Estación meteorológica utilizando ESP32.

1. Descripción general del proyecto.

La idea del proyecto es tener una estación meteorológica barrial cooperativa, donde se tiene un nodo central al cual distintos nodos cliente van a ir enviando datos. Con estos datos se puede tener una visión en “tiempo real” del estado, historial y datos concretos del clima de un barrio.

2. Alcance y objetivos del Proyecto.

- Crear una red de sensores IoT que recolectan y transmiten datos.
- Establecer un sistema central que reciba, filtre, y almacene los datos de todas las estaciones.
- Analizar y visualizar los datos utilizando herramientas como Grafana.

3. Arquitectura del sistema.

La arquitectura del sistema está conformada por los siguientes dispositivos y/o aplicaciones:

- Nodos cliente: Conformados por ESP32 que recolectan datos de sus sensores y los publican en un tópico de MQTT específico.
- Nodo central: Recibe la información de los nodos cliente y la procesa con los siguientes componentes.
    - Broker MQTT: recibe todos los mensajes publicados por los ESP32.
    - Node-RED: Se utiliza para gestionar la suscripción a los tópicos, procesar y parsear los datos y enviarlos finalmente a InfluxDB.
    - InfluxDB: almacena los datos recibidos. Cada medición se guarda con su valor; también se guarda una marca de tiempo y la ubicación del dispositivo. Estos últimos conforman los tags.
    - Grafana: se conecta a InfluxDB. Se encarga de renderizar los datos creando gráficos, mapas y tablas que muestra en un dashboard..

4. Nodo cliente.

El nodo cliente es el encargado de recolectar los datos de sus respectivos sensores y de enviarlos al nodo central. Para enviar los datos, debe de seguir comunicándose por MQTT enviando sus datos por el tópico barrio/estación/{CLAVE}.

CLAVE es un valor que permite identificar al nodo cliente e indica al nodo central los tópicos en los cuales debe de escuchar. Los motivos de esta implementación se detallan en la sección correspondiente al nodo central. El valor CLAVE se obtiene en la página web levantada desde node-red.

El código del cliente está hecho de tal manera que sea fácil para el usuario poder modificarlo a su gusto, siendo así una “plantilla” para las necesidades específicas de cada uno. También tiene por separado el archivo de configuración “config.h” donde se puede especificar fácilmente los datos que se van a utilizar, como la clave, las coordenadas y la localidad.

El código de cada cliente puede ser distinto siempre y cuando respeten el protocolo y utilicen la clave que se les dio, caso contrario no va a ser aceptado por el sistema.

El nodo cliente envía los datos como CBOR. Decidimos utilizar este tipo de formato de datos para tener una mayor eficiencia respecto a JSON, al utilizar una representación binaria es considerablemente más liviano, lo que hace que consuma menos memoria y CPU en formatear los datos. Si bien no es una mejora sustancial si comparamos un solo envío, al estar el ESP32 enviando datos frecuentemente, este formato permite un mayor ahorro de batería al consumir menos recursos, especialmente a largo plazo.

El nodo cliente envía mediante MQTT los siguientes datos:

- Temperatura
- Humedad
- Sensación térmica
- Presión
- Lluvia
- Viento
- Radiación solar
- Calidad del aire
- Estación ID
- Localidad
- Latitud
- Longitud

Es importante que en “Estación ID”, “Localidad”, “Latitud” y “Longitud” cada cliente especifique sus propios datos.

5. Nodo central.

5.1 Página web.

Para tener un sistema más completo, tenemos una página web que va a dar información en general sobre el proyecto, y va a servir de guía para alguien que quiera ingresar a la red por primera vez. Así como proporcionar un mapa de visualización de los datos, y un link al dashboard completo. En esta misma página se va a generar la clave y se va a guardar en el sistema.

5.2 Node-RED.

Node-RED contiene toda la lógica de procesamiento del servidor. Es el encargado de recibir los datos de los nodos clientes y guardarlos en la base de datos. Además de manejar la lógica del bot de Telegram.

El servidor escucha en los /barrios/estaciones/{clave}. Esto se diseñó por un motivo: como el servidor no procesa todos los mensajes que se envía a un tópico, sino que tiene un temporizador que le indica cuándo “tomar” un mensaje de los tópicos (los mensajes intermedios se descartan), es importante que los clientes no se solapen entre sí. Si un cliente modifica el código de su dispositivo para enviar mensajes cada 15 segundos,

muchos de estos no van a ser recibidos gracias al temporizador, pero es posible que el cliente que modificó su código envíe datos a un tópico que es utilizado por otro cliente y sobrescribe continuamente los mensajes de este último. Como el temporizador da ventaja a quien mando por última vez al tópico, aquellos clientes que manden continuamente podrían tener ventaja sobre aquellos que respeten el tiempo. Asignando la clave, es matemáticamente imposible que los clientes usen la misma clave, por ende no se cruzan entre sí y puede chequearse que cada cliente respete el tiempo de manera particular.

El Node-RED es también el encargado de cargar y servir la página web y los recursos que la misma requiera.

5.3. Telegram

Tenemos un bot en Telegram que envía los datos recopilados diariamente de forma automática así como también a demanda por parte de los usuarios mediante comandos. Pueden ver los datos de una localidad en concreto o de todas las localidades.

5.4. Grafana

En Grafana vamos a tener disponible toda la visualización de los datos, con distintas métricas y un mapa que permita ver la información de manera más clara. Este dashboard está disponible públicamente para el acceso de los usuarios.

5.5. InfluxDB

En nuestra base de datos, aparte de almacenar todos los datos previamente formateados que envían los clientes, vamos a almacenar las claves generadas en la página web, aprovechando así la infraestructura que ya está disponible.