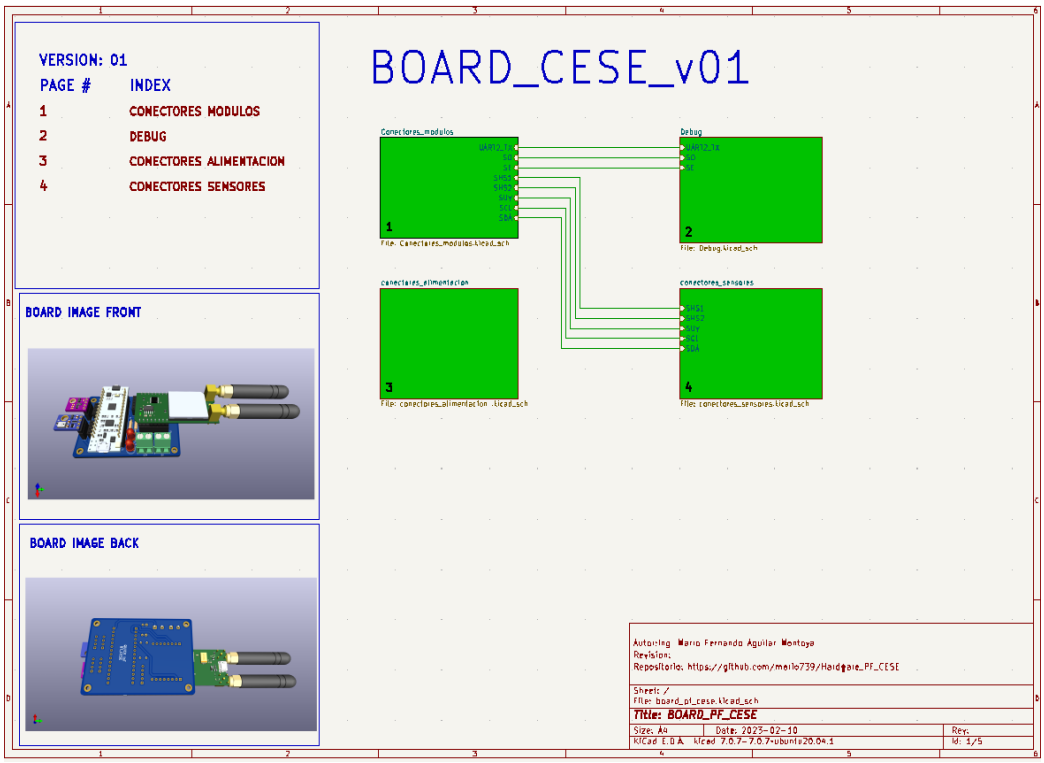


FIGURA 3.10. Esquematico pagina root.

Para el diseño del shield se consideraron las medidas de la modulos que se conectarian,en la figura 3.11 se muestra los dos conectores de los modulos mas importantes el modulo de comuicacion BG96, modulo NUCLEO-L432KC y las conexiones entre ellos.

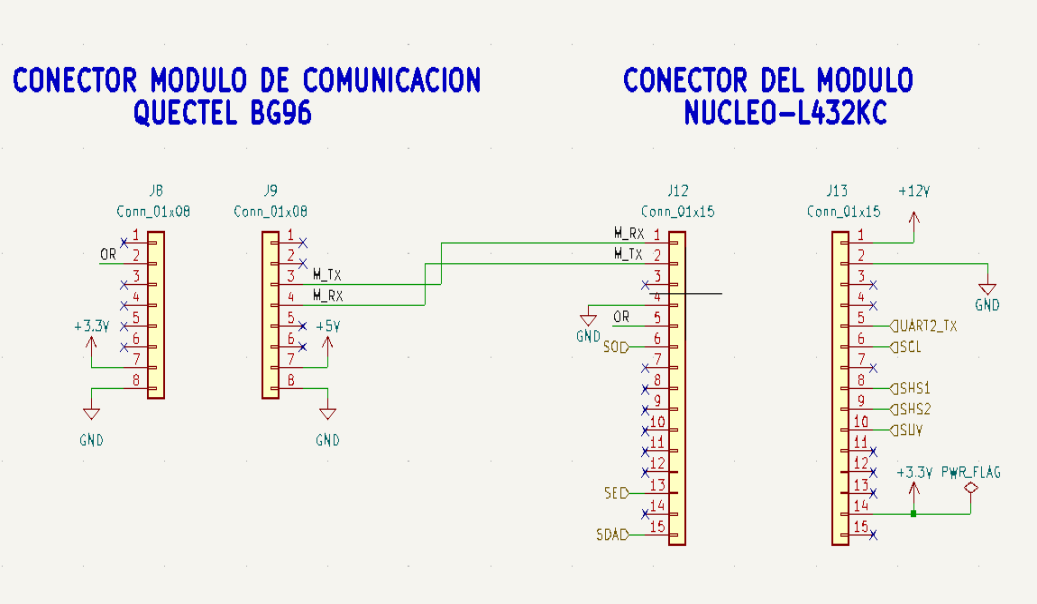


FIGURA 3.11. Esquematico conectores modulos.

Se colocaron dos leds para debug, uno para senalizar si se logro conectar al servidor MQTT y el otro led para senalizar si el sistema entra en un estado de error, se coloco un conector para una puerto serial por donde el modulo manda la secuencias de comandos que manda y recive al modulo de comunicacion.En la figura 3.12 se puede ver el circuito.

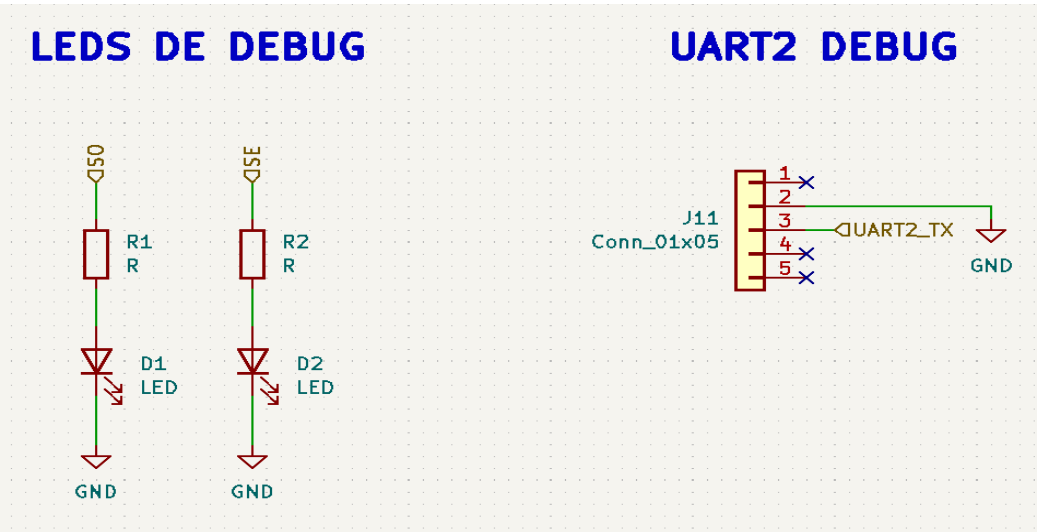


FIGURA 3.12. Esquematico interfaz de debug.

En la figura 3.13 se pueden ver los conectores que se pusieron 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.

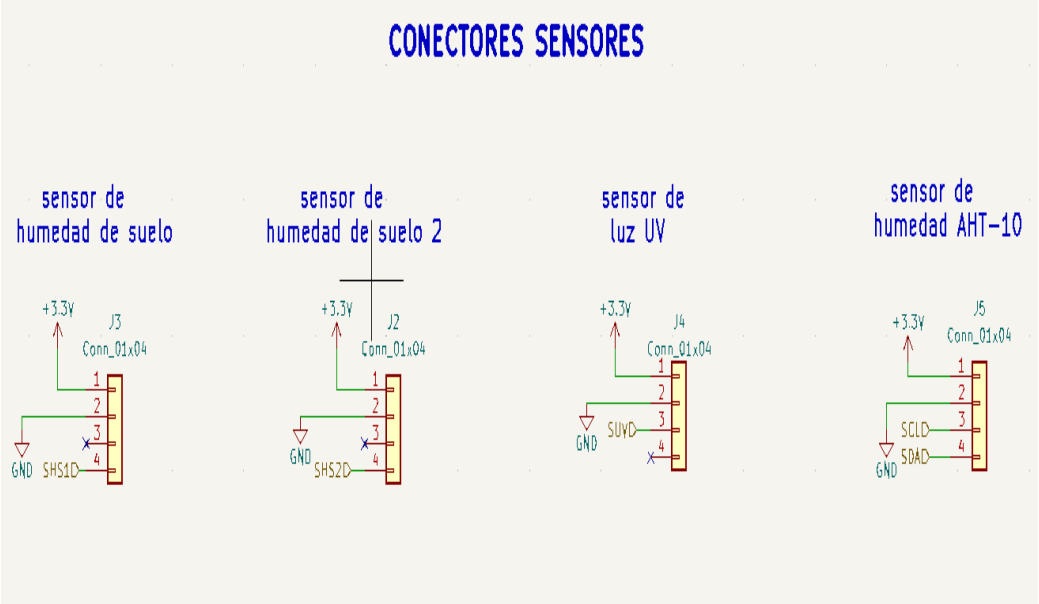


FIGURA 3.13. Esquematico conectores sensores.

Se colocaron dos colectores de alimentacion como se muestra en la figura 3.14, el conector de 12 v para alimentar al modulo del microcontroaldor y el otro conector para alimentar al modulo de comunicacion.

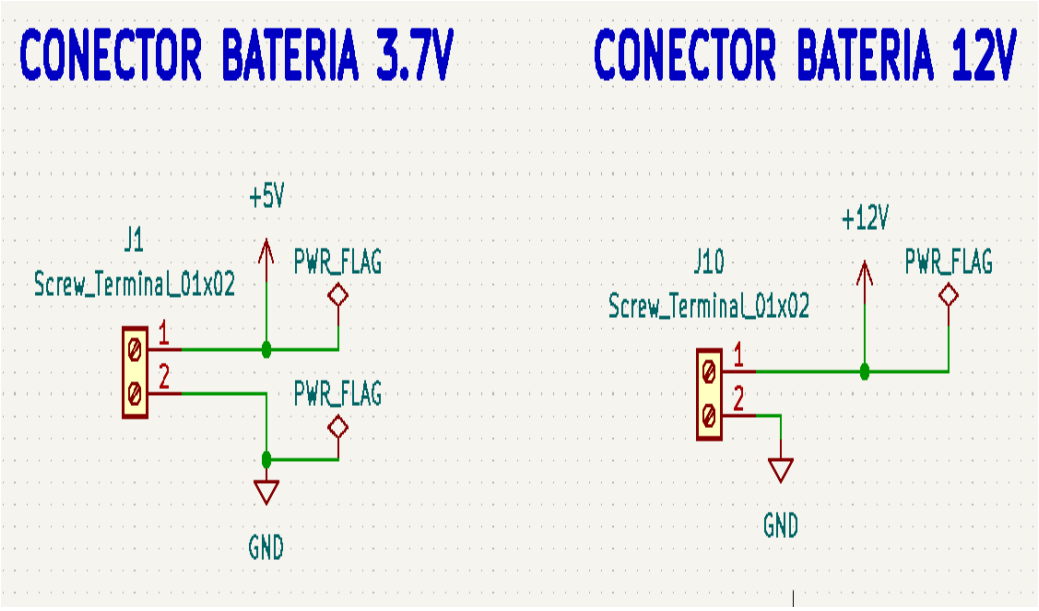
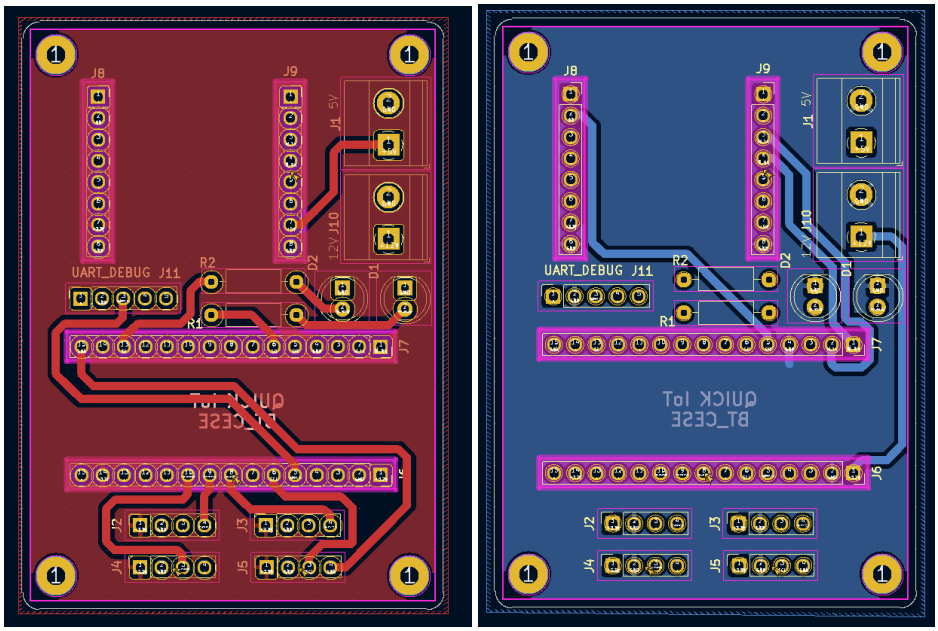


FIGURA 3.14. Esquematico conectores alimentacion.

3.5.2. PCB del hardware

La figura 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 se muestra del diseno de la tarjeta del circuito impreso en 3D.

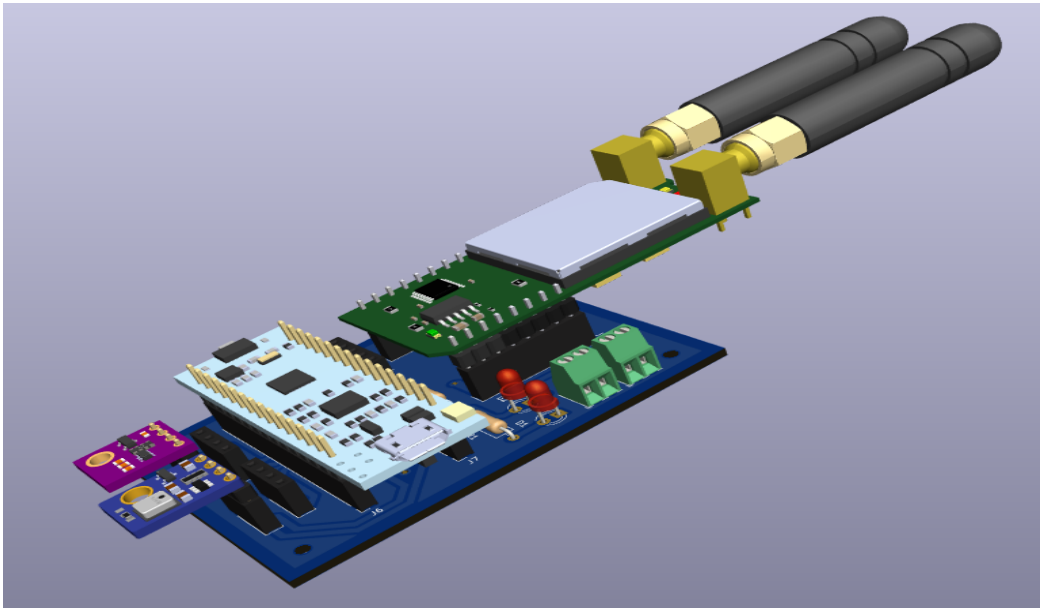


FIGURA 3.16. 3D del tarjeta desarrollada.

3.6. Paneles de visualizacion

La herramienta de visualizacion de ThingsBoard es muy versatil para el armado de paneles de visualizacion escalables y altamente configurables.

Se armaron dos paneles de visualizacion 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 3.17 se aprecia el panel principal de la interfaz grafica.A continuación se describira cada zona del panel principal. El panel esta 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 visualizacion del nodo donde podemos ver con mas detalle las variables medidas.
- Zona 2: Graficas 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 ubicacion de los nodo sensores implmentados.

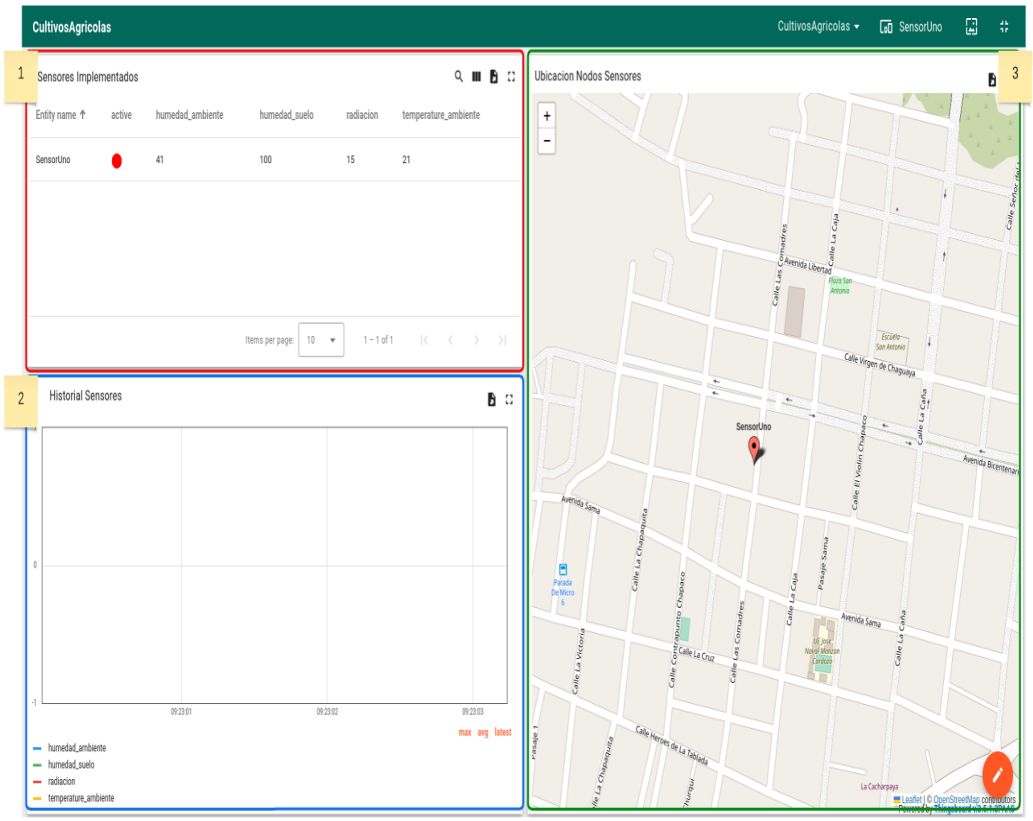


FIGURA 3.17. Panel principal de la interfaz grafica.

3.6.2. Panel nodo sensor

Para tener un mayor detalle de todos los parametros de monitoreo de cada nodo sensor se creo un panel secundario que se muestra en la figura 3.18.

El panel esta dividido en las siguientes zonas:

- Zona 1: Graficas que muestra los cambios que van teniendo los valores de las variables medidas por los sensores con respecto al tiempo.
- Zona 2: Widgets que muestran el ultimo valor obtenido por el nodo sensor de cada variable monitoreada.
- Zona 3: Tabla que monitorea las alarmas implementadas.
- Zona 4: Se tiene un mapa que muestra la ubicacion del nodo sensor.

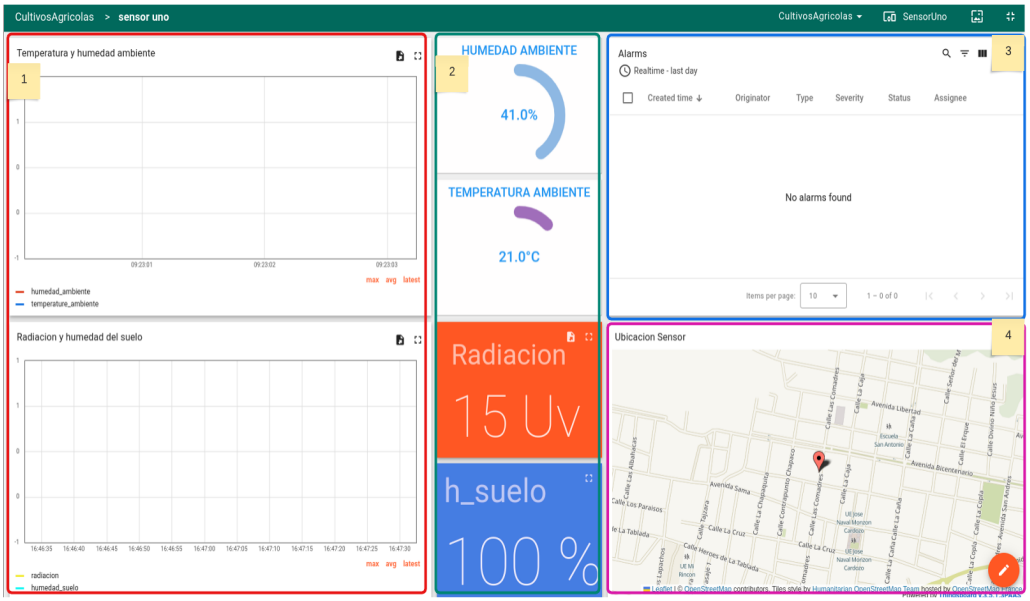


FIGURA 3.18. Panel nodo sensor.

Capítulo 4

Ensayos y resultados

4.1. Pruebas funcionales del hardware

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

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.

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] 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.

[7] MikroElectronic. *Especificacion de Producto,LTE IOT 2 CLICK*. Visitado: 2023-05-27. URL: https://www.mikroe.com/lte-iot-2-click.

[8] 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.

[9] naylampmechatronics. *Especificacion de Producto,MÓDULO SENSOR DE LUZ ULTRAVIOLETA (UV) ML8511*. Visitado: 2023-05-27. URL: https://ssdielect.com/temperatura/3885-aht10.html.

[10] 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/.

[11] STMicroelectronics. *Descripción del producto,Integrated Development Environment for STM32*. Visitado: 2023-05-27. URL: https://www.st.com/en/development-tools/stm32cubeide.html.

[12] FreeRTOS. *Acerca del Sistema Operativo,Descripción General*. Visitado: 2023-05-27. URL: https://www.freertos.org/RTOS.html.

[13] CEEDLING. *Descripción del producto,Descripción General*. Visitado: 2023-05-27. URL: http://www.throwtheswitch.org/ceedling.

[14] 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.

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

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

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

[18] ThingsBoard. *Descripción de la plataforma IoT,Definición de ThingsBoard*. Visitado: 2023-05-27. URL: https://thingsboard.io/.