Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Proyecto estudiantil categoría B:

Implantación de Sistemas Embebidos para una Solución IoT completa

Autores:
Fourcade, Santiago
Ramonda, José
Scozzina, Tomás Agustín

Docente a Cargo:
Belmonte, Javier Gustavo

Introducción

Este trabajo, realizado en el marco de la materia Sistemas Embebidos Avanzados I, consistió en diseñar e implementar una red IoT con múltiples nodos para el monitoreo y control remoto de variables distribuidas. Se integraron tecnologías como MQTT, microcontroladores con sistemas embebidos, bases de datos SQL y una app Android para visualización y control. La red implementada permite visualizar tanto mediciones actuales como datos históricos, además de comandar dispositivos de forma remota, consolidando así una solución completa de medición y control distribuido.

Objetivos Generales

- Implementar una red de medición y control distribuida basada en tecnologías IoT, que permita la adquisición de datos desde al menos tres nodos físicamente independientes.
- Habilitar el control remoto de al menos un dispositivo mediante una interfaz accesible desde dispositivos móviles y computadoras.
- Desplegar un servidor IoT completo sobre Linux embebido que actúe como broker MQTT, centro de procesamiento y visualización de datos.
- Desarrollar un tablero de control accesible local y remotamente, que permita monitorear y comandar variables de la red.
- Permitir el acceso a la red tanto desde dispositivos fijos como móviles, empleando distintos tipos de conexión (WiFi, UART, etc).

Objetivos particulares

La red se organiza en torno a una Raspberry Pi 3 B+ con sistema operativo Linux, que desempeña múltiples funciones dentro del sistema. Esta unidad aloja un bróker MQTT encargado de centralizar las comunicaciones, una base de datos para el almacenamiento de la información, y un servidor Node-RED que permite gestionar y visualizar los datos a través de un panel (dashboard), además de facilitar la interacción con la base de datos. Asimismo, la Raspberry Pi actúa como nodo funcional, enviando datos provenientes de sensores conectados directamente a ella.

Los demás nodos son ejecutados en:

- Un módulo ESP8266 NodeMCU V2 encargado de adquirir y transmitir una señal analógica proveniente de un potenciómetro.

- Una placa de desarrollo ESP32S que transmite la medición de un fotorresistor (LDR).
- Un módulo ESP 8266 Node MCU Wemos Mini conectado al kit de desarrollo FRDM-KL43Z de NXP mediante UART. La ESP deberá funcionar como nexo entre el servidor y la KL43Z para que esta última pueda brindar información del estado de los leds y pulsadores integrados, así como de las mediciones del sensor de luz ambiental y el acelerómetro MEMS, todos ellos incluidos en el kit, así como tambien recibir comandos para encender y apagar los leds.

Las interfaces para el usuario serán:

- Un dashboard de node-red accesible mediante navegador por cualquier dispositivo dentro de la misma red.
- Una aplicación móvil nativa para Android, desarrollada específicamente para interactuar con el sistema mediante MQTT.

En ambas se debe poder visualizar el estado de todos los sensores de la red, así como de los pulsadores y leds del kit KL43Z, pudiendo estos últimos ser comandados. Además, se deberán poder visualizar gráficamente los datos históricos guardados en la base de datos.

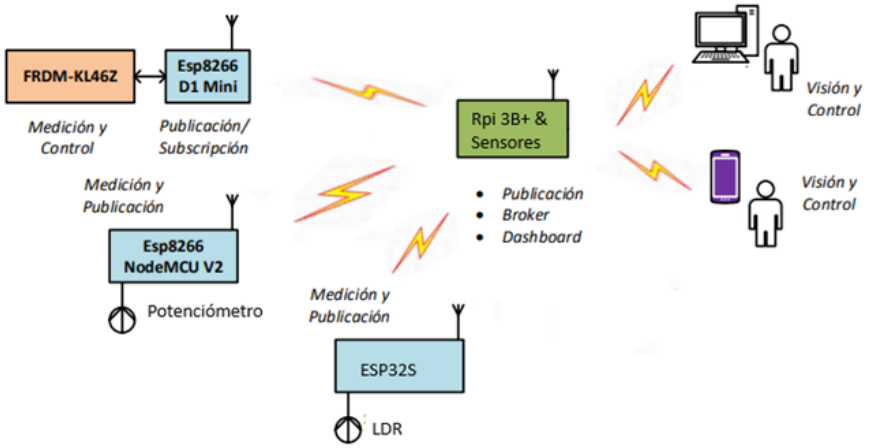

Figura 1: Esquema de interconexión

Desarrollo

El diseño del sistema se centró en una arquitectura IoT distribuida compuesta por varios nodos, siendo el nodo principal una Raspberry Pi. Este dispositivo cumple múltiples funciones: adquisición de datos desde sensores conectados directamente y publicación a través de MQTT, almacenamiento en base de datos, visualización vía dashboard web y operación como broker MQTT.

Nodo Raspberry Pi

Se instaló Linux con entorno gráfico, aunque el manejo fue íntegramente por terminal vía SSH. Se conectaron dos sensores I2C (BMP280 y TMP175), integrados al sistema como dispositivos nativos mediante un overlay de Device Tree compilado (.dtbo), lo que habilita los drivers automáticamente al iniciar. De esta forma, los datos se exponen como archivos accesibles desde el espacio de usuario.

Se desarrolló una aplicación en lenguaje C, utilizando la librería MQTT de Eclipse Paho, que periódicamente (cada 30 segundos) accede a los archivos de sensores y publica los

valores formateados y escalados en tópicos MQTT. Se incluyen, además de los sensores externos, los datos del sensor de temperatura interno del CPU.

El sistema configura mensajes de conexión y desconexión (will message) usando la opción retain, lo que permite a los clientes conocer el estado en línea de los nodos. Además, la Raspberry Pi aloja el broker Mosquitto configurado para aceptar conexiones externas (además del loopback). Se instaló también el servidor de base de datos MariaDB, junto con Apache, PHP y PHPMyAdmin, utilizados para la gestión visual de los datos.

Se instaló Node-RED como servicio, que permite crear flujos visuales en un dashboard web accesible desde la red local, mostrando el estado de cada nodo y sus variables, tanto en tiempo real como históricos. Se configuraron tres flujos separados: uno para cargar datos en MariaDB escuchando los tópicos de sensores, almacenar la información y consultar los históricos para graficarlos; otro para la visualización en tiempo real con valores actualizados continuamente desde todos los nodos; y un tercero para gestionar los comandos de encendido y apagado de LEDs del nodo KL43Z, además de mostrar el estado de sus switches y sensores.

Nodo KL43Z + ESP8266 D1 Mini

Este nodo implementa una separación funcional: la KL43Z realiza adquisiciones de hardware y la ESP8266 publica los datos por MQTT. Se comunican vía UART, con ring buffers en la KL43Z para mayor robustez. La ESP8266, programada en C++ con PubSubClient, publica los datos que recibe de la KL43Z (aceleración cada 200ms, luz, switches y estados de LEDs cada 5s o ante cambios) y retransmite los comandos recibidos hacia la KL43Z.

Nodos ESP8266 NodeMCU V2 y ESP32S

Ambos nodos funcionan de manera análoga: leen sensores analógicos (un potenciómetro en el ESP8266 y una fotorresistencia en el ESP32S), escalan el valor a un porcentaje (0–100 %) y lo publican cada 200ms en un tópico MQTT. Están programados en C++ con la misma biblioteca y emiten mensajes de estado al conectarse o desconectarse.

Aplicación Android

Se desarrolló una aplicación nativa en Java que actúa como cliente MQTT. Permite al usuario ingresar manualmente la dirección IP del broker y visualizar, en un dashboard simple, los valores en tiempo real de los sensores. También se incorporaron botones para controlar los LEDs de la KL43Z. La aplicación muestra los estados de conexión de los nodos mediante los tópicos de status. Cuando los dispositivos están desconectados, los widgets indican “Desconocido” y, al reconectarse, los valores se actualizan en tiempo real.

Resultados

Tras muchos cambios producto de errores tanto conceptuales como en la implementación, se logró que el sistema funcione como es esperado. Los dashboards muestran la información de forma clara y fluida y el comportamiento es robusto frente a las desconexiones. En el siguiente link, se pueden observar capturas del dashboard y de la aplicación.

Discusión y conclusiones

Este proyecto fue desarrollado con fines didácticos, combinando funciones poco comunes para cubrir necesidades típicas de una red como comunicación, retención de datos y visualización en tiempo real. Se destaca su versatilidad y escalabilidad, útiles para adaptarlo a distintos contextos. Una mejora pendiente es la accesibilidad: por falta de tiempo se limitó a una red local. A futuro, se propone habilitar acceso remoto vía Internet, lo que exige mejorar la seguridad, ya que actualmente cualquier cliente con la IP del servidor puede publicar. Una posible solución sería implementar autenticación con claves.