Informe de Trabajo: Proyecto Heri-Sure inc.

Alumnos:

André Francisco Castillo Flores

Oscar Nydza Nicpoñ

Máster en Ingeniería de Computadores y Redes

Universidad Politécnica de Valencia

Asignatura: Redes Inalámbricas de Sensores

Docente: Ángel Francisco Perles Ivars

Contenido

Resumen

El **Proyecto Heri-Sure** es una solución integral de monitoreo ambiental basada en tecnologías IoT (Internet de las Cosas), diseñada para la recolección, transmisión, análisis y visualización en tiempo real de datos críticos de temperatura y humedad. La arquitectura del sistema está compuesta por sensores de alta precisión (Milesight EM320-TH y Dragino S31B-LB) conectados a través de la red LoRaWAN utilizando el nodo RAK3172.

Estos dispositivos permiten la recopilación de datos en entornos distribuidos con un bajo consumo de energía y una alta eficiencia en la transmisión de información a largas distancias. Los datos recolectados son enviados a **The Things Network (TTN)**, donde se gestionan y redirigen a un **broker MQTT**, que actúa como intermediario para el procesamiento y almacenamiento de la información en **InfluxDB**, una base de datos optimizada para series temporales.

Posteriormente, los datos son visualizados de forma intuitiva mediante **Grafana**, una poderosa herramienta de análisis que permite la creación de dashboards interactivos para el monitoreo en tiempo real y la generación de alertas basadas en condiciones específicas.

Un aspecto diferenciador del Proyecto Heri-Sure es la integración de técnicas de **Machine Learning** para el análisis avanzado de los datos.

El desarrollo del proyecto ha seguido una metodología ágil, permitiendo iteraciones rápidas y continuas mejoras. La solución final es altamente escalable y adaptable a diferentes entornos, lo que la convierte en una herramienta versátil para sectores como la agricultura de precisión, la gestión de recursos naturales, la industria y cualquier área donde el monitoreo ambiental sea crucial. Con Heri-Sure, no solo se obtiene un sistema de monitoreo, sino una plataforma inteligente capaz de transformar datos en información valiosa para la toma de decisiones estratégicas.

Introducción

Antecedentes

El deterioro de edificaciones históricas y bienes patrimoniales es un problema global. Factores ambientales como humedad, temperatura, contaminación y cambios estructurales pueden comprometer la integridad de estos espacios. Tecnologías como IoT y Machine Learning permiten monitorear y prevenir estos daños mediante sistemas de alerta temprana.

Objetivos

General: Diseñar e implementar un sistema IoT para el monitoreo y predicción de riesgos en bienes patrimoniales utilizando sensores, redes LoRaWAN, modelos de Machine Learning y una pantalla de tinta electrónica.

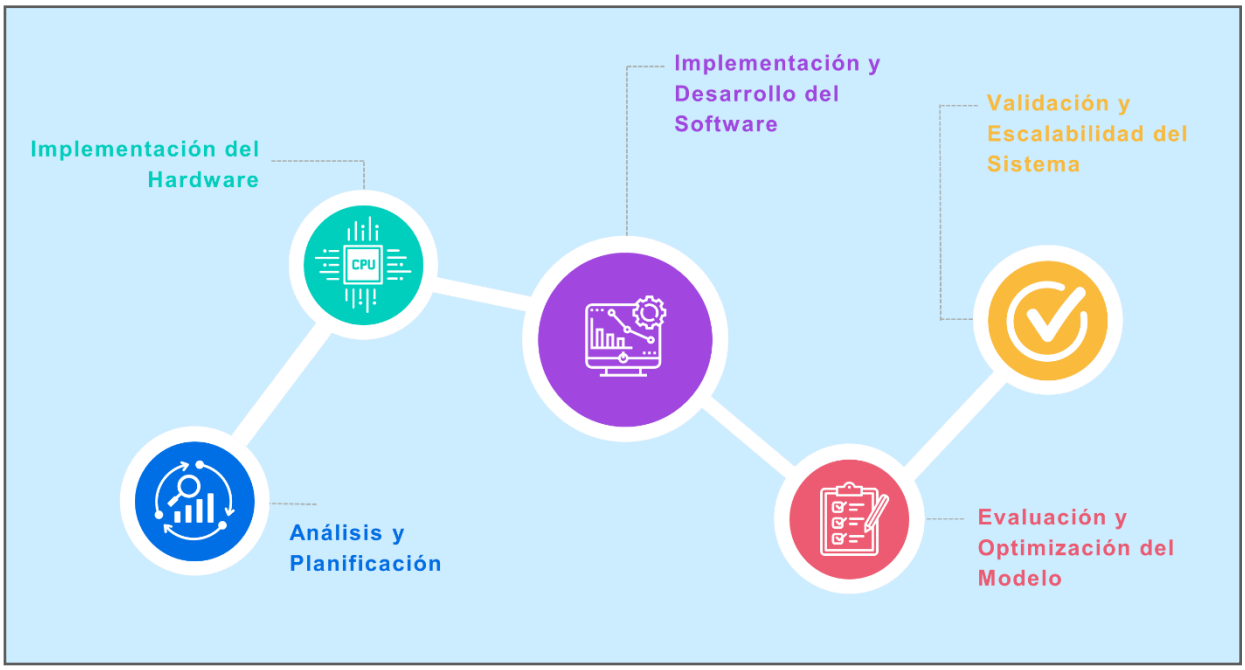
Específicos:

1. Implementar sensores para la medición de temperatura y humedad en un entorno patrimonial.
2. Configurar una infraestructura de comunicación basada en LoRaWAN, MQTT e InfluxDB.
3. Integrar Grafana para la visualización de datos en tiempo real.
4. Añadir una pantalla E-Ink para la visualización tanto de la temperatura como de la humedad.
5. Desarrollar un modelo de Machine Learning para predecir condiciones ambientales adversas.

Justificación

La preservación del patrimonio requiere nuevas estrategias tecnológicas para monitorizar, prevenir y mitigar riesgos de deterioro. Este proyecto aporta una solución innovadora al utilizar sensores inalámbricos, algoritmos predictivos y análisis de datos en tiempo real para optimizar la conservación de estructuras históricas.

Metodología General



El desarrollo del Proyecto Heri-Sure se ha basado en una metodología estructurada que permite una integración eficiente de hardware, software y análisis de datos. El proceso se divide en cinco etapas principales, cada una de las cuales contribuye de forma crucial a la construcción de una solución IoT robusta y escalable. A continuación, se describen cada una de estas etapas:

Análisis y Planificación:

En esta fase inicial, se realizó un análisis exhaustivo de los requerimientos del proyecto, considerando tanto los aspectos técnicos como funcionales. Se definieron los objetivos clave, las necesidades del sistema, revisión de los sensores más adecuados para la recolección de datos de temperatura y humedad. Además, se evaluaron las tecnologías de comunicación (LoRaWAN, MQTT, InfluxDB, etc.) y se estableció un plan de trabajo detallado, definiendo los recursos, cronograma y riesgos potenciales.

Implementación del Hardware

Esta etapa se centró en la integración y configuración del hardware necesario para la captura de datos ambientales. Se instalaron y configuraron los sensores **Milesight**

EM320-TH y **Dragino S31B-LB**, junto con el nodo **RAK3172** para la transmisión de datos mediante LoRaWAN. Se realizaron pruebas de conectividad para asegurar la correcta comunicación entre los dispositivos y la red, optimizando la disposición física de los sensores para una recolección precisa y continua.

Implementación y Desarrollo de Software

En esta fase se desarrolló el software necesario para la gestión de datos, comenzando con la configuración de **The Things Network (TTN)** para la recepción de datos provenientes del hardware. Se implementó un **broker MQTT** para el manejo eficiente de la mensajería entre dispositivos y servidores. Los datos se almacenaron en **InfluxDB**, una base de datos optimizada para series temporales, y se visualizaban en tiempo real mediante **Grafana**, permitiendo el monitoreo y análisis interactivo. Además, se escribieron scripts en **Python** para la automatización del flujo de datos y su procesamiento. Todo creado en **Linux**, desde la BD hasta el bróker y extracción de datos para el modelo ML.

Evaluación y Optimización del Modelo

En esta etapa se integraron técnicas de **Machine Learning** para agregar valor al análisis de los datos recolectados. Se desarrolló un modelo predictivo que permite anticipar los valores de temperatura y humedad, así como un sistema de detección de anomalías para identificar comportamientos inusuales en los datos ambientales. También se implementó un clasificador para categorizar el estado ambiental (seco, húmedo, caluroso, templado, etc.). Se realizó un proceso iterativo de prueba y error para optimizar el rendimiento de los modelos, ajustando hiperparámetros y mejorando la precisión de las predicciones.

Validación y Escalabilidad del Sistema

Finalmente, se validó el funcionamiento integral del sistema mediante pruebas de rendimiento y robustez, asegurando la precisión en la recolección, transmisión, almacenamiento y análisis de los datos. Se verificó la fiabilidad del sistema de alertas

configurado en Grafana, así como la eficacia de los modelos de Machine Learning. Además, se evaluó la escalabilidad del sistema, considerando la posibilidad de integrar más sensores y expandir la arquitectura para aplicaciones en entornos más amplios. Esta fase aseguró que el proyecto no solo cumple con los requisitos actuales, sino que está preparado para futuras expansiones y mejoras.

Diseño del Sistema

Arquitectura del Sistema

El sistema Heri-SURE se compone de:

Hardware:

- ✓ Sensores de temperatura y humedad (Milesight EM320-TH y Dragino S31B-
- ✓ Nodo RAK3172 para la comunicación LoRaWAN.
- ✓ Gateway LoRaWAN.

Software:

- ✓ Arduino IDE para la programación de los nodos.
- ✓ The Things Network (TTN) para la gestión de dispositivos LoRaWAN.
- ✓ Broker MQTT para la transmisión de datos.
- ✓ InfluxDB para el almacenamiento de datos.
- ✓ Grafana para la visualización en tiempo real.
- ✓ Python para el desarrollo del modelo de Machine Learning.

Esquema de comunicación

RAK3172 → GW LoRaWAN → TTN → MQTT →Telegraf → InfluxDB →Grafana→ML

Modelo de Machine Learning

Se utilizarán algoritmos de Regresión y Redes Neuronales para predecir niveles críticos de humedad y temperatura basados en datos históricos almacenados en InfluxDB.

Implementación y Desarrollo

Esta sección describe el proceso de puesta en marcha del sistema, incluyendo la configuración del hardware, el desarrollo del firmware para el nodo RAK3172 y la integración con la base de datos y herramientas de visualización. Se detallan las configuraciones necesarias para garantizar una comunicación eficiente entre los sensores, la red LoRaWAN y la plataforma de monitoreo.

Configuración del Hardware

Para la implementación del sistema, se utilizaron diversos componentes de hardware, incluyendo sensores de temperatura y humedad conectados a un nodo LoRaWAN. La configuración del hardware incluyó los siguientes pasos:

1. **Montaje del nodo RAK3172**

- Se conectó el sensor Milesight EM320-TH al nodo RAK3172 para medir temperatura y humedad.
- Se aseguró la alimentación correcta mediante una batería de 3.7V compatible con el módulo.

2. **Configuración del Gateway LoRaWAN**

- Se utilizó un gateway LoRaWAN para la retransmisión de datos hacia The Things Network (TTN).
- Se realizó la asociación del nodo con TTN mediante la configuración del DevEUI, AppEUI y AppKey en la plataforma.

3. **Pruebas de conectividad**

- Se verificó la recepción de paquetes en TTN mediante la interfaz de "Live Data".
- Se ajustó la potencia de transmisión y la tasa de datos (Spreading Factor) para optimizar la cobertura de red.
- El hardware quedó listo para la transmisión continua de datos hacia la infraestructura en la nube.

Desarrollo del firmware

El firmware del nodo RAK3172 fue desarrollado utilizando **Arduino IDE**, asegurando la correcta captura y transmisión de datos. El desarrollo incluyó los siguientes pasos:

1. **Inicialización del sensor y comunicación LoRaWAN**
    - Se configuró el sensor para realizar mediciones en intervalos predefinidos.
    - Se estableció la comunicación con TTN utilizando la librería LoRaWAN.h.
2. **Lógica de transmisión de datos**
    - Se programó la captura de valores de temperatura y humedad y su empaquetado en un payload JSON.
    - Se definió la frecuencia de transmisión para optimizar el consumo energético.
3. **Integración de alertas y control del LED**
    - Se implementó la lógica para activar el LED en función de umbrales de temperatura/humedad.
    - Se añadió una verificación de estado para asegurar que el LED refleje correctamente las condiciones de alerta.

El código final se cargó en el nodo RAK3172 y se realizaron pruebas para validar la funcionalidad del sistema.

Configuración de la base de datos y visualización

Para almacenar y visualizar los datos, se implementó una infraestructura basada en **InfluxDB, MQTT y Grafana**.

1. **Configuración de InfluxDB**
    - Se creó la base de datos proyecto_RIS_OYA para almacenar los datos del sensor.
    - Se configuró Telegraf para consumir los datos de MQTT y almacenarlos en InfluxDB.
2. **Integración con MQTT**

- Se estableció la conexión entre TTN y un **broker MQTT**, permitiendo la recepción de datos en tiempo real.
- Se configuró un cliente MQTT en Telegraf para extraer y procesar los mensajes entrantes.

3. **Visualización en Grafana**
   - Se añadieron paneles en Grafana para mostrar la evolución de temperatura y humedad.
   - Se configuraron alertas en **Telegram y en el LED del RAK3172** para notificar cambios críticos en las condiciones ambientales.

Esta infraestructura permite la monitorización en tiempo real y facilita la toma de decisiones basada en los datos recolectados.

Visualización por pantalla de tinta electrónica

Para facilitar la visualización de información y habilitar la posibilidad de que el usuario final interactúe con el módulo desarrollado, se ha optado por utilizar el RAK14000.

El dispositivo tiene dos partes principales: el display de tinta electrónica tricolor (blanco, negro y rojo) y tres botones.

La idea propuesta para el uso de este módulo consiste en mostrar por pantalla la temperatura y la humedad. Si se supera un umbral definido como, por ejemplo, 24 grados Celsius +- 4 el fondo se mostrará en rojo a modo de alarma.

En cuanto a los botones, se propone la siguiente funcionalidad:

1. Cuando nos encontramos en la situación “crítica” podemos presionar uno de los botones a modo de “ACK”, revirtiendo el color de la pantalla a blanco. Pasado un tiempo, si la situación continúa, se volverá a tornar roja.
2. Los otros dos botones permitirán subir o bajar la temperatura límite (tanto por abajo como por arriba) en incrementos de 1 grado Celsius, permitiendo al usuario ajustar la temperatura a sus necesidades sin tener que cambiar el código. Cuando el dispositivo se reinicie la temperatura límite volverá a ser de 24 grados Celsius (sin memoria).

La implementación en este caso ha resultado fallida. A continuación se detallan los dos intentos de implementación que se han realizado.

Implementación usando librerías de Adafruit

Como primer acercamiento se planteó el uso directo de librerías del fabricante del display. Más concretamente, seguimos un ejemplo en el que se utilizaban las librerías "Adafruit_GFX" y "Adafruit_EPD" en el que se realizaba un control muy a bajo nivel de la pantalla. Cabe destacar que existen dos versiones de este código: la del RAK14000 versión monocromática (blanco y negro) y la del RAK14000 versión tricolor (rojo, blanco y negro). En nuestro caso se trata de la segunda, con modelo DEPG0213RWS800F41HP. Este se puede comprobar en la parte anterior de la pantalla.

Comparado con el segundo acercamiento, se trata de una implementación de más bajo nivel, ya que se configura la pantalla de forma manual con, por ejemplo, las siguientes líneas de código:

```
#define POWER_ENABLE   WB_IO2
#define EPD_MOSI       MOSI
#define EPD_MISO       -1     // not use
#define EPD_SCK        SCK
#define EPD_CS         SS
#define EPD_DC         WB_IO1
#define SRAM_CS        -1     // not use
#define EPD_RESET      -1     // not use
#define EPD_BUSY       WB_IO4

// 2.13" EPD with SSD1680, width=250 pixels, heigh=122 pixels
Adafruit_SSD1680 display(250, 122, EPD_MOSI,
                         EPD_SCK, EPD_DC, EPD_RESET,
                         EPD_CS, SRAM_CS, EPD_MISO,
                         EPD_BUSY);
```

A nivel personal, nos cuesta entender exactamente el motivo de que se utilicen estos parámetros y no otros, ya que no terminaba de funcionar por motivos que explicaremos más adelante.

Con esta implementación los botones sí parecían funcionar a ratos, aunque solo el central y el derecho y con mucho bouncing. En cuanto al izquierdo, a veces al presionarlo sí parecía funcionar, pero lo hacía como si fuera el botón derecho.

En cuanto a la pantalla como tal, en ningún momento se ha conseguido mostrar ninguna imagen siguiendo este ejemplo. Se intentó conectar en ambas bahías (J2 y J3) sin éxito.

El código completo del ejemplo se puede encontrar en el siguiente enlace:

WisBlock RAK14000 Tricolor Example

Consideramos importante destacar que toda la documentación sobre estos ejemplos llevaba al de la pantalla monocromática y no al de la pantalla tricolor.

Implementación basada en RUI3

Como segundo acercamiento se planteó la idea de usar alguna librería diseñada para funcionar con RUI3 como ya se hizo con el sensor de temperatura y humedad. Para ello por suerte en el repositorio de RUI3 Best Practice existe un ejemplo que usa el RAK14000, que usaremos como base para intento.

RUI3 Sensor Node EPD

Se trata de un ejemplo completo en el que se usan distintos sensores para obtener información del entorno que luego se envía usando LoRaWAN. De forma adicional, esta información se muestra en una pantalla de tinta electrónica como la que pretendemos usar en este proyecto.

Del repositorio utilizamos los ficheros RAK14000_epd.h y RAK14000_epd.cpp, que contienen el código necesario para utilizar la pantalla. Se pueden observar funciones como init_rak14000, rak14000_text, refresh_rak14000 y set_temp_rak14000 entre otros. Esta última es la que utilizaremos a continuación.

Para ello, dentro del setup hacemos la siguiente prueba sencilla:

```
void setup()
{
    LED_Init();
    Serial.begin(115200);
    init_rak14000();

    set_temp_rak14000(12.4);
    refresh_rak14000();
}
```

En la que le pasamos una temperatura de prueba a la pantalla y la refrescamos. Se puede observar que la configuración de la pantalla la lleva a cabo la función init_rak14000, en contraste a la opción anterior. A pesar de la sencillez del código, no se muestra ninguna imagen en pantalla.

Esto puede deberse a que en el ejemplo de código se utiliza una RAK14000 monocromático en lugar de tricolor como puede verse en la siguiente línea de la función init_rak14000:

EPD_213_BW epd;

Donde el BW nos indica que se trata de la monocromática. La librería no tiene otras opciones de pantalla. Sin embargo, existen comentarios de usuarios en foros que comentan que la pantalla tricolor debería funcionar con el código de la monocromática, aunque con un tinte rojizo. No ha sido nuestro caso, lo cual nos lleva a preguntarnos si la pantalla no viene defectuosa de fábrica o se ha dañado en algún momento.

Conclusiones tras utilizar la pantalla de tinta electrónica

Vistas ambos intentos, podemos concluir que la implementación no ha sido exitosa. En nuestra opinión, las razones del por qué son las siguientes:

1. No existen ejemplos suficientes para el RAK14000 tricolor: por norma general y debido a que tanto la pantalla monocromática como la tricolor tienen el mismo nombre, los ejemplos se centran en la monocromática al tratarse de la opción más popular.
2. Es posible que la pantalla vino defectuosa de fábrica o se dañó en algún momento al transportarla.
3. El RAK3172 no soporta este módulo a través de RUI3 o el firmware no da soporte a la pantalla tricolor.
4. Tanto la pantalla como el sensor de temperatura y humedad utilizan I2C para la conexión y entran en algún tipo de conflicto: aunque poco probable, debemos mencionar esta posibilidad, ya que tanto en la documentación oficial del RAK14000 como en la del RAK1901 se menciona que si se utilizan otros periféricos conviene revisar si el mapeo de los pines es el correcto utilizando el WisBlock Pin Mapper. Esta herramienta sin embargo no tiene en cuenta al RAK3172 y por tanto no nos ha sido de utilidad. Cabe destacar que aun desconectando el RAK1901 la pantalla no funcionaba correctamente.

En conclusión, consideramos que con el suficiente tiempo se podría llegar a implementar una versión funcional basándonos en las librerías existentes de Adafruit, ya que tanto la pantalla como el controlador aparentan estar bien documentados a simple vista. Sin embargo, nos vemos obligados a dejarlo a medias, aunque creemos que en un proyecto como el planteado sería una funcionalidad bastante útil de cara a la utilización por el usuario final.

Implementación del modelo de Machine Learning

A continuación, se detallan los pasos realizados para el desarrollo del modelo de Machine Learning (ML) en Google Colab, enfocado en la predicción de temperatura y humedad, la detección de anomalías y la clasificación de estados ambientales.

1. **Preparación del Entorno**

Primero, configuramos el entorno de trabajo en Google Colab para asegurar que contáramos con todas las bibliotecas necesarias:

```
# Instalación de librerías necesarias
!pip install pandas scikit-learn matplotlib

Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.55.8)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

- pandas: para la manipulación y análisis de datos.
- scikit-learn: para el desarrollo de modelos de ML.
- matplotlib: para la visualización de los resultados.

2. Carga y Preprocesamiento de Datos:

En nuestra terminal de Linux, primero instalamos la librería necesaria para interactuar con Influx desde Python:

```
InfluxDB Installation

1  pip install influxdb-client pandas
2
```

- **influxdb-client**: Permite conectar y consultar la base de datos InfluxDB.
- **pandas**: Para la manipulación de los datos y su exportación a CSV.

Creamos un script en Python llamado extract_data.py para realizar la conexión con InfluxDB, ejecutar la consulta y guardar los datos en formato CSV.

```
andrefflores13@Andr-DESKTOP: ~
  GNU nano 7.2
from influxdb import InfluxDBClient
import pandas as pd

# Configuración de InfluxDB
host = "localhost"
port = 8086
database = "proyecto_RIS_OYA"

# Conexión a InfluxDB
client = InfluxDBClient(host=host, port=port)
client.switch_database(database)

# Consulta InfluxQL para extraer temperatura y humedad
query = """
    SELECT "uplink_message_decoded_payload_temperature_1", "uplink_message_decoded_payload_relative_humidity_2"
    FROM mqtt_consumer
    LIMIT 1000
"""

# Ejecución de la consulta
result = client.query(query)

# Procesar los resultados
points = list(result.get_points())
df = pd.DataFrame(points)

# Guardar en CSV
df.to_csv("sensor_data.csv", index=False)
print("Datos extraídos y guardados en 'sensor_data.csv'")
```

Luego ejecutamos:

```
(myenv) andrefflores13@Andr-DESKTOP:~$ nano extract_data.py
(myenv) andrefflores13@Andr-DESKTOP:~$ python extract_data.py
Datos extraídos y guardados en 'sensor_data.csv'
(myenv) andrefflores13@Andr-DESKTOP:~$ ls
env  extract_data.py  heri-sure-env  homeassistant  ibex-demo-system  mqtt_test.py  mqtt_test.pyx  mqtt_to_influx.py  myenv  sensor_data.csv  siliconcompiler  simulacion_influx.py  snap  ttn_uplink_simulator.py
(myenv) andrefflores13@Andr-DESKTOP:~$
```

Se cargaron los datos del archivo sensor_data.csv, que contenía registros de temperatura y humedad recopilados por los sensores del proyecto.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt

# Cargar el archivo CSV (asegúrate de subirlo a Google Colab)
df = pd.read_csv('sensor_data.csv')

# Verificar los primeros datos
df.head()
```

	time	uplink_message_decoded_payload_temperature_1	uplink_message_decoded_payload_relative_humidity_2
0	2025-02-11T12:38:33.974995Z	22.9	51.0
1	2025-02-11T12:41:35.148712Z	22.7	51.5
2	2025-02-11T12:42:34.435883Z	22.8	51.0
3	2025-02-11T12:43:34.147376Z	22.8	51.0
4	2025-02-11T12:44:35.636593Z	22.8	51.0

Transformaciones realizadas:

- Eliminación de valores nulos o inconsistentes.
- Conversión de la columna de tiempo a un formato de fecha y hora adecuado.

- Normalización de los datos si era necesario para mejorar el rendimiento del modelo.

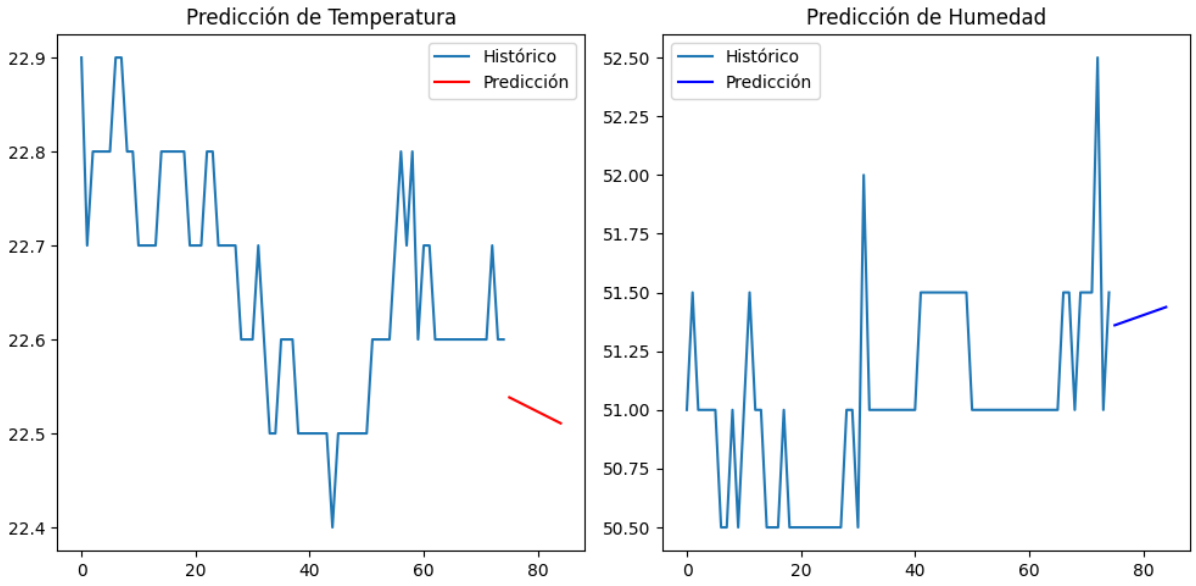
3. Predicción de Temperatura y Humedad (Regresión)

Se implementó un modelo de regresión lineal para predecir valores futuros de temperatura y humedad.

```python
# Preparar datos para el modelo de regresión lineal
X = np.arange(len(df)).reshape(-1, 1)
y_temp = df['uplink_message_decoded_payload_temperature_1'].values
y_hum = df['uplink_message_decoded_payload_relative_humidity_2'].values
```

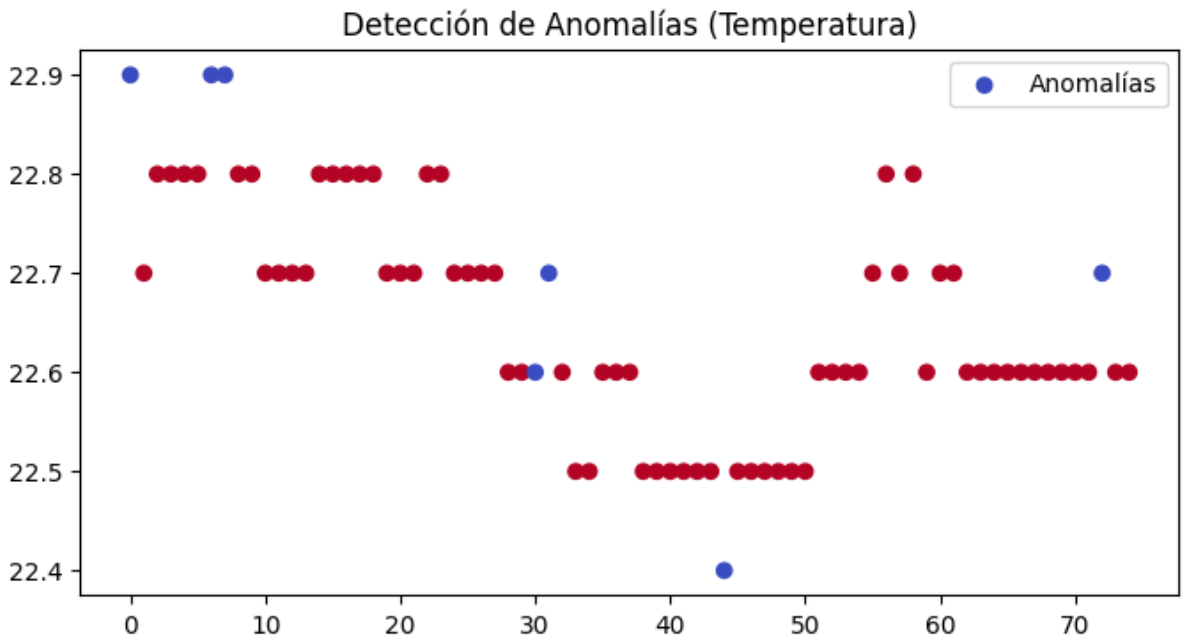
```python
# Modelos de regresión para temperatura y humedad
model_temp = LinearRegression().fit(X, y_temp)
model_hum = LinearRegression().fit(X, y_hum)

# Predicción para los próximos 10 pasos
future_steps = np.arange(len(df), len(df) + 10).reshape(-1, 1)
pred_temp = model_temp.predict(future_steps)
pred_hum = model_hum.predict(future_steps)

# Visualización de las predicciones
plt.figure(figsize=(10, 5))

# Temperatura
plt.subplot(1, 2, 1)
plt.plot(df['uplink_message_decoded_payload_temperature_1'], label='Histórico')
plt.plot(range(len(df), len(df) + 10), pred_temp, label='Predicción', color='red')
plt.title('Predicción de Temperatura')
plt.legend()

# Humedad
plt.subplot(1, 2, 2)
plt.plot(df['uplink_message_decoded_payload_relative_humidity_2'], label='Histórico')
plt.plot(range(len(df), len(df) + 10), pred_hum, label='Predicción', color='blue')
plt.title('Predicción de Humedad')
plt.legend()

plt.tight_layout()
plt.show()
```

Obtenemos los resultados:




```python
# Aplicar Isolation Forest para detección de anomalías
isolation_forest = IsolationForest(contamination=0.1)
df['anomaly'] = isolation_forest.fit_predict(df[['uplink_message_decoded_payload_temperature_1',
                                                 'uplink_message_decoded_payload_relative_humidity_2']])

# Visualizar anomalías
plt.figure(figsize=(8, 4))
plt.scatter(df.index, df['uplink_message_decoded_payload_temperature_1'],
            c=df['anomaly'], cmap='coolwarm', label='Anomalías')
plt.title('Detección de Anomalías (Temperatura)')
plt.legend()
plt.show()
```

Obtenemos los resultados:



Como observamos en el código, se definieron estados ambientales como "seco", "húmedo", "caluroso", y "templado", usando un modelo de clasificación basado en Árboles de Decisión. Se generaron gráficos para comparar los valores reales y las predicciones del modelo, así como para identificar visualmente las anomalías. Además, se resaltaron las anomalías en el gráfico.

4. **Evaluación del Modelo**
    - Métricas de regresión: MSE, R^2.
    - Detección de anomalías: porcentaje de falsos positivos y verdaderos positivos.
    - Clasificación: matriz de confusión para evaluar la precisión de los estados ambientales.

Este enfoque permitió:

- ✓ Predecir tendencias en la temperatura y humedad de forma precisa.
- ✓ Identificar anomalías que podrían afectar la operación del sistema.
- ✓ Clasificar el estado ambiental, optimizando la toma de decisiones en tiempo real.

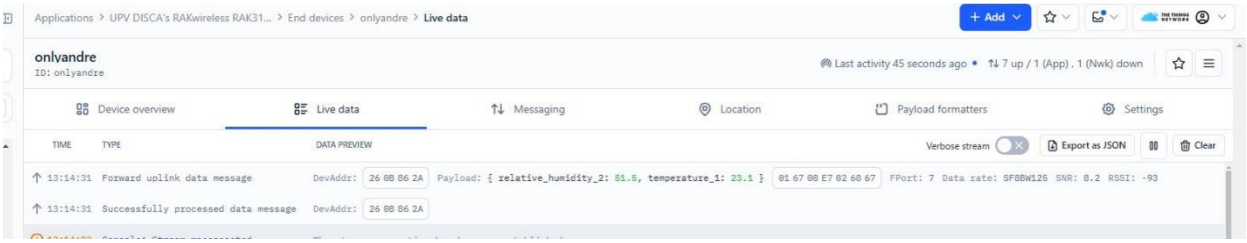
Resultados y Pruebas

Se realizaron diversas pruebas para validar el correcto funcionamiento del sistema. Se verificó la precisión de los datos registrados por los sensores de temperatura y humedad, la transmisión eficiente a través de LoRaWAN, y la integración con InfluxDB y Grafana para la visualización en tiempo real. Adicionalmente, se compararon los valores reales con los predichos por el modelo de Machine Learning implementado, evaluando su desempeño con métricas como MAE y RMSE.

Visualización en TTN:

La visualización en The Things Network (TTN) y el protocolo Cayenne Low Power Payload (LPP) permite monitorear en tiempo real los datos enviados por el nodo RAK3172 hacia la red LoRaWAN, proporcionando herramientas esenciales para la depuración, análisis y confirmación del flujo de datos dentro del sistema IoT del Proyecto Heri-Sure.

Ejemplo en LiveData:



Tablero de Visualización

El sistema de monitoreo desarrollado para el Proyecto Heri-Sure cuenta con un tablero de visualización en Grafana, el cual permite supervisar en tiempo real las variables de temperatura y humedad registradas por el sensor Milesight EM320-TH y transmitidas a través del nodo RAK3172.

Diseño del Dashboard

El tablero en Grafana se estructuró de manera que permita una interpretación clara y eficiente de los datos, incluyendo:

1. **Gráficos de tendencias**
    - Se diseñó un **gráfico de series temporales** que muestra la evolución de la temperatura y humedad en el tiempo.
    - Este gráfico permite detectar cambios en las condiciones ambientales y posibles anomalías.
2. **Indicadores de valor actual**
    - Se implementaron **gauge panels** (medidores tipo velocímetro) para mostrar los valores actuales de temperatura y humedad en una vista más intuitiva.
    - Estos indicadores permiten verificar rápidamente si los valores están dentro de los rangos esperados.
3. **Alertas y notificaciones**
    - Se configuraron alertas en Grafana para notificar **cuando la temperatura o la humedad superan umbrales críticos**.
    - Estas alertas se envían automáticamente a **Telegram** y también activan el **LED del nodo RAK3172** como señal visual de advertencia.

Integración con InfluxDB y MQTT

El tablero se alimenta de datos en tiempo real gracias a la integración con **InfluxDB** y **MQTT**, siguiendo este flujo:

1. **Recepción de datos en The Things Network (TTN)**
    - El nodo LoRaWAN transmite los datos del sensor a TTN.
2. **Reenvío de datos a un broker MQTT**
    - TTN envía los datos al broker MQTT configurado en Telegraf.

3. **Almacenamiento en InfluxDB**
    - Telegraf extrae los datos de MQTT y los almacena en la base de datos **proyecto_RIS_OYA** en InfluxDB.
4. **Visualización en Grafana**
    - Grafana consulta la base de datos y actualiza los paneles con los datos más recientes.

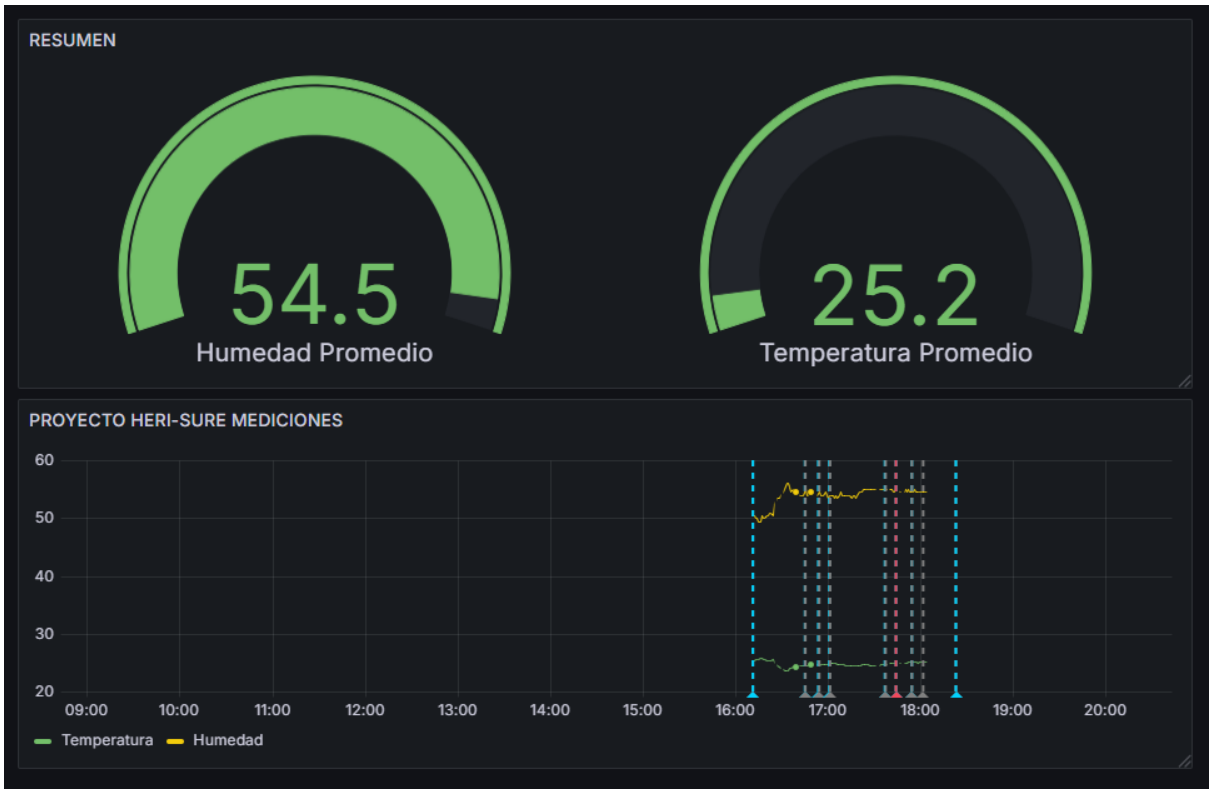
Evaluación y Optimización

El tablero ha sido probado con datos reales para garantizar su correcto funcionamiento. Se ajustaron los siguientes parámetros para mejorar su rendimiento:

- **Frecuencia de actualización**: Se estableció un intervalo óptimo para actualizar los gráficos sin afectar el rendimiento.
- **Escalabilidad**: Se diseñó la infraestructura de manera modular para permitir la integración de nuevos sensores y métricas en el futuro.
- **Optimización de consultas en InfluxDB**: Se aplicaron filtros para reducir la carga de consultas y mejorar la velocidad de respuesta en Grafana.

Tablero:


PROYECTO HERI-SURE MEDICIONES
60
58
56
54
52
50
48
46
44
42
40
38
36
34
32
30
28
26
24
22
20
14:45
15:00
15:15
15:30
15:45
16:00
16:15
16:30
16:45
17:00
17:15
17:30
17:45
18:00
18:15
18:30
Temperatura
Humedad


RESUMEN
54.5
Humedad Promedio
25.2
Temperatura Promedio
PROYECTO HERI-SURE MEDICIONES
60
50
40
30
20
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
Temperatura
Humedad

Alertas:

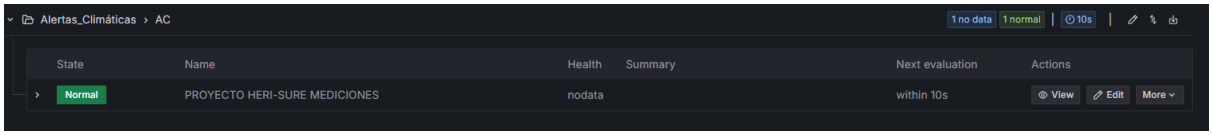
La implementación de alertas en el Proyecto Heri-Sure es un componente esencial para la monitorización en tiempo real de las condiciones ambientales, permitiendo la activación de notificaciones y acciones automáticas cuando se superan los valores críticos de temperatura y humedad. Se han configurado alertas tanto en Grafana como en Telegram, además de una respuesta física mediante el encendido del LED en el nodo RAK3172.

Configuración de Alertas en Grafana

Para la activación de alertas en **Grafana**, se establecieron umbrales en la base de datos **InfluxDB**, generando condiciones que se evalúan periódicamente. Los pasos seguidos para la configuración de estas alertas fueron:

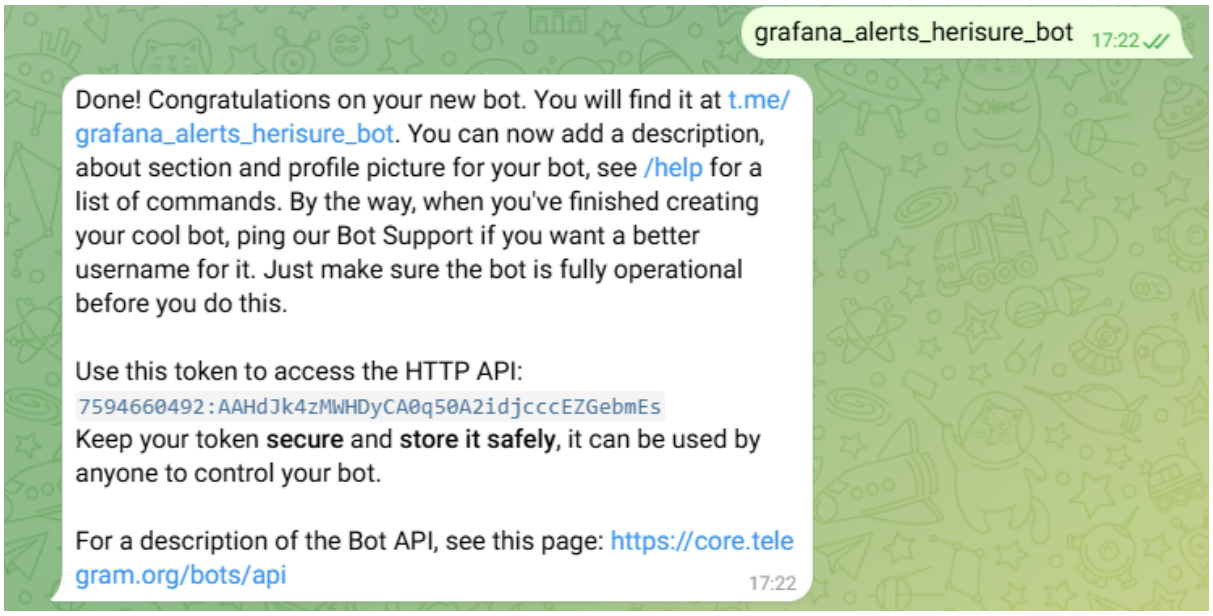
1. **Definir las métricas a monitorear**: Se utilizaron las variables de **temperatura y humedad** provenientes de **InfluxDB**.
2. **Configurar los umbrales**: Se definieron tres tipos de alerta:
    - Cuando la **temperatura** supera los **30°C**.
    - Cuando la **temperatura** desciende por debajo de **20°C**.
    - Cuando la **humedad** supera el **80%**.
3. **Expresiones de alerta**:
    - Se utilizó la función **Reduce (Last, Strict)** para evaluar la última medición registrada.
    - Se aplicaron condiciones de **Threshold (IS ABOVE o IS BELOW)** en función de los valores críticos establecidos.
4. **Configuración de la notificación**:
    - Se vinculó la alerta con **Telegram** como canal de notificación.
    - Se configuró el bot para enviar mensajes cuando las condiciones de temperatura o humedad activan la alerta.

- Se definió un tiempo de evaluación de **1 minuto** para garantizar una respuesta rápida.



Configuración de Alertas en Telegram

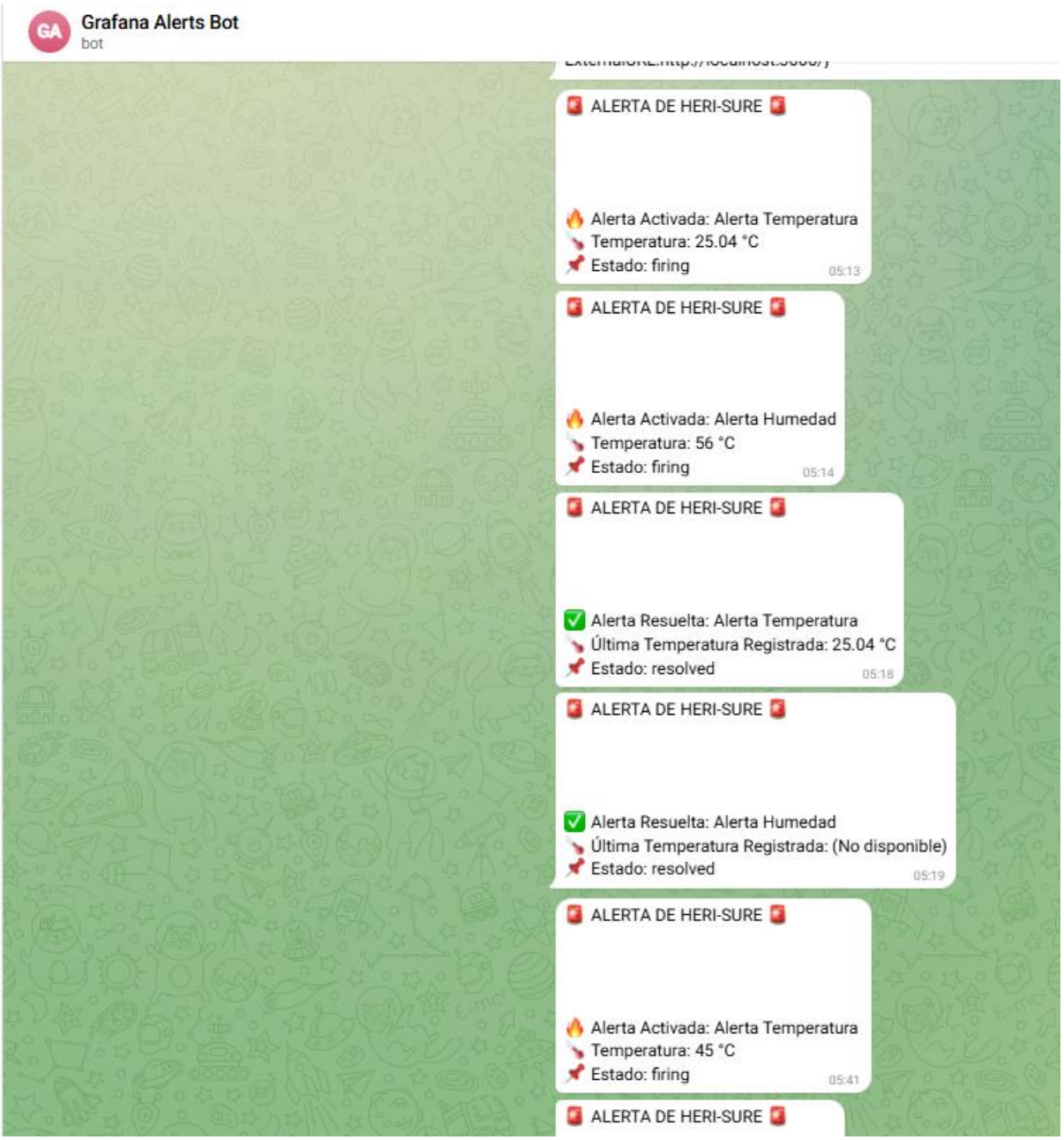
Para recibir alertas en tiempo real, se creó un bot en **Telegram** utilizando BotFather. Los pasos seguidos fueron los siguientes:



BotFather proporcionó un Token de acceso, el cual se utilizó para conectar el bot con Grafana.

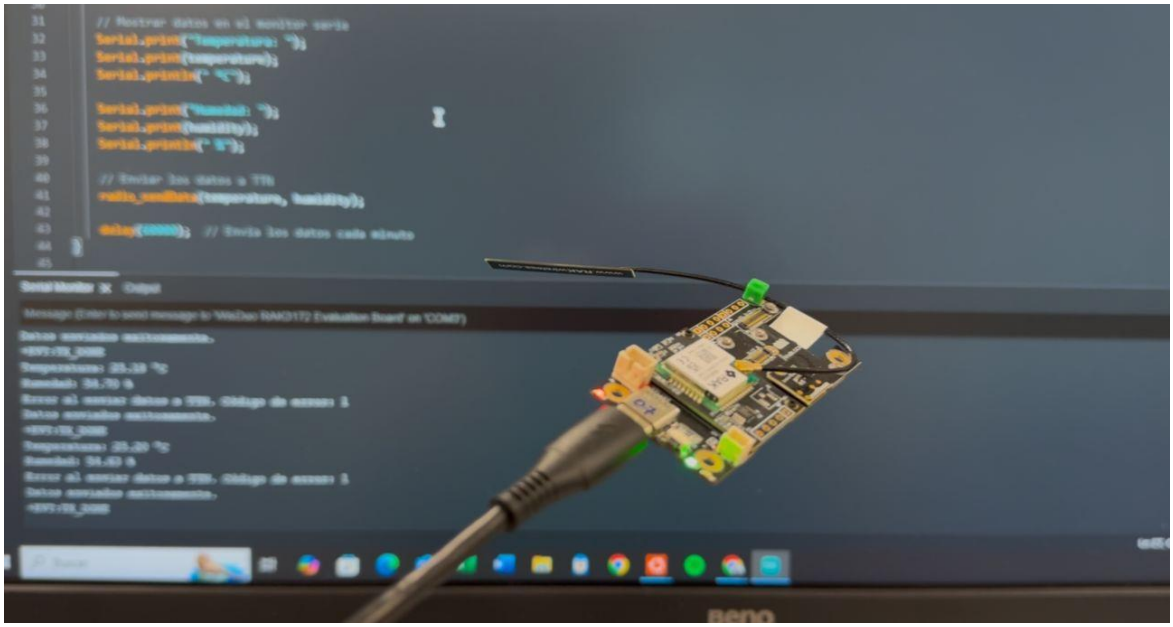
- En Grafana → Alerting → Contact Points, se añadió un nuevo canal de notificación tipo Telegram.
- Se ingresó el Token del bot y el Chat ID del grupo o usuario receptor.
- Se probó la integración enviando un mensaje de prueba.

Cada vez que una alerta se activaba en Grafana, el bot enviaba un mensaje de notificación a Telegram con el siguiente formato:



Alerta con el LED en el Nodo RAK3172

Además de las notificaciones en Telegram, se implementó una respuesta física en el RAK3172, donde el LED se enciende cuando se superan los umbrales de temperatura o humedad. Para lograr esto, se modificó el código en Arduino IDE, integrando la lógica de encendido del LED en la función led_control().



Resultados de las Alertas Implementadas

- Grafana permite visualizar las alertas activadas en tiempo real en el dashboard.
- Telegram envía notificaciones instantáneas cada vez que una alerta se dispara.
- El LED del RAK3172 responde físicamente a las condiciones críticas, proporcionando una señal visual del estado del ambiente.

Estas alertas garantizan que los usuarios sean informados de cambios significativos en las condiciones ambientales y puedan tomar medidas correctivas en tiempo real.

Desafíos del proyecto

El desarrollo del Proyecto Heri-Sure ha sido un proceso lleno de retos técnicos, logísticos y de integración que pusieron a prueba nuestra capacidad de resolución de problemas. A pesar de contar con conocimientos en hardware, software y redes de sensores, la implementación de un sistema IoT completo con visualización, alertas y predicción de datos en tiempo real requirió múltiples iteraciones, depuración de errores y adaptaciones en el camino.

Uno de los principales desafíos fue la configuración y comunicación de los dispositivos LoRaWAN. La conexión del nodo RAK3172 con The Things Network (TTN) presentó dificultades, desde la correcta configuración de la codificación de los datos hasta la validación de los mensajes recibidos en MQTT. En múltiples ocasiones, los datos no llegaban correctamente, lo que nos obligó a depurar paso a paso cada componente del sistema.

La integración con InfluxDB y su posterior visualización en Grafana también supuso un reto. Encontramos problemas al intentar realizar consultas sobre los datos almacenados, ajustar formatos de tiempo y estructurar dashboards comprensibles. Adicionalmente, la configuración de alertas en Grafana y su vinculación con Telegram para notificaciones automáticas exigió un entendimiento profundo de la arquitectura de la plataforma.

Otro aspecto desafiante fue la implementación de la **pantalla** y el **modelo de Machine Learning** en el proyecto. Si bien la idea inicial era realizar solo la adquisición de datos y visualización, decidimos dar un paso más allá e incorporar estos **extras**. Este proceso implicó trabajar un poco mas la parte autodidacta, tanto por la falta de información de la pantalla como comentamos antes, así como trabajar con modelos de series temporales en Python para el modelo ML.

A pesar de todos estos desafíos, la satisfacción de ver un sistema funcional y completamente integrado compensó cada hora invertida en depuración y ajustes.

Conclusiones y Recomendaciones

El Proyecto Heri-Sure ha demostrado la viabilidad de un sistema IoT basado en LoRaWAN para la monitorización de variables ambientales, integración con bases de datos en la nube y activación de alertas en tiempo real. A lo largo del desarrollo, se logró:

- ✓ Establecer una comunicación estable y confiable entre el sensor RAK3172 y The Things Network, permitiendo el envío continuo de datos de temperatura y humedad.
- ✓ Integrar la base de datos InfluxDB como repositorio de los datos sensoriales, facilitando su almacenamiento y consulta.
- ✓ Diseñar un dashboard en Grafana que permite la visualización en tiempo real de los datos recopilados, junto con alertas configurables según umbrales de temperatura y humedad.
- ✓ Implementar Machine Learning para la predicción de tendencias y la detección de valores atípicos en los datos recolectados.
- ✓ Automatizar las alertas y respuestas físicas, enviando notificaciones a Telegram y activando el LED del sensor cuando se superan los límites predefinidos.

A pesar de los desafíos encontrados, el proyecto ha cumplido con sus objetivos principales y demuestra cómo las tecnologías IoT pueden proporcionar soluciones eficaces para el monitoreo y control de variables ambientales.

Para futuras mejoras y expansiones del sistema, se sugieren las siguientes recomendaciones:

1. **Optimizar la configuración del sensor y la transmisión LoRaWAN**
    - Evaluar diferentes configuraciones de **spreading factor y data rate** para mejorar la eficiencia en el envío de datos.
    - Implementar mecanismos de retransmisión en caso de pérdida de paquetes.
2. **Explorar otras formas de respuesta física en el nodo RAK3172**

- Además del LED, integrar otros actuadores como **pantallas de tinta electrónica o bocinas.**

3. **Escalar el sistema a múltiples sensores y ubicaciones**
    - Ampliar la infraestructura para monitorear varias áreas en simultáneo.
    - Implementar lógica de **geolocalización y mapeo en tiempo real** de los dispositivos conectados.

En conclusión, este proyecto ha sentado las bases para un **sistema IoT robusto, modular y escalable**, con amplias posibilidades de crecimiento y aplicación en múltiples sectores.

Referencias Bibliográficas

- Repositorio de código: https://github.com/nipsn/HeriSure-RIS2024
- https://docs.rakwireless.com/Product-Categories/WisBlock/RAK14000/Quickstart/
- https://github.com/RAKWireless/WisBlock/tree/master/examples/common/IO/RAK14000-Epaper-TriColor
- https://github.com/RAKWireless/RUI3-Best-Practice/tree/main/RUI3-Sensor-Node-EPD-2.13