Internet de las Cosas: Sistema centralizado de medición de temperatura y humedad

Alesandria, Alejo Samuel (Autor)
Departamento de Ingeniería Electrónica, UTN Facultad Regional San Francisco
Av. de la Universidad 501, San Francisco, Córdoba – Argentina
alealesandria@gmail.com
Cignetti, Mateo Antonio (Autor)
Departamento de Ingeniería Electrónica, UTN Facultad Regional San Francisco
Av. de la Universidad 501, San Francisco, Córdoba – Argentina
mateo@cignetti.ar
Delfino, Ramiro Jesús (Autor)
Departamento de Ingeniería Electrónica, UTN Facultad Regional San Francisco
Av. de la Universidad 501, San Francisco, Córdoba – Argentina
pokramiro12@gmail.com
Galliano, Ignacio (Autor)
Departamento de Ingeniería Electrónica, UTN Facultad Regional San Francisco
Av. de la Universidad 501, San Francisco, Córdoba – Argentina
igalliano@facultad.sanfrancisco.utn.edu.ar
Rinaudo, Facundo Nicolás (Autor)
Departamento de Ingeniería Electrónica, UTN Facultad Regional San Francisco
Av. de la Universidad 501, San Francisco, Córdoba – Argentina
facurinaudo99@gmail.com
Rodriguez, Manuel (Autor)
Departamento de Ingeniería Electrónica, UTN Facultad Regional San Francisco
Av. de la Universidad 501, San Francisco, Córdoba – Argentina
manuel.rodr.98@gmail.com

RESUMEN

Este proyecto implementa un sistema de IoT para la medición centralizada de temperatura y humedad. Utiliza tres placas ESP32-S3 conectadas a sensores DHT11, las cuales envían datos a una Raspberry Pi que actúa como computadora central. La Raspberry Pi corre un servidor MQTT y emplea Node-RED para procesar y almacenar los datos en una base de datos SQL. Los datos recopilados se pueden visualizar tanto en una aplicación móvil desarrollada con App Inventor como en una interfaz de PC creada con Node-RED.

Palabras clave: IoT, MQTT, SQL, Node-RED.

Objetivos

- ✓ Aplicar correctamente las técnicas de IoT vistas en clase, para así realizar un desarrollo sobre una situación posible en la vida profesional.
- ✓ Implementar una interconexión entre ESP y Broker, para almacenar los datos en una Base de Datos y visualizarlos en un dashboard o aplicación móvil.

INTRODUCCIÓN

Las placas ESP32-S3 recopilan datos de temperatura y humedad, transmitiéndolos al servidor MQTT integrado en Raspberry Pi 4. A su vez, Node-RED (integrado en Raspberry) recibe estos datos, los procesa y los guarda en una base de datos SQL.

La aplicación móvil permite a los usuarios conectarse al servidor, visualizar los datos en tiempo real y recibir alertas si las condiciones superan ciertos umbrales. También ofrece opciones para gestionar las suscripciones a los sensores, dependiendo del sensor seleccionado se accede a determinado valores proporcionados por el tópico correspondiente.

Por otro lado, la interfaz de PC proporciona una visualización gráfica de los datos históricos y permite realizar consultas detalladas. Este sistema ofrece una solución completa y accesible para la monitorización ambiental centralizada, demostrando la eficacia de las tecnologías IoT.

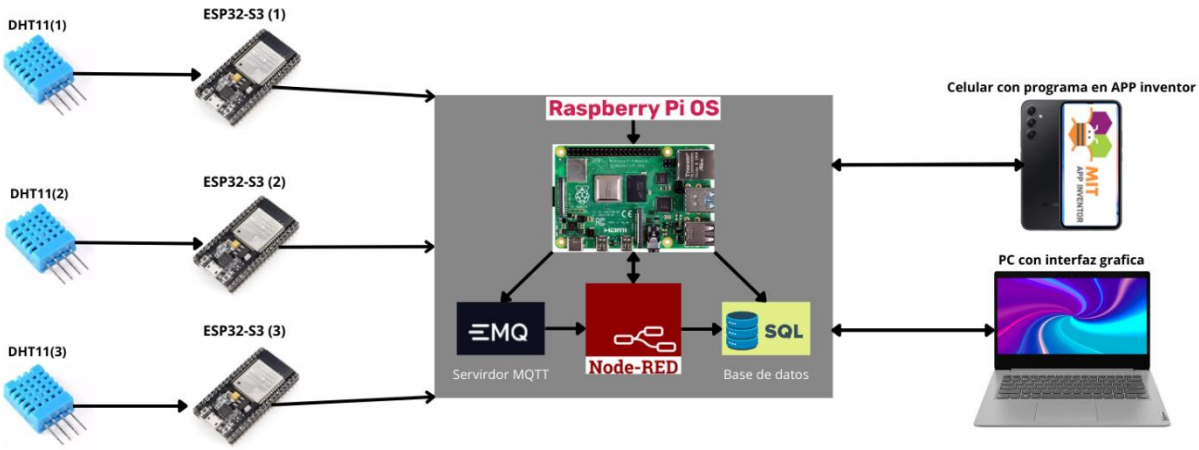

Figura 1. Diagrama del sistema

"MIT App Inventor es un entorno de programación visual intuitivo que permite a todos crear aplicaciones completamente funcionales para teléfonos Android, iPhone y tabletas Android/iOS. El proyecto MIT App Inventor busca democratizar el desarrollo de software empoderando a todas las personas, especialmente a los jóvenes, para pasar del consumo de tecnología a la creación de tecnología". (Instituto Tecnológico de Massachusetts, 2024)

DESARROLLO

Integración en Raspberry Pi

Para el servidor central, se utilizó una Raspberry Pi 4 Model B de 8 GB de RAM, una computadora de pequeña escala con un microprocesador ARM que corre el sistema operativo Raspberry Pi OS, basado en Debian Linux. En la computadora, se instalan los programas necesarios para realizar la comunicación entre dispositivos, así como la toma de datos y el guardado en una base de datos.

Broker MQTT – EMQX

MQTT es un protocolo de mensajería basado en estándares, o un conjunto de reglas, que se utiliza para la comunicación de un equipo a otro, está diseñado como un transporte de mensajería de publicación/suscripción liviano ideal para conectar dispositivos remotos de bajos recursos y limitado ancho de banda.

El servidor mencionado se encuentra alojado en una Raspberry Pi, haciendo que múltiples dispositivos conectados a una misma red Wi-Fi puedan enviar datos a un tópico de este servidor, estos tópicos son canales de comunicación a los cuales los dispositivos pueden publicar mensajes o suscribirse para recibir mensajes

Una vez los datos enviados, Node-Red accede al tópico para almacenarlos en la Base de Datos o enviarlos a la aplicación móvil.

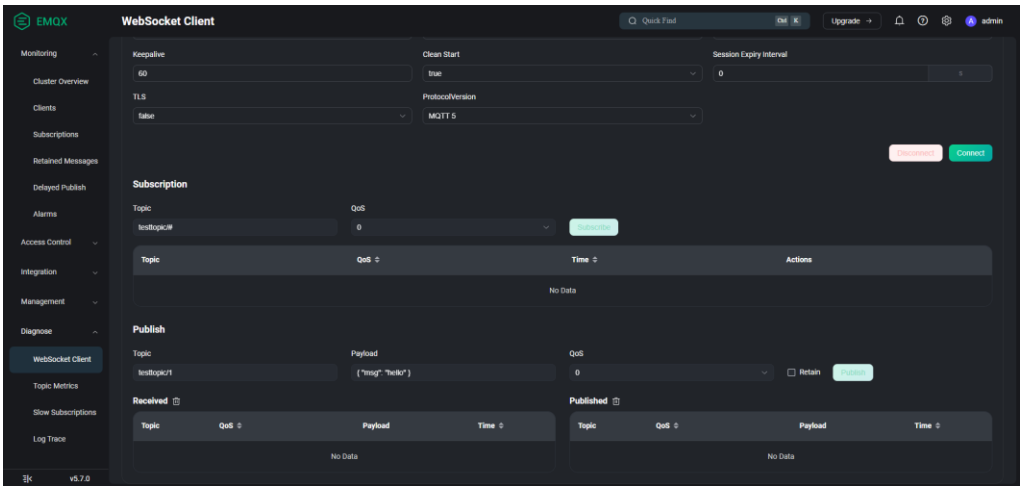

Figura 2. Servidor MQTT

Node-Red

En la Raspberry Pi, se instala Node-Red para programar la lógica del sistema en general. Utilizando el mismo, se programa para que cuando se reciba un dato de algún sensor, se guarde en una base de datos y se envíe mediante el bróker MQTT para la recepción en los dispositivos móviles. Por otra parte, se programa un Dashboard accesible por computadoras en la misma red local, que permite visualizar el último dato, así como las diez últimas mediciones por sensor guardadas en la base de datos.

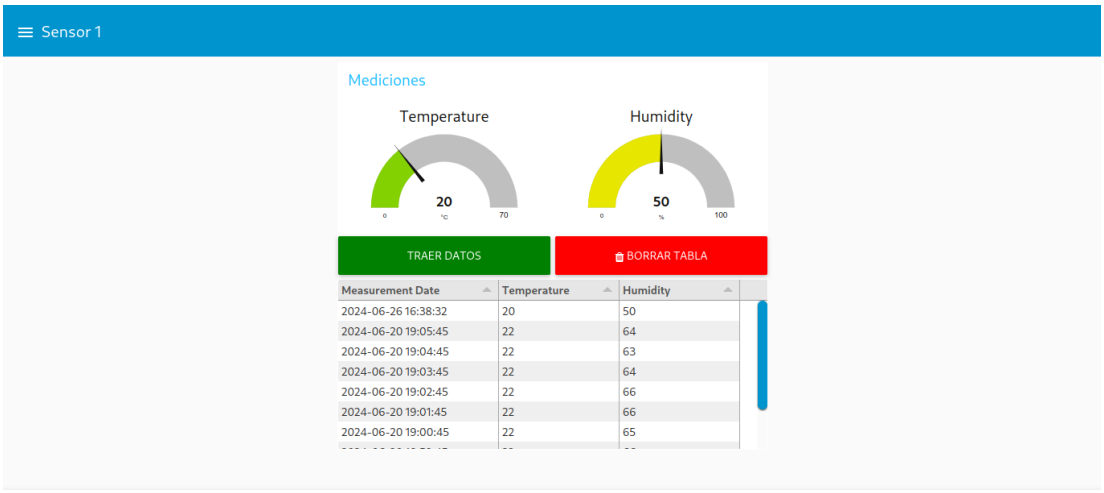

Figura 3. Dashboard local en Raspberry

Base de Datos – SQLite

Para la gestión de la base de datos, se utiliza, junto con su plugin en Node-Red, para poder realizar la creación de la base de datos, la carga de datos y la lectura de estos. En la base de datos, se crean dos tablas:

- "Sensor", donde se tiene el ID del sensor y su nombre, los cuales son "Laboratorio", "Laboratorio Aula" y "Laboratorio Oficina"
- "Measurement", donde se tiene la ID de la medición (autoincrementable), la ID del sensor que realizó la medición, la fecha y hora de la medición, el valor de temperatura y el valor de humedad.

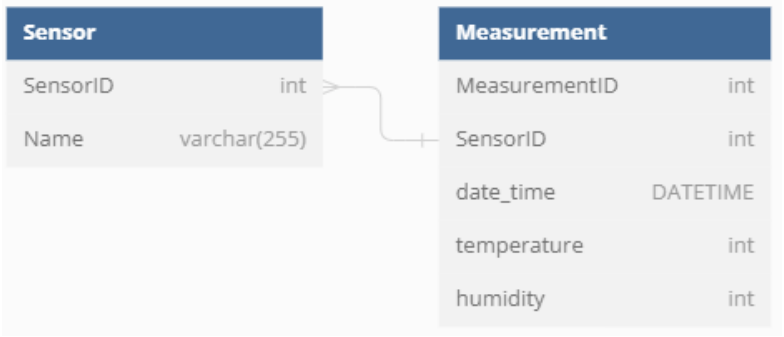

Figura 4. Esquema de base de datos

ESP-32: Mediciones y Conexión con Broker MQTT

En esta sección, se detalla el proceso de desarrollo de la programación del ESP32-S3 para la obtención de mediciones de temperatura y humedad de un sensor DHT11 y la publicación de estos datos a un broker MQTT, en este caso, EMQX. Este desarrollo se divide en tres códigos principales: "main.c", "wifi.c", y "mqtt.c".

Programación en ESP-IDF

El archivo "main.c" contiene el código principal del programa que se ejecuta en el ESP32-S3. Las principales tareas que realiza este código son:

- Inicialización del sensor DHT11: Para esta acción se hace uso de la librería "dht.c" (Uss, 2024). Aquí se configura el tipo de sensor como DHT11 para empezar a obtener mediciones de temperatura y humedad.
- Lectura periódica de datos: Se implementa un bucle que obtiene periódicamente los datos del sensor.
- Procesamiento de Datos: Los datos obtenidos del sensor se procesan para asegurar que estén en el formato correcto para luego de enviarlos al broker MQTT.

El archivo "wifi.c" se encarga de establecer la conexión Wi-Fi necesaria para que el ESP32-S3 pueda comunicarse con el broker MQTT (Espressif, 2024). La función de este código es:

- Inicialización del Módulo WiFi: Aquí se configuran los parámetros de red (SSID y contraseña), y se establece una conexión con dicha red, en modo estación.

El archivo "mqtt.c" contiene el código necesario para la comunicación con el broker MQTT y la publicación de los datos del sensor (Espressif, 2024). Las funciones clave incluyen:

- Inicialización del Cliente MQTT: Configuración de los parámetros de conexión al broker (dirección del servidor, puerto, credenciales).
- Publicación de Mensajes: Envío de los datos de temperatura y humedad al broker MQTT en un formato adecuado.

Configuración adicional

Para poder utilizar el ESP32-S3 con estas funciones, se deben realizar las siguientes configuraciones:

Dentro de "main.c":

- Seleccionar el pin GPIO al que se conectará el sensor DHT11 (SENSOR_GPIO)
- Actualizar el URL del broker MQTT (CONFIG_BROKER_URL)
- Elegir el *topic* bajo el cual se enviarán los datos hacia el broker (TOPIC)

Dentro de "wifi.c":

- Actualizar el SSID y la contraseña a la cual se debe conectar el dispositivo (ESP_WIFI_SSID y ESP_WIFI_PASS)

App Inventor

En esta sección se describe una interfaz de usuario para una aplicación que permite conectarse y desconectarse de un servidor MQTT, usando dos botones que cambian de color según el estado de la conexión, verde conectado, rojo desconectado. Los datos de

conexión, como la dirección del servidor y las credenciales, están configurados por defecto para que el usuario se conecte fácilmente, sin necesidad de que ingrese algún dato adicional.

Otra parte del programa se encarga de controlar la recepción de mensajes desde el servidor MQTT, según el tópico suscripto. Cuando llega un mensaje, se inserta en una lista y se actualiza su visualización en la interfaz del usuario. Además, se hacen comprobaciones para actualizar tres listas de monitoreo basadas en el estado de tres sensores, mostrando un mensaje de error si alguna de las condiciones no se cumple.

El programa también muestra el comportamiento al pulsar en uno de los tres botones de referencia de las habitaciones en la aplicación. Al presionar un botón, se gestiona la suscripción a un tópico MQTT designado. Si el tópico ya está suscrito, se muestra una alerta notificando esto. Si no, se suscribe al tópico designado, se limpia la lista visual y se marca el sensor como suscrito. Además, se cancela la suscripción a los tópicos de los otros dos sensores.

Por último, el botón de borrar dato elimina el dato seleccionado de la lista y, si ya no hay datos que borrar, se muestra una alerta notificando que la lista está vacía.

Figura 5. Aplicación en smartphone.

CONCLUSIÓN

Se aplicó un sistema IoT donde diversos dispositivos, ESP32, envían datos a un servidor MQTT alojado en una Raspberry Pi, dentro de la misma se procesan los datos que se almacenan en una base de datos para luego poder visualizarlos.

Además, la conexión entre el ESP y Broker se pudo hacer efectiva de manera correcta y funcional.

REFERENCIAS

[1] Massachusetts Institute of Technology. *MIT App Inventor*. 2024. URL: https://appinventor.mit.edu/

[2] Espressif. *Wi-Fi Driver*. Espressif Systems Co, 2024. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-guides/wifi.html

[3] Espressif. *ESP-MQTT*. Espressif Systems Co, 2024. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/protocols/mqtt.html

[4] Ruslan V. Uss. *ESP-IDF Components library*. 2024. URL: https://github.com/UncleRus/esp-idf-lib