IoT-BASED ENVIRONMENTAL MONITORING SYSTEM USING RASPBERRY PI AND DOCKER

A PROJECT REPORT SUBMITTED TO
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS

BY

SHANTHOSH KANNAN
REG. NO. DA2332306010007

UNDER THE GUIDANCE OF
Dr. A. THIRUMURTHI RAJA M.C.A., M.E., PH.D.,
ASSISTANT PROFESSOR,
SRMIST-DDE,

DEPARTMENT OF COMPUTER APPLICATIONS
DIRECTORATE OF DISTANCE EDUCATION
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be university u/s 3 of UGC Act, 1956)
Kattankulathur – 603 203
Chennai, Tamil Nadu
NOV 2024

BONAFIDE CERTIFICATE

This is to certify that the project report titled **"IoT-BASED ENVIRONMENTAL MONITORING SYSTEM USING RASPBERRY PI AND DOCKER"** is a bonafide work carried out by **SHANTHOSH KANNAN (DA2332306010007)** under my supervision for the award of the Degree of Master of Computer Applications. To my knowledge the work reported herein is the original work done by these students.

PROJECT GUIDE
HEAD OF DEPARTMENT
Dr.A.Thirumurthi Raja
Assistant Professor,
SRMIST-DDE,

HEAD OF DEPARTMENT
Dr.A.Thirumurthi Raja
Program Coordinator
Department of Computer Applications,
Directorate of Distance Education(DDE)
SRMIST,

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

With profound gratitude to the ALMIGHTY, I take this chance to thank the people who helped me to complete this project.

We take this as a right opportunity to say THANKS to my parents who are there to stand with me always with the words “YOU CAN”.

We are thankful to **Dr.T.R. Paarivendhar, Chancellor**, SRM Institute of Science & Technology who gave us the platform to establish me to reach greater heights.

I express my deep sense of gratitude to the Director **Dr. Manoranjan Pon Ram** for their wholehearted support and encouragement.

I earnestly thank **Dr.A.Thirumurthi Raja** , Programme Coordinator, CA/MCA- Directorate of Distance Education, SRM Institute of Science & Technology who always encourage us to do novel things.

It is our delight to thank our project guide **Dr.A.Thirumurthi Raja** Assistant Professor, Department of Computer Applications(DDE), for his help, support, encouragement, suggestions, and guidance throughout the development phases of the project.

We convey our gratitude to all the faculty members of the department who extended their support through valuable comments and suggestions during the reviews.

A great note of gratitude to friends and people who are known and unknown to me who helped in carrying out this project work a successful one.

SHANTHOSH KANNAN

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The increasing concerns regarding environmental changes and their impacts on health necessitate effective monitoring systems. This project presents an IoT-Based Environmental Monitoring System using a Raspberry Pi 5 and Raspberry Pi Pico W, integrated with a DHT11 sensor to measure temperature and humidity. The system employs MicroPython for sensor data acquisition and utilizes the MQTT protocol for efficient data transmission. Leveraging Docker, the project orchestrates a suite of services, including Eclipse Mosquitto for messaging, InfluxDB for time-series data storage, and Grafana for real-time visualization.

The architecture allows for modular deployment of components, enhancing scalability and maintainability. By utilizing Node-RED, the system facilitates the creation of dynamic data flows, enabling seamless integration and data management. The collected environmental data can be visualized through intuitive dashboards, allowing users to monitor real-time conditions and trends. This project demonstrates a practical approach to environmental monitoring, emphasizing the role of IoT in addressing modern challenges related to climate and public health. The system serves as a foundational platform for further research and development in smart environmental applications, contributing to efforts in sustainable development and disaster prevention.

CHAPTER 1

INTRODUCTION

In an era marked by rapid industrialization and urbanization, the degradation of the environment poses significant risks to public health and ecological stability. The ability to monitor environmental parameters in real time has become critical for informed decision-making and proactive interventions. This project introduces an IoT-Based Environmental Monitoring System designed to track vital atmospheric conditions, specifically temperature and humidity, using a combination of cutting-edge hardware and software technologies.

The system employs a Raspberry Pi 5 as the central processing unit, providing a robust platform for running various applications in a containerized environment via Docker. Complementing the Raspberry Pi is the Raspberry Pi Pico W, which interfaces with a DHT11 sensor to gather environmental data. The choice of the DHT11 sensor is based on its affordability and reliability for basic temperature and humidity measurements, making it suitable for various applications, including smart agriculture, indoor climate control, and health monitoring.

To facilitate effective data transmission and storage, the system utilizes the MQTT protocol, a lightweight messaging protocol ideal for IoT applications. This enables the Pico W to efficiently send sensor data to the Raspberry Pi, where it is processed and stored in InfluxDB, a time-series database optimized for handling vast amounts of time-stamped data. Furthermore, Grafana is integrated to create dynamic, interactive dashboards, allowing users to visualize the collected data in real time.

Additionally, Node-RED is employed as a flow-based programming tool, enabling users to design complex data processing workflows without requiring extensive programming knowledge. This enhances the usability of the system, making it accessible to a broader audience, including researchers, educators, and hobbyists.

By leveraging the principles of IoT, this project not only provides a practical solution for environmental monitoring but also serves as an educational platform for understanding the complexities of sensor integration, data management, and real-time visualization. As climate change and environmental degradation continue to challenge communities worldwide, this IoT-based system contributes to the broader effort of developing smart, responsive technologies that promote sustainability and improve quality of life.

CHAPTER 2

SOFTWARE REQUIREMENT ANALYSIS

1. Functional Requirements:

- **Data Acquisition:** The system should collect temperature and humidity data from the DHT11 sensor connected to the Raspberry Pi Pico W.
- **Data Transmission:** The collected data should be transmitted to the Raspberry Pi using the MQTT protocol.
- **Data Storage:** The system should store the transmitted data in Influx-DB for historical analysis.
- **Data Visualization:** The system should visualize real-time data using Grafana dashboards.
- **User Interface:** A user-friendly interface (via Node-RED) should be provided for monitoring and managing data flows.

2. Non-Functional Requirements:

- **Scalability:** The system should accommodate additional sensors and devices in the future.
- **Performance:** The data transmission and visualization should be real-time, with minimal latency.
- **Reliability:** The system should maintain consistent performance without failure.
- **Usability:** The interface should be intuitive and easy to use for users without extensive technical knowledge.

3. Constraints:

- The system must operate within the limitations of the Raspberry Pi and Pico W regarding processing power and memory.
- Internet connectivity may be required for certain functionalities (e.g., accessing Grafana).

2.1 HARDWARE SPECIFICATION

2.1.1. RASPBERRY PI 5

- **Processor:** Quad-core ARM Cortex-A76, 64-bit
- **RAM:** 4 GB or 8 GB LPDDR4-3200 SDRAM
- **Connectivity:** Gigabit Ethernet, Wireless LAN (802.11ac), Bluetooth 5.0
- **Ports:** USB 3.0, USB 2.0, HDMI, GPIO pins, Power input
- **Power Supply:** 27W power supply (5V/3A)

2.1.2. RASPBERRY PI PICO W

- **Processor:** Dual-core ARM Cortex-M0+
- **Memory:** 264 KB SRAM, 2 MB Flash
- **Connectivity:** Wi-Fi (802.11n), Bluetooth (optional)
- **Power Supply:** 5W power supply or powered through a micro USB cable

2.1.3. DHT11 SENSOR

- **Measurement Range:** Temperature: 0 to 50°C, Humidity: 20% to 90%
- **Accuracy:** ±2°C for temperature, ±5% for humidity
- **Interface:** Digital output

2.1.4. OTHER COMPONENTS

- **Breadboard:** For connecting the Pico W and DHT11 sensor.
- **Jumper Wires:** For making connections between the Pico W, DHT11, and breadboard.

2.2 SOFTWARE SPECIFICATION

2.2.1. OPERATING SYSTEM

- **Raspberry Pi OS (or any compatible Linux-based OS)**: To run Docker and other applications.

2.2.2. REQUIRED SOFTWARE TOOLS

- **MicroPython:** For programming the Raspberry Pi Pico W to read sensor data.
- **Python:** For backend processing and communication management on the Raspberry Pi 5.
- **Docker:** To containerize and manage the various services (Node-RED, InfluxDB, Grafana, and MQTT broker).
- **Node-RED:** For creating data flow management between devices and services.
- **Eclipse Mosquitto MQTT:** For messaging protocol to transmit data between Pico W and Raspberry Pi.
- **InfluxDB:** For storing time-series data.
- **Grafana:** For visualizing data in a user-friendly dashboard.

2.2.3. SOFTWARE REQUIREMENTS

- **MicroPython** version compatible with Raspberry Pi Pico W.
- **Docker** and **Docker Compose** installed on Raspberry Pi 5.
- **Node-RED**, **Mosquitto**, **InfluxDB**, and **Grafana** configured via Docker.

2.3 ABOUT THE SOFTWARE AND ITS FEATURE

This **IoT-Based Environmental Monitoring System** aims to provide an innovative solution for real-time monitoring of environmental parameters such as temperature and humidity. Leveraging the capabilities of **Raspberry Pi 5** and **Raspberry Pi Pico W**, this project demonstrates the integration of hardware and software to create a cohesive IoT ecosystem.

2.3.1 KEY FEATURES

- **Real-Time Data Monitoring:** The system collects and displays real-time environmental data, enabling users to monitor conditions dynamically.
- **Modular Architecture:** The use of Docker allows for a modular setup, making it easy to add or modify components without disrupting the entire system.
- **Interactive Visualization:** With Grafana, users can create custom dashboards to visualize historical and real-time data trends, making data interpretation straightforward and effective.
- **User-Friendly Interface:** Node-RED offers an intuitive interface for users to create data flows and manage system components, reducing the need for advanced programming skills.
- **Scalability:** The system is designed to be scalable, allowing for the integration of additional sensors or devices as needed, accommodating future expansions.
- **Data Storage and Analysis:** InfluxDB serves as a robust backend for storing time-series data, enabling users to conduct further analysis on historical trends and patterns.
- **Notifications and Alerts:** Users can configure alerts for specific conditions (e.g., high temperature or humidity) using Node-RED to receive notifications via email or SMS.
- **Energy Efficiency:** The system is designed to operate efficiently with low power consumption, making it suitable for long-term deployment in various environments.

This project not only addresses the critical need for environmental monitoring but also serves as a foundational platform for further research and innovation in smart IoT applications. By combining various technologies, it exemplifies the potential of IoT in enhancing our understanding and management of environmental conditions.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

3.1.1. OVERVIEW OF EXISTING SOLUTIONS

Traditional environmental monitoring systems often rely on standalone sensors and manual data collection methods. These systems may lack real-time data transmission, efficient data management, and user-friendly interfaces.

3.1.2. LIMITATIONS OF EXISTING SYSTEMS

- **Manual Data Collection:** Data must be recorded manually, leading to potential human error and delays in response.
- **Limited Data Visualization:** Many systems do not provide effective visualization tools, making it difficult for users to interpret data trends.
- **Lack of Scalability:** Existing systems may be rigid, making it challenging to integrate new sensors or functionalities.
- **High Cost:** Standalone solutions can be expensive, with high maintenance costs and limited accessibility for smaller organizations or individuals.

3.1.3. USE CASES

- Standalone temperature and humidity sensors with basic data logging capabilities.
- Commercial monitoring systems that require expensive subscriptions and proprietary hardware.

3.2 PROPOSED SYSTEM

3.2.1. OVERVIEW OF THE PROPOSED SOLUTION

The proposed IoT-Based Environmental Monitoring System integrates various components to create a cohesive solution for real-time environmental monitoring. It utilizes the Raspberry Pi ecosystem to gather, process, and visualize data in a user-friendly manner.

3.2.2. KEY FEATURES OF THE PROPOSED SYSTEM

- **Real-Time Data Acquisition:** Automatic collection of data from the DHT11 sensor using Raspberry Pi Pico W.
- **Efficient Data Transmission:** Utilization of MQTT for low-latency communication between devices.
- **Scalable Architecture:** Modular design allowing for easy integration of additional sensors and functionalities.
- **Dynamic Visualization:** Real-time dashboards created with Grafana for monitoring trends and conditions.
- **Alerts and Notifications:** Customizable alerts for specific environmental conditions.

3.2.3. SYSTEM ARCHITECTURE

The system architecture consists of:

- **Sensor Layer:** DHT11 sensor connected to the Raspberry Pi Pico W for data collection.
- **Communication Layer:** MQTT broker (Eclipse Mosquitto) for transmitting data to the Raspberry Pi 5.
- **Processing Layer:** Raspberry Pi 5 running Docker containers for data storage (InfluxDB) and visualization (Grafana).
- **User Interface Layer:** Node-RED for managing data flows and configuring user interfaces.

3.3 FEASIBILITY STUDY

3.3.1. TECHNICAL FEASIBILITY

- The proposed system is technically feasible as it utilizes widely available hardware and open-source software tools. The integration of these components is well-documented, allowing for straightforward implementation.
- The Raspberry Pi platform provides sufficient processing power and connectivity options to support the required applications.

3.3.2. ECONOMIC FEASIBILITY

- The project is cost-effective, with minimal investment required for hardware (Raspberry Pi, DHT11 sensor, etc.) and software (open-source tools). The potential return on investment includes reduced costs associated with manual monitoring and improved decision-making based on real-time data.

3.3.3. OPERATIONAL FEASIBILITY

- The system is designed to be user-friendly, making it accessible to individuals with limited technical expertise.
- Training may be required for users to fully leverage the system's capabilities, particularly in configuring alerts and using the visualization tools.

3.3.4. RISK ANALYSIS

- **Technical Risks:** Integration challenges may arise due to hardware compatibility or software dependencies and mitigated through testing and reliance on good tools.
- **Operational Risks:** Users may require training to effectively use the system, which can delay adoption.
- **Market Risks:** While the demand for environmental monitoring solutions is growing, competition from existing solutions may pose a challenge.

4. SYSTEM DESIGN

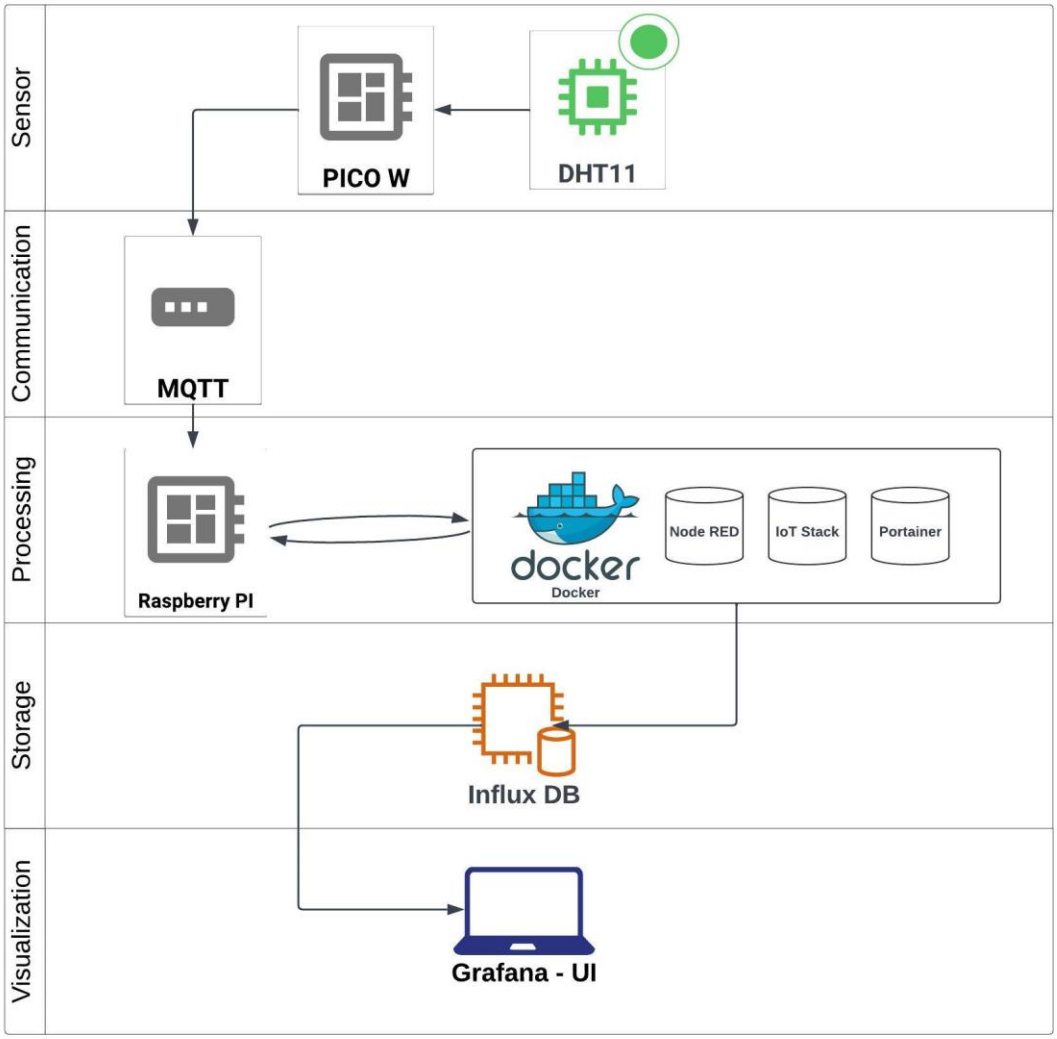

Figure 4.1 System Design

4.1 DATA FLOW DIAGRAM

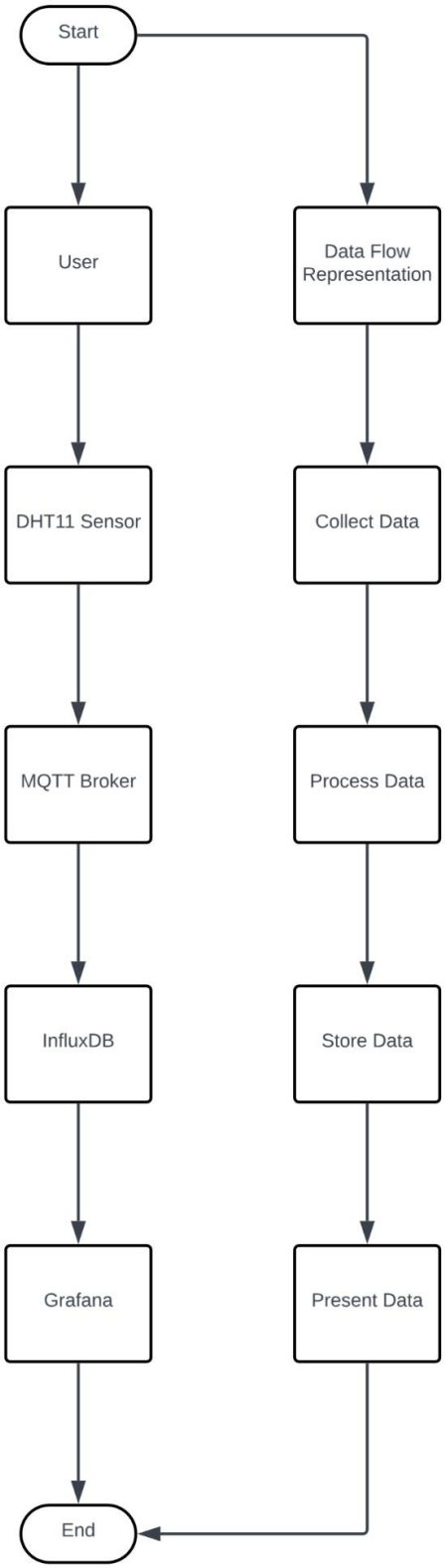

Figure 4.2 Data Flow Diagram

CHAPTER 4

UML DIAGRAMS

4.1 USE CASE DIAGRAM

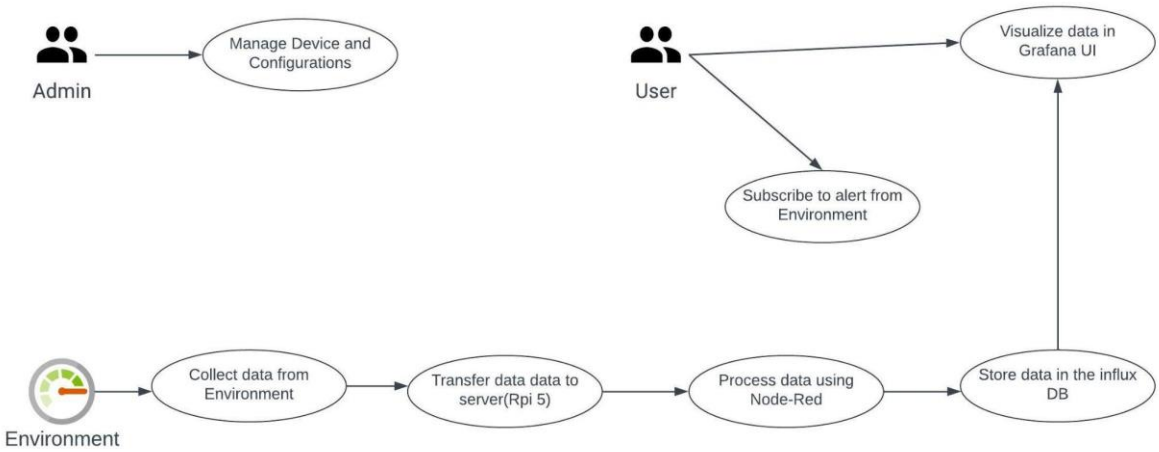

Figure 4. 1 Use Case Diagram

4.2 ACTIVITY DIAGRAM

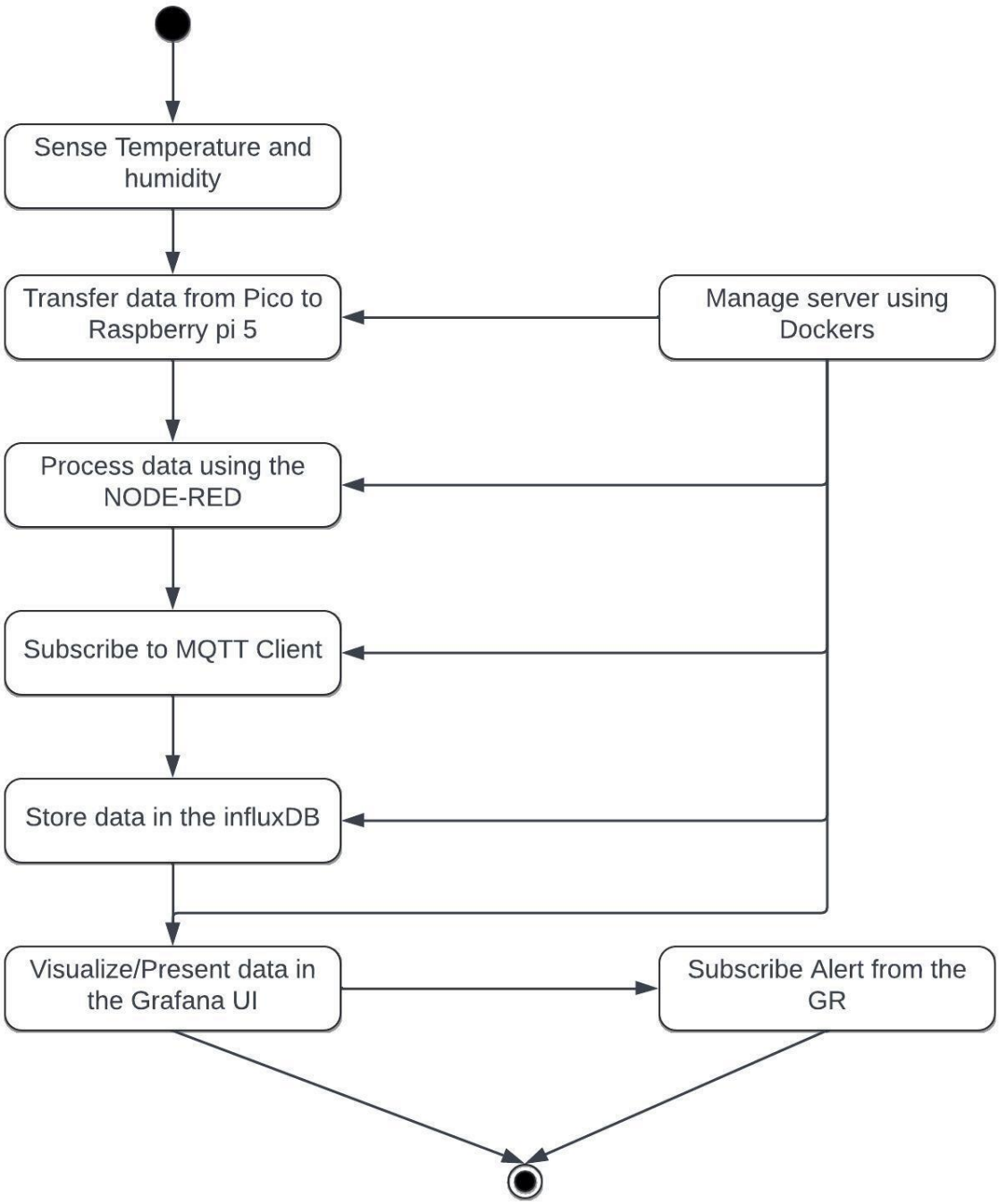

Figure 4. 2 Activity Diagram

4.3 SEQUENCE DIAGRAM

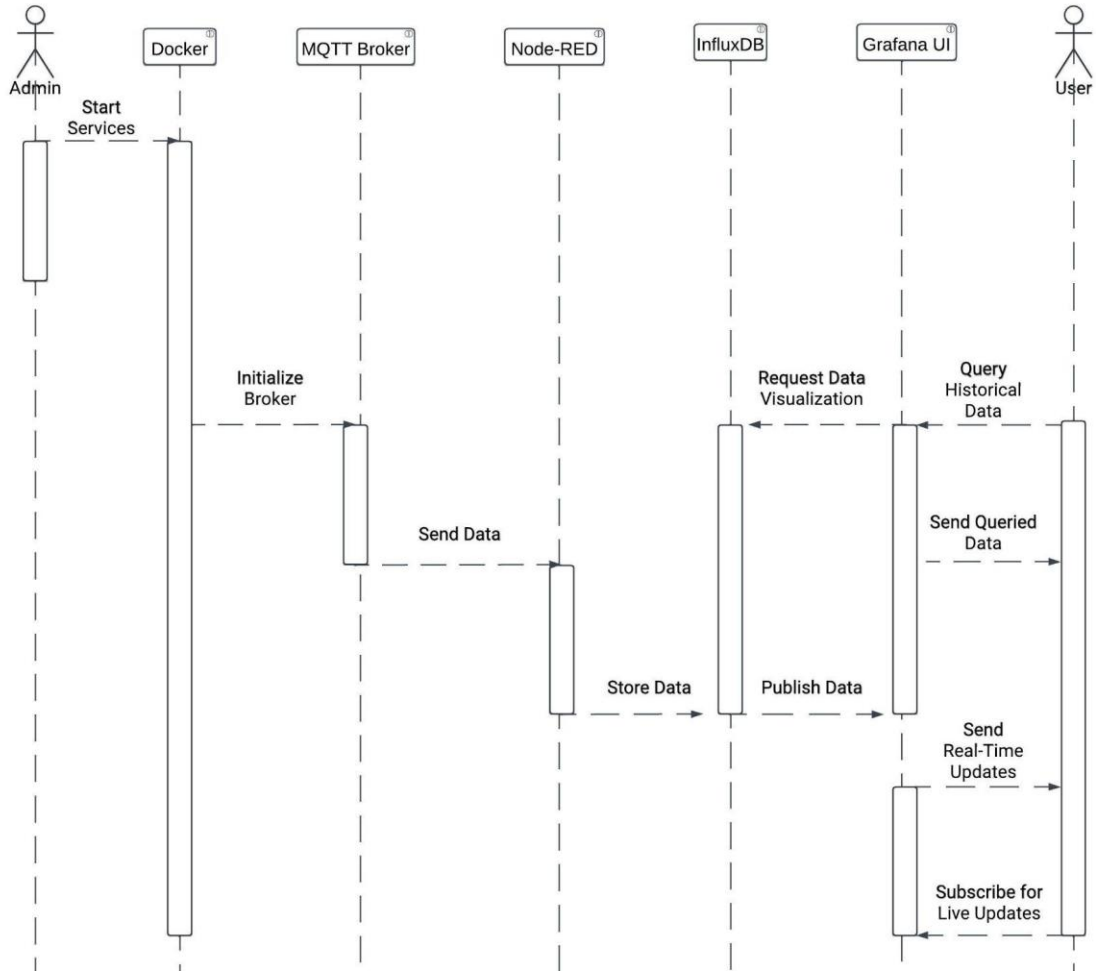

Figure 4. 3 Sequence Diagram

4.4 DATABASE DESIGN

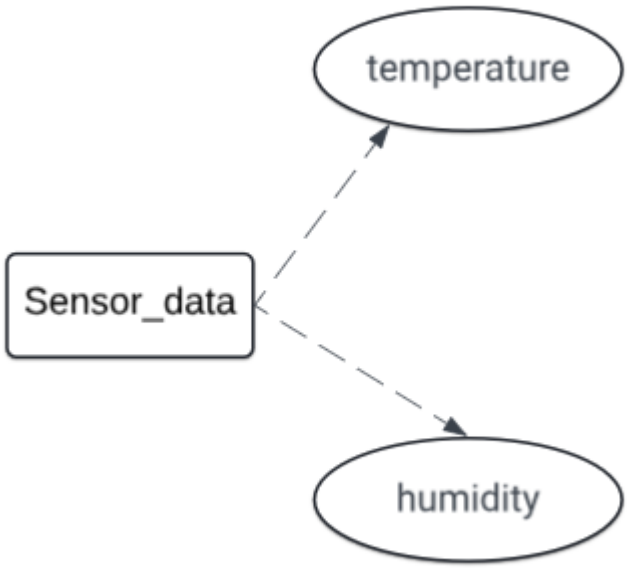

Figure 4. 4 ER Diagram

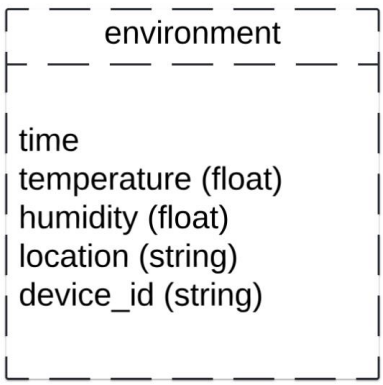

Figure 4. 5 Database Diagram

4.5 CLASS DESIGN

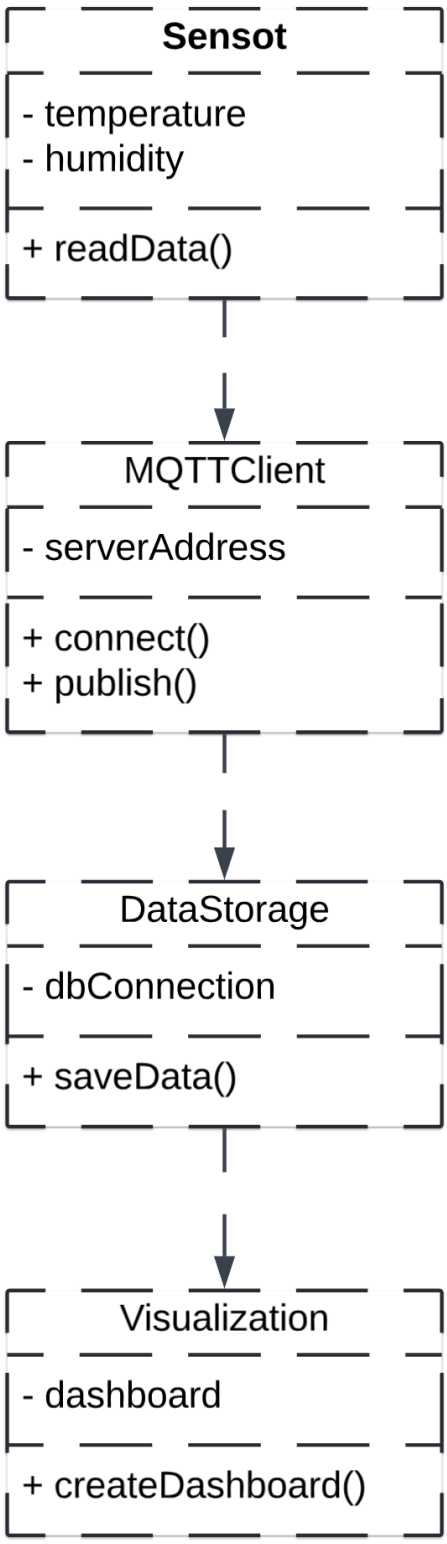

Figure 4. 6 Class diagram

CHAPTER 5

CODE TEMPLATES

5.1 MODULE DESCRIPTION

5.1.1 MODULES OVERVIEW

The project is divided into distinct modules, each handling a specific functionality.

5.1.2 SENSOR MODULE

Responsible for reading data from the DHT11 sensor and preparing it for transmission.

5.1.2.1 CODE TEMPLATE

```
#This file should be placed in the pico w with name "main.py" to auto start.

import network

import time

import dht

import machine

from umqtt.simple import MQTTClient  # This should now work without error

# DHT11 setup

sensor = dht.DHT11(machine.Pin(15))  # Use GPIO 15 for data pin

# LED setup (built-in LED on GPIO 25)

led = machine.Pin("LED", machine.Pin.OUT)

# Wi-Fi setup //both your server and the pico w should be connected in the same network.

ssid = 'YourWifiName'

password = 'YourWifiPassword'

# MQTT setup
```

```
mqtt_server = '192.168.31.183'  # Replace with your Raspberry Pi 5 IP
client_id = 'pico_w_dht11'
topic_pub_temp = 'sensor/dht11/temperature'
topic_pub_humidity = 'sensor/dht11/humidity'

# Connect to Wi-Fi
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)

    while wlan.isconnected() == False:
        print('Connecting to Wi-Fi...')
        time.sleep(1)

    print('Connected to Wi-Fi')
    print(wlan.ifconfig())

# MQTT publish function
def mqtt_connect():
    client = MQTTClient(client_id, mqtt_server, port=1883)
    client.connect()
    print(f'Connected to {mqtt_server} MQTT broker')
```

```
    return client

# Blink the LED
def blink_led(dur_ms):
    led.on()  # Turn on the LED
    time.sleep(dur_ms)  # Keep it on for 'dur_ms' milliseconds
    led.off()  # Turn off the LED

# Read DHT11 and publish data
def read_and_publish(client):
    while True:
        try:
            sensor.measure()
            temp = sensor.temperature()
            humidity = sensor.humidity()

            # Publish to MQTT broker
            client.publish(topic_pub_temp, str(temp))
            client.publish(topic_pub_humidity, str(humidity))

            print(f'Temperature: {temp}Â°C, Humidity: {humidity}%')

            # Blink LED to indicate successful data publish
```

```
            blink_led(0.2)  # Blink for 200ms to show successful publish
            #time.sleep(1)

        except OSError as e:
            print(f'Failed to read sensor: {e}')

        time.sleep(10)  # Adjust delay for how often to send data

# Main execution
try:
    connect_wifi()
    client = mqtt_connect()
    read_and_publish(client)
except KeyboardInterrupt:
    print('Script interrupted')
```

5.1.3 DOCKER MODULE

Handles docker container creation and services ports, volumes with the file name: docker-compose.yml.

5.1.3.1 CODE TEMPLATE

```
networks:
  default:
    driver: bridge
    ipam:
      driver: default
  nextcloud:
```

```
    driver: bridge
    internal: true
    ipam:
      driver: default

services:
  grafana:
    container_name: grafana
    image: grafana/grafana
    restart: unless-stopped
    user: "0"
    ports:
    - "3000:3000"
    environment:
    - TZ=Etc/UTC
    - GF_PATHS_DATA=/var/lib/grafana
    - GF_PATHS_LOGS=/var/log/grafana
    volumes:
    - ./volumes/grafana/data:/var/lib/grafana
    - ./volumes/grafana/log:/var/log/grafana
    healthcheck:
      test: ["CMD", "wget", "-O", "/dev/null", "http://localhost:3000"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 30s

  mosquitto:
    container_name: mosquitto
    build:
      context: ./.templates/mosquitto/.
      args:
      - MOSQUITTO_BASE=eclipse-mosquitto:latest
```

```
  restart: unless-stopped
  environment:
  - TZ=${TZ:-Etc/UTC}
  ports:
  - "1883:1883"
  - "9001:9001"
  volumes:
  - ./volumes/mosquitto/config:/mosquitto/config
  - ./volumes/mosquitto/data:/mosquitto/data
  - ./volumes/mosquitto/log:/mosquitto/log
  - ./volumes/mosquitto/pwfile:/mosquitto/pwfile

nodered:
  container_name: nodered
  build:
    context: ./services/nodered/.
    args:
    - DOCKERHUB_TAG=latest
    - EXTRA_PACKAGES=
  restart: unless-stopped
  user: "0"
  environment:
  - TZ=${TZ:-Etc/UTC}
  ports:
  - "1880:1880"
  volumes:
  - ./volumes/nodered/data:/data
  - ./volumes/nodered/ssh:/root/.ssh
portainer-ce:
  container_name: portainer-ce
  image: portainer/portainer-ce
  restart: unless-stopped
  ports:
```

```
  - "8000:8000"
  - "9000:9000"
  # HTTPS
  - "9443:9443"
  volumes:
  - /var/run/docker.sock:/var/run/docker.sock
  - ./volumes/portainer-ce/data:/data

influxdb2:
  container_name: influxdb2
  image: "influxdb:latest"
  restart: unless-stopped
  environment:
  - TZ=Etc/UTC
  - DOCKER_INFLUXDB_INIT_USERNAME=me
  - DOCKER_INFLUXDB_INIT_PASSWORD=mypassword
  - DOCKER_INFLUXDB_INIT_ORG=myorg
  - DOCKER_INFLUXDB_INIT_BUCKET=mybucket
  - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=my-super-secret-auth-token
  - DOCKER_INFLUXDB_INIT_MODE=setup
# - DOCKER_INFLUXDB_INIT_MODE=upgrade
  ports:
  - "8087:8086"
  volumes:
  - ./volumes/influxdb2/data:/var/lib/influxdb2
  - ./volumes/influxdb2/config:/etc/influxdb2
  - ./volumes/influxdb2/backup:/var/lib/backup
# - ./volumes/influxdb.migrate/data:/var/lib/influxdb:ro
  healthcheck:
    test: ["CMD", "influx", "ping"]
    interval: 30s
    timeout: 10s
    retries: 3
```

start_period: 30s

5.1.4 RUN SCRIPT

To start the services in the linux booting.

5.1.4.1 CODE TEMPLATE

```
#!/bin/

# Navigate to the directory containing the docker-compose.yml

cd /path/to/your/project

# Start Docker containers

echo "Starting services using Docker Compose..."

docker-compose up -d

echo "All services are up and running!"
```

5.1.5 STOP SCRIPT

To stop the services in the Linux booting.

5.1.5.1 CODE TEMPLATE

```
#!/bin/

# Navigate to the directory containing the docker-compose.yml

cd /path/to/your/project

# Stop Docker containers

echo "Stopping services..."

docker-compose down

echo "All services have been stopped."
```

5.1.6 NODE RED MODULE

To create a flow between the MQTT client, Broker and to store the data into the Influx DB.

5.1.6.1 CODE TEMPLATE

```
[
    {
        "id": "cb87a6105818786f",
        "type": "tab",
        "label": "Pico_W",
        "disabled": false,
        "info": "Dht11 is read from the Pico W trough Eclipse mosquito MQTT protocol and is connected to the node red",
        "env": []
    },
```

```
{
    "id": "864a34dcff58dbc8",
    "type": "mqtt in",
    "z": "cb87a6105818786f",
    "name": "Temperature",
    "topic": "sensor/dht11/temperature",
    "qos": "2",
    "datatype": "auto-detect",
    "broker": "59da11c03e8c4521",
    "nl": false,
    "rap": true,
    "rh": 0,
    "inputs": 0,
    "x": 330,
    "y": 140,
    "wires": [
        [
            "3fcc9b2e5d917427",
            "36ce48acbcd8cede"
        ]
    ],
    "outputLabels": [
        "sensor/dht11/humidity"
```

```
    ]
},
{
    "id": "3fcc9b2e5d917427",
    "type": "debug",
    "z": "cb87a6105818786f",
    "name": "debug 1",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "true",
    "targetType": "full",
    "statusVal": "",
    "statusType": "auto",
    "x": 640,
    "y": 140,
    "wires": []
},
{
    "id": "15528e623eb2d3cb",
    "type": "mqtt in",
    "z": "cb87a6105818786f",
```

```
    "name": "Humidity",
    "topic": "sensor/dht11/humidity",
    "qos": "2",
    "datatype": "auto-detect",
    "broker": "59da11c03e8c4521",
    "nl": false,
    "rap": true,
    "rh": 0,
    "inputs": 0,
    "x": 320,
    "y": 240,
    "wires": [
        [
            "d385fec921ab7757",
            "bf4fcd753a189cb0"
        ]
    ],
    "outputLabels": [
        "sensor/dht11/humidity"
    ]
},
{
    "id": "d385fec921ab7757",
```

```
    "type": "debug",
    "z": "cb87a6105818786f",
    "name": "debug 2",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "true",
    "targetType": "full",
    "statusVal": "",
    "statusType": "auto",
    "x": 640,
    "y": 240,
    "wires": []
},
{
    "id": "36ce48acbcd8cede",
    "type": "influxdb out",
    "z": "cb87a6105818786f",
    "influxdb": "4b60519ac92346cf",
    "name": "Temperature Celsius",
    "measurement": "°C",
    "precision": "",
```

```
    "retentionPolicy": "",
    "database": "database",
    "precisionV18FluxV20": "ms",
    "retentionPolicyV18Flux": "",
    "org": "myorg",
    "bucket": "sensor_data",
    "x": 560,
    "y": 60,
    "wires": []
},
{
    "id": "bf4fcd753a189cb0",
    "type": "influxdb out",
    "z": "cb87a6105818786f",
    "influxdb": "4b60519ac92346cf",
    "name": "Humidity Percentage",
    "measurement": "%",
    "precision": "",
    "retentionPolicy": "",
    "database": "database",
    "precisionV18FluxV20": "ms",
    "retentionPolicyV18Flux": "",
    "org": "myorg",
```

```
    "bucket": "sensor_data",
    "x": 550,
    "y": 340,
    "wires": []
},
{
    "id": "59da11c03e8c4521",
    "type": "mqtt-broker",
    "name": "dht11-picow",
    "broker": "localhost",
    "port": "1883",
    "clientid": "",
    "autoConnect": true,
    "usetls": false,
    "protocolVersion": "4",
    "keepalive": "60",
    "cleansession": true,
    "autoUnsubscribe": true,
    "birthTopic": "",
    "birthQos": "0",
    "birthRetain": "false",
    "birthPayload": "",
    "birthMsg": {},
```

```
    "closeTopic": "",
    "closeQos": "0",
    "closeRetain": "false",
    "closePayload": "",
    "closeMsg": {},
    "willTopic": "",
    "willQos": "0",
    "willRetain": "false",
    "willPayload": "",
    "willMsg": {},
    "userProps": "",
    "sessionExpiry": ""
},
{
    "id": "4b60519ac92346cf",
    "type": "influxdb",
    "hostname": "localhost",
    "port": "8086",
    "protocol": "http",
    "database": "sensor_data",
    "name": "influxdb",
    "usetls": false,
    "tls": "",
```

```
    "influxdbVersion": "2.0",
    "url": "http://localhost:8087",
    "timeout": "10",
    "rejectUnauthorized": false
  }
]
```

5.1.7 GRAFANA MODULE

To create a dashboard in the grafana UI to display the temperature and humidity.

5.1.7.1 CODE TEMPLATE

```
{{
 "__inputs": [
  {
   "name": "DS_INFLUXDB",
   "label": "influxdb",
   "description": "",
   "type": "datasource",
   "pluginId": "influxdb",
   "pluginName": "InfluxDB"
  }
 ],
 "__elements": {},
 "__requires": [
  {
   "type": "panel",
   "id": "gauge",
   "name": "Gauge",
   "version": ""
  },
  {
```

```
    "type": "grafana",
    "id": "grafana",
    "name": "Grafana",
    "version": "11.3.0"
   },
   {
    "type": "datasource",
    "id": "influxdb",
    "name": "InfluxDB",
    "version": "1.0.0"
   },
   {
    "type": "panel",
    "id": "timeseries",
    "name": "Time series",
    "version": ""
   }
  ],
  "annotations": {
   "list": [
    {
     "builtIn": 1,
     "datasource": {
      "type": "grafana",
      "uid": "-- Grafana --"
     },
     "enable": true,
     "hide": true,
     "iconColor": "rgba(0, 211, 255, 1)",
     "name": "Annotations & Alerts",
     "type": "dashboard"
    }
   ]
```

```
  },
  "description": "1. DHT-11 sensor module data is read using the Raspberry Pi - PICO W and published to Eclipse Mosquito MQTT broker.\nUsing Node RED - the data are stored to the influxdb2 that received from the MQTT Client(Pico W)\nUsing Grafana - to create the dashboard to create aggregation of 10 seconds",
  "editable": true,
  "fiscalYearStartMonth": 0,
  "graphTooltip": 0,
  "id": null,
  "links": [],
  "liveNow": true,
  "panels": [
   {
    "datasource": {
     "type": "influxdb",
     "uid": "${DS_INFLUXDB}"
    },
    "description": "Basic - Indoor humidity should be 30% ~ 50%",
    "fieldConfig": {
     "defaults": {
      "color": {
       "mode": "thresholds"
      },
      "custom": {
       "neutral": 50
      },
      "fieldMinMax": false,
      "mappings": [],
      "thresholds": {
       "mode": "percentage",
       "steps": [
        {
         "color": "blue",
```

```
          "value": null
        },
        {
          "color": "green",
          "value": 30
        },
        {
          "color": "orange",
          "value": 60
        }
      ]
    },
    "unit": "percent"
  },
  "overrides": []
},
"gridPos": {
  "h": 8,
  "w": 11,
  "x": 0,
  "y": 0
},
"id": 3,
"options": {
  "minVizHeight": 34,
  "minVizWidth": 93,
  "orientation": "auto",
  "reduceOptions": {
    "calcs": [
      "lastNotNull"
    ],
    "fields": "",
    "values": false
```

```
  },
  "showThresholdLabels": true,
  "showThresholdMarkers": true,
  "sizing": "auto",
  "text": {}
},
"pluginVersion": "11.3.0",
"targets": [
  {
    "groupBy": [
      {
        "params": [
          "$__interval"
        ],
        "type": "time"
      },
      {
        "params": [
          "null"
        ],
        "type": "fill"
      }
    ],
    "measurement": "%",
    "orderByTime": "ASC",
    "policy": "default",
    "refId": "A",
    "resultFormat": "time_series",
    "select": [
      [
        {
          "params": [
            "value"
```

```
          ],
          "type": "field"
        },
        {
          "params": [],
          "type": "mean"
        }
      ]
    ],
    "tags": [],
    "datasource": {
      "type": "influxdb",
      "uid": "${DS_INFLUXDB}"
    }
  }
 ],
 "title": "Humidity",
 "transparent": true,
 "type": "gauge"
},
{
 "datasource": {
  "type": "influxdb",
  "uid": "${DS_INFLUXDB}"
 },
 "description": "The World Health Organization (WHO) recommends a temperature of 18–24 °C (64–75 °F) for healthy adults.",
 "fieldConfig": {
  "defaults": {
   "color": {
    "mode": "thresholds"
   },
   "custom": {
```

```
      "neutral": 0
    },
    "fieldMinMax": false,
    "mappings": [],
    "max": -2,
    "thresholds": {
      "mode": "percentage",
      "steps": [
        {
          "color": "semi-dark-green",
          "value": null
        },
        {
          "color": "orange",
          "value": 40
        }
      ]
    },
    "unit": "celsius"
  },
  "overrides": []
},
"gridPos": {
  "h": 8,
  "w": 11,
  "x": 13,
  "y": 0
},
"id": 4,
"options": {
  "minVizHeight": 75,
  "minVizWidth": 75,
  "orientation": "auto",
```

```
  "reduceOptions": {
    "calcs": [
      "lastNotNull"
    ],
    "fields": "",
    "values": false
  },
  "showThresholdLabels": true,
  "showThresholdMarkers": true,
  "sizing": "auto",
  "text": {}
},
"pluginVersion": "11.3.0",
"targets": [
  {
    "groupBy": [
      {
        "params": [
          "$__interval"
        ],
        "type": "time"
      },
      {
        "params": [
          "null"
        ],
        "type": "fill"
      }
    ],
    "measurement": "°C",
    "orderByTime": "ASC",
    "policy": "default",
    "refId": "A",
```

```
      "resultFormat": "time_series",
      "select": [
       [
        {
         "params": [
          "value"
         ],
         "type": "field"
        },
        {
         "params": [],
         "type": "mean"
        }
       ]
      ],
      "tags": [],
      "datasource": {
       "type": "influxdb",
       "uid": "${DS_INFLUXDB}"
      }
     }
    ],
    "title": "Temperature",
    "transparent": true,
    "type": "gauge"
   },
   {
    "datasource": {
     "type": "influxdb",
     "uid": "${DS_INFLUXDB}"
    },
    "description": "Room 1",
    "fieldConfig": {
```

```
"defaults": {
  "color": {
    "fixedColor": "green",
    "mode": "shades",
    "seriesBy": "last"
  },
  "custom": {
    "axisBorderShow": true,
    "axisCenteredZero": false,
    "axisColorMode": "text",
    "axisLabel": "Temperature",
    "axisPlacement": "auto",
    "barAlignment": 0,
    "barWidthFactor": 0.6,
    "drawStyle": "line",
    "fillOpacity": 18,
    "gradientMode": "hue",
    "hideFrom": {
      "legend": false,
      "tooltip": false,
      "viz": false
    },
    "insertNulls": false,
    "lineInterpolation": "stepAfter",
    "lineStyle": {
      "fill": "solid"
    },
    "lineWidth": 1,
    "pointSize": 5,
    "scaleDistribution": {
      "type": "linear"
    },
    "showPoints": "auto",
```

```
      "spanNulls": false,
      "stacking": {
        "group": "A",
        "mode": "none"
      },
      "thresholdsStyle": {
        "mode": "line"
      }
    },
    "mappings": [],
    "thresholds": {
      "mode": "absolute",
      "steps": [
        {
          "color": "green",
          "value": null
        },
        {
          "color": "red",
          "value": 33
        }
      ]
    },
    "unit": "celsius"
  },
  "overrides": []
},
"gridPos": {
  "h": 9,
  "w": 24,
  "x": 0,
  "y": 8
},
```

```
"id": 1,
"options": {
  "legend": {
    "calcs": [
      "lastNotNull"
    ],
    "displayMode": "list",
    "placement": "bottom",
    "showLegend": true
  },
  "tooltip": {
    "mode": "single",
    "sort": "none"
  }
},
"pluginVersion": "11.3.0",
"targets": [
  {
    "datasource": {
      "type": "influxdb",
      "uid": "${DS_INFLUXDB}"
    },
    "groupBy": [
      {
        "params": [
          "1m"
        ],
        "type": "time"
      },
      {
        "params": [
          "none"
        ],
```

```
          "type": "fill"
        }
      ],
      "hide": false,
      "measurement": "°C",
      "orderByTime": "ASC",
      "policy": "default",
      "refId": "A",
      "resultFormat": "time_series",
      "select": [
        [
          {
            "params": [
              "value"
            ],
            "type": "field"
          },
          {
            "params": [],
            "type": "mean"
          }
        ]
      ],
      "tags": []
    }
  ],
  "title": "Room Temperature",
  "transparent": true,
  "type": "timeseries"
},
{
  "datasource": {
    "type": "influxdb",
```

```
  "uid": "${DS_INFLUXDB}"
},
"description": "Room 1",
"fieldConfig": {
  "defaults": {
    "color": {
      "mode": "palette-classic"
    },
    "custom": {
      "axisBorderShow": false,
      "axisCenteredZero": false,
      "axisColorMode": "text",
      "axisLabel": "Humidity",
      "axisPlacement": "auto",
      "barAlignment": 0,
      "barWidthFactor": 0.6,
      "drawStyle": "line",
      "fillOpacity": 0,
      "gradientMode": "none",
      "hideFrom": {
        "legend": false,
        "tooltip": false,
        "viz": false
      },
      "insertNulls": false,
      "lineInterpolation": "stepAfter",
      "lineWidth": 1,
      "pointSize": 5,
      "scaleDistribution": {
        "type": "linear"
      },
      "showPoints": "auto",
      "spanNulls": false,
```

```
      "stacking": {
        "group": "A",
        "mode": "none"
      },
      "thresholdsStyle": {
        "mode": "off"
      }
    },
    "mappings": [],
    "thresholds": {
      "mode": "absolute",
      "steps": [
        {
          "color": "green",
          "value": null
        },
        {
          "color": "red",
          "value": 80
        }
      ]
    }
  },
  "overrides": []
},
"gridPos": {
  "h": 8,
  "w": 24,
  "x": 0,
  "y": 17
},
"id": 2,
"options": {
```

```
  "legend": {
    "calcs": [
      "lastNotNull"
    ],
    "displayMode": "list",
    "placement": "bottom",
    "showLegend": true
  },
  "tooltip": {
    "mode": "single",
    "sort": "none"
  }
},
"pluginVersion": "11.3.0",
"targets": [
  {
    "groupBy": [
      {
        "params": [
          "1m"
        ],
        "type": "time"
      },
      {
        "params": [
          "none"
        ],
        "type": "fill"
      }
    ],
    "measurement": "%",
    "orderByTime": "ASC",
    "policy": "default",
```

```
        "refId": "A",
        "resultFormat": "time_series",
        "select": [
          [
            {
              "params": [
                "value"
              ],
              "type": "field"
            },
            {
              "params": [],
              "type": "mean"
            }
          ]
        ],
        "tags": [],
        "datasource": {
          "type": "influxdb",
          "uid": "${DS_INFLUXDB}"
        }
      }
    ],
    "title": "Room Humidity",
    "transparent": true,
    "type": "timeseries"
  }
],
"schemaVersion": 40,
"tags": [],
"templating": {
  "list": []
},
```

```
  "time": {
    "from": "now-1h",
    "to": "now"
  },
  "timepicker": {},
  "timezone": "",
  "title": "DHT11-Data",
  "uid": "be24b4tsem800d",
  "version": 25,
  "weekStart": ""
}
```

```
  "__inputs": [
    {
      "name": "DS_INFLUXDB",
      "label": "influxdb",
      "description": "",
      "type": "datasource",
      "pluginId": "influxdb",
      "pluginName": "InfluxDB"
    }
  ],
  "__elements": {},
  "__requires": [
    {
      "type": "panel",
      "id": "gauge",
      "name": "Gauge",
      "version": ""
    },
    {
      "type": "grafana",
      "id": "grafana",
```

```
    "name": "Grafana",
    "version": "11.3.0"
   },
   {
    "type": "datasource",
    "id": "influxdb",
    "name": "InfluxDB",
    "version": "1.0.0"
   },
   {
    "type": "panel",
    "id": "timeseries",
    "name": "Time series",
    "version": ""
   }
  ],
  "annotations": {
   "list": [
    {
     "builtIn": 1,
     "datasource": {
      "type": "grafana",
      "uid": "-- Grafana --"
     },
     "enable": true,
     "hide": true,
     "iconColor": "rgba(0, 211, 255, 1)",
     "name": "Annotations & Alerts",
     "type": "dashboard"
    }
   ]
  },
```

```
  "description": "1. DHT-11 sensor module data is read using the Raspberry Pi - PICO W and published to Eclipse Mosquito MQTT broker.\nUsing Node RED - the data are stored to the influxdb2 that received from the MQTT Client(Pico W)\nUsing Grafana - to create the dashboard to create aggregation of 10 seconds",
  "editable": true,
  "fiscalYearStartMonth": 0,
  "graphTooltip": 0,
  "id": null,
  "links": [],
  "liveNow": true,
  "panels": [
   {
    "datasource": {
     "type": "influxdb",
     "uid": "${DS_INFLUXDB}"
    },
    "description": "Basic - Indoor humidity should be 30% ~ 50%",
    "fieldConfig": {
     "defaults": {
      "color": {
       "mode": "thresholds"
      },
      "custom": {
       "neutral": 50
      },
      "fieldMinMax": false,
      "mappings": [],
      "thresholds": {
       "mode": "percentage",
       "steps": [
        {
         "color": "blue",
         "value": null
```

```
        },
        {
          "color": "green",
          "value": 30
        },
        {
          "color": "orange",
          "value": 60
        }
      ]
    },
    "unit": "percent"
  },
  "overrides": []
},
"gridPos": {
  "h": 8,
  "w": 11,
  "x": 0,
  "y": 0
},
"id": 3,
"options": {
  "minVizHeight": 34,
  "minVizWidth": 93,
  "orientation": "auto",
  "reduceOptions": {
    "calcs": [
      "lastNotNull"
    ],
    "fields": "",
    "values": false
  },
```

```
  "showThresholdLabels": true,
  "showThresholdMarkers": true,
  "sizing": "auto",
  "text": {}
},
"pluginVersion": "11.3.0",
"targets": [
  {
    "groupBy": [
      {
        "params": [
          "$__interval"
        ],
        "type": "time"
      },
      {
        "params": [
          "null"
        ],
        "type": "fill"
      }
    ],
    "measurement": "%",
    "orderByTime": "ASC",
    "policy": "default",
    "refId": "A",
    "resultFormat": "time_series",
    "select": [
      [
        {
          "params": [
            "value"
          ],
```

```
            "type": "field"
           },
           {
            "params": [],
            "type": "mean"
           }
          ]
         ],
         "tags": [],
         "datasource": {
          "type": "influxdb",
          "uid": "${DS_INFLUXDB}"
         }
        }
       ],
       "title": "Humidity",
       "transparent": true,
       "type": "gauge"
      },
      {
       "datasource": {
        "type": "influxdb",
        "uid": "${DS_INFLUXDB}"
       },
       "description": "The World Health Organization (WHO) recommends a temperature of 18–24 °C (64–75 °F) for healthy adults.",
       "fieldConfig": {
        "defaults": {
         "color": {
          "mode": "thresholds"
         },
         "custom": {
          "neutral": 0
```

```
    },
    "fieldMinMax": false,
    "mappings": [],
    "max": -2,
    "thresholds": {
     "mode": "percentage",
     "steps": [
      {
       "color": "semi-dark-green",
       "value": null
      },
      {
       "color": "orange",
       "value": 40
      }
     ]
    },
    "unit": "celsius"
   },
   "overrides": []
  },
  "gridPos": {
   "h": 8,
   "w": 11,
   "x": 13,
   "y": 0
  },
  "id": 4,
  "options": {
   "minVizHeight": 75,
   "minVizWidth": 75,
   "orientation": "auto",
   "reduceOptions": {
```

```
    "calcs": [
      "lastNotNull"
    ],
    "fields": "",
    "values": false
  },
  "showThresholdLabels": true,
  "showThresholdMarkers": true,
  "sizing": "auto",
  "text": {}
},
"pluginVersion": "11.3.0",
"targets": [
  {
    "groupBy": [
      {
        "params": [
          "$__interval"
        ],
        "type": "time"
      },
      {
        "params": [
          "null"
        ],
        "type": "fill"
      }
    ],
    "measurement": "°C",
    "orderByTime": "ASC",
    "policy": "default",
    "refId": "A",
    "resultFormat": "time_series",
```

```
      "select": [
        [
          {
            "params": [
              "value"
            ],
            "type": "field"
          },
          {
            "params": [],
            "type": "mean"
          }
        ]
      ],
      "tags": [],
      "datasource": {
        "type": "influxdb",
        "uid": "${DS_INFLUXDB}"
      }
    }
  ],
  "title": "Temperature",
  "transparent": true,
  "type": "gauge"
},
{
  "datasource": {
    "type": "influxdb",
    "uid": "${DS_INFLUXDB}"
  },
  "description": "Room 1",
  "fieldConfig": {
    "defaults": {
```

```
"color": {
  "fixedColor": "green",
  "mode": "shades",
  "seriesBy": "last"
},
"custom": {
  "axisBorderShow": true,
  "axisCenteredZero": false,
  "axisColorMode": "text",
  "axisLabel": "Temperature",
  "axisPlacement": "auto",
  "barAlignment": 0,
  "barWidthFactor": 0.6,
  "drawStyle": "line",
  "fillOpacity": 18,
  "gradientMode": "hue",
  "hideFrom": {
    "legend": false,
    "tooltip": false,
    "viz": false
  },
  "insertNulls": false,
  "lineInterpolation": "stepAfter",
  "lineStyle": {
    "fill": "solid"
  },
  "lineWidth": 1,
  "pointSize": 5,
  "scaleDistribution": {
    "type": "linear"
  },
  "showPoints": "auto",
  "spanNulls": false,
```

```
      "stacking": {
        "group": "A",
        "mode": "none"
      },
      "thresholdsStyle": {
        "mode": "line"
      }
    },
    "mappings": [],
    "thresholds": {
      "mode": "absolute",
      "steps": [
        {
          "color": "green",
          "value": null
        },
        {
          "color": "red",
          "value": 33
        }
      ]
    },
    "unit": "celsius"
  },
  "overrides": []
},
"gridPos": {
  "h": 9,
  "w": 24,
  "x": 0,
  "y": 8
},
"id": 1,
```

```
"options": {
  "legend": {
    "calcs": [
      "lastNotNull"
    ],
    "displayMode": "list",
    "placement": "bottom",
    "showLegend": true
  },
  "tooltip": {
    "mode": "single",
    "sort": "none"
  }
},
"pluginVersion": "11.3.0",
"targets": [
  {
    "datasource": {
      "type": "influxdb",
      "uid": "${DS_INFLUXDB}"
    },
    "groupBy": [
      {
        "params": [
          "1m"
        ],
        "type": "time"
      },
      {
        "params": [
          "none"
        ],
        "type": "fill"
```

```
        }
      ],
      "hide": false,
      "measurement": "°C",
      "orderByTime": "ASC",
      "policy": "default",
      "refId": "A",
      "resultFormat": "time_series",
      "select": [
        [
          {
            "params": [
              "value"
            ],
            "type": "field"
          },
          {
            "params": [],
            "type": "mean"
          }
        ]
      ],
      "tags": []
    }
  ],
  "title": "Room Temperature",
  "transparent": true,
  "type": "timeseries"
},
{
  "datasource": {
    "type": "influxdb",
    "uid": "${DS_INFLUXDB}"
```

```
},
"description": "Room 1",
"fieldConfig": {
  "defaults": {
    "color": {
      "mode": "palette-classic"
    },
    "custom": {
      "axisBorderShow": false,
      "axisCenteredZero": false,
      "axisColorMode": "text",
      "axisLabel": "Humidity",
      "axisPlacement": "auto",
      "barAlignment": 0,
      "barWidthFactor": 0.6,
      "drawStyle": "line",
      "fillOpacity": 0,
      "gradientMode": "none",
      "hideFrom": {
        "legend": false,
        "tooltip": false,
        "viz": false
      },
      "insertNulls": false,
      "lineInterpolation": "stepAfter",
      "lineWidth": 1,
      "pointSize": 5,
      "scaleDistribution": {
        "type": "linear"
      },
      "showPoints": "auto",
      "spanNulls": false,
      "stacking": {
```

```
        "group": "A",
        "mode": "none"
      },
      "thresholdsStyle": {
        "mode": "off"
      }
    },
    "mappings": [],
    "thresholds": {
      "mode": "absolute",
      "steps": [
        {
          "color": "green",
          "value": null
        },
        {
          "color": "red",
          "value": 80
        }
      ]
    }
  },
  "overrides": []
},
"gridPos": {
  "h": 8,
  "w": 24,
  "x": 0,
  "y": 17
},
"id": 2,
"options": {
  "legend": {
```

```
    "calcs": [
      "lastNotNull"
    ],
    "displayMode": "list",
    "placement": "bottom",
    "showLegend": true
  },
  "tooltip": {
    "mode": "single",
    "sort": "none"
  }
},
"pluginVersion": "11.3.0",
"targets": [
  {
    "groupBy": [
      {
        "params": [
          "1m"
        ],
        "type": "time"
      },
      {
        "params": [
          "none"
        ],
        "type": "fill"
      }
    ],
    "measurement": "%",
    "orderByTime": "ASC",
    "policy": "default",
    "refId": "A",
```

```
      "resultFormat": "time_series",
      "select": [
       [
        {
         "params": [
          "value"
         ],
         "type": "field"
        },
        {
         "params": [],
         "type": "mean"
        }
       ]
      ],
      "tags": [],
      "datasource": {
       "type": "influxdb",
       "uid": "${DS_INFLUXDB}"
      }
     }
    ],
    "title": "Room Humidity",
    "transparent": true,
    "type": "timeseries"
  }
],
"schemaVersion": 40,
"tags": [],
"templating": {
  "list": []
},
"time": {
```

```
    "from": "now-1h",
    "to": "now"
  },
  "timepicker": {},
  "timezone": "",
  "title": "DHT11-Data",
  "uid": "be24b4tsem800d",
  "version": 25,
  "weekStart": ""
}
```


5.2 TABLES

5.2.1 MEASUREMENT TABLE

FIELD	TYPE	DESCRIPTION
temperature	float	Captures temperature values.
humidity	float	Captures humidity values.

Table 5. 2. 1 Measurement Table

5.2.2 TAG TABLE

FIELD	TYPE	DESCRIPTION
location	string	Describes the sensor's location.
device_id	string	Identifies the sensor device.

Table 5. 2. 2 Tag Table

CHAPTER 6

TESTING

6.1 TESTING METHODOLOGIES

6.1.1 UNIT TESTING

- **Objective:** Test individual components or modules in isolation to ensure they work as intended.
- **Example:** Test the DHT11 sensor data retrieval functionality or MQTT message publishing.

6.1.2. INTEGRATION TESTING

- **Objective:** Verify that different modules interact correctly when integrated.
- **Example:** Test the interaction between the MQTT broker, InfluxDB, and Grafana to ensure seamless data flow.

6.1.3. FUNCTIONAL TESTING

- **Objective:** Validate that the system meets all functional requirements.
- **Example:** Test whether temperature and humidity readings are correctly displayed on the Grafana dashboard.

6.1.4. SYSTEM TESTING

- **Objective:** Test the entire system as a whole, including hardware, software, and network components.
- **Example:** Simulate real-world scenarios like data loss during Wi-Fi disconnection and ensure the system recovers gracefully.

6.1.5. PERFORMANCE TESTING

- **Objective:** Evaluate the system's performance under various conditions, such as heavy loads or network delays.
- **Example:** Measure latency in data transmission from the DHT11 sensor to Grafana.

6.1.6. USABILITY TESTING

- **Objective:** Assess how easily users can interact with the system, including configuring alerts and viewing dashboards.
- **Example:** Test if a user can set up a Grafana dashboard with minimal guidance.

6.1.7. REGRESSION TESTING

- **Objective:** Ensure that new changes or updates do not introduce defects into existing functionality.
- **Example:** Test the system after adding support for an additional sensor to confirm previous functionalities remain unaffected.

6.2 TEST CASE

6.2.1 UNIT TEST CASE: SENSOR MODULE

Test Case ID	TC-01-Sensor
Description	Validate that the DHT11 sensor returns accurate temperature and humidity data.
Input	Readings from the DHT11 sensor (simulated or real).
Expected Output	Accurate temperature and humidity values (within ±2°C and ±5% RH of actual conditions).
Steps	1. Connect DHT11 sensor to Pico W. 2. Run the MicroPython script to fetch data. 3. Compare output with a calibrated device.
Pass Criteria	Output matches the expected range of readings.
Fail Criteria	Output deviates beyond acceptable limits.

Table 6. 2. 1 Unit Test Case: Sensor Module

6.2.2 INTEGRATION TEST CASE: MQTT BROKER

Test Case ID	TC-02-MQTT
Description	Test the interaction between the MQTT client (Pico W) and the MQTT broker (Mosquitto).
Input	JSON payload: {"temperature": 25, "humidity": 50}
Expected Output	Message is successfully published to the broker and received by the subscriber.
Steps	1. Configure MQTT broker and client. 2. Publish a test message from the Pico W. 3. Verify the message is received on the subscriber's end.
Pass Criteria	Message is received correctly and without delay.
Fail Criteria	Message fails to publish, or is delayed/lost.

Table 6. 2. 2 Integration Test Case: MQTT Broker

6.2.3 FUNCTIONAL TEST CASE: DASHBOARD DISPLAY

Test Case ID	TC-03-Dashboard
Description	Ensure real-time data is displayed accurately on the Grafana dashboard.
Input	Sensor readings: Temperature = 26°C, Humidity = 55%
Expected Output	Grafana dashboard updates with the correct data within 5 seconds.
Steps	1. Configure Grafana to query InfluxDB. 2. Simulate sensor data transmission. 3. Observe dashboard updates.
Pass Criteria	Data appears on the dashboard in real-time.

Fail Criteria	Data does not appear or updates with significant delay.

Table 6. 2. 3 Functional Test Case: Dashboard Display

6.2.4 SYSTEM TEST CASE: NETWORK DISCONNECTION

Test Case ID	**TC-04-Network**
Description	Validate the system's behavior during network disconnection.
Input	Disconnect Wi-Fi for 30 seconds.
Expected Output	Data transmission resumes without issues when Wi-Fi reconnects.
Steps	1. Disconnect the Raspberry Pi Pico W from Wi-Fi. 2. Wait for 30 seconds and reconnect. 3. Check if data continues to be published and received.
Pass Criteria	Data transmission resumes automatically without errors.
Fail Criteria	Data is lost or requires manual intervention to reconnect.

Table 6. 2. 4 System Test Case: Network Disconnection

6.2.5 PERFORMANCE TEST CASE: DATA LOAD

Test Case ID	TC-05-Performance
Description	Test the system's ability to handle high-frequency data transmission.
Input	Simulate sensor readings at intervals of 1 second for 10 minutes.
Expected Output	All messages are received without loss or significant delay.
Steps	1. Simulate data publishing at 1-second intervals. 2. Monitor the MQTT broker and InfluxDB for message receipt.
Pass Criteria	All 600 messages (10 minutes × 60 seconds) are logged in InfluxDB.
Fail Criteria	Messages are dropped or delayed significantly.

Table 6. 2. 5 Performance Test Case: Data Load

CHAPTER 7

OUTPUT SCREENS

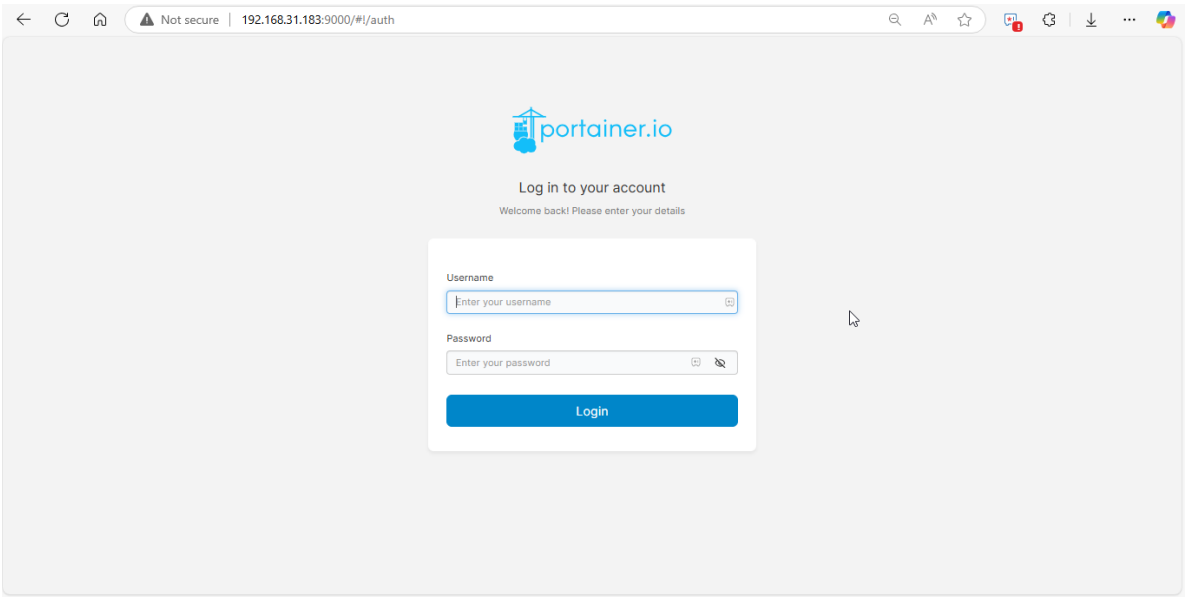

Photo 7. 1 Docker Login Page

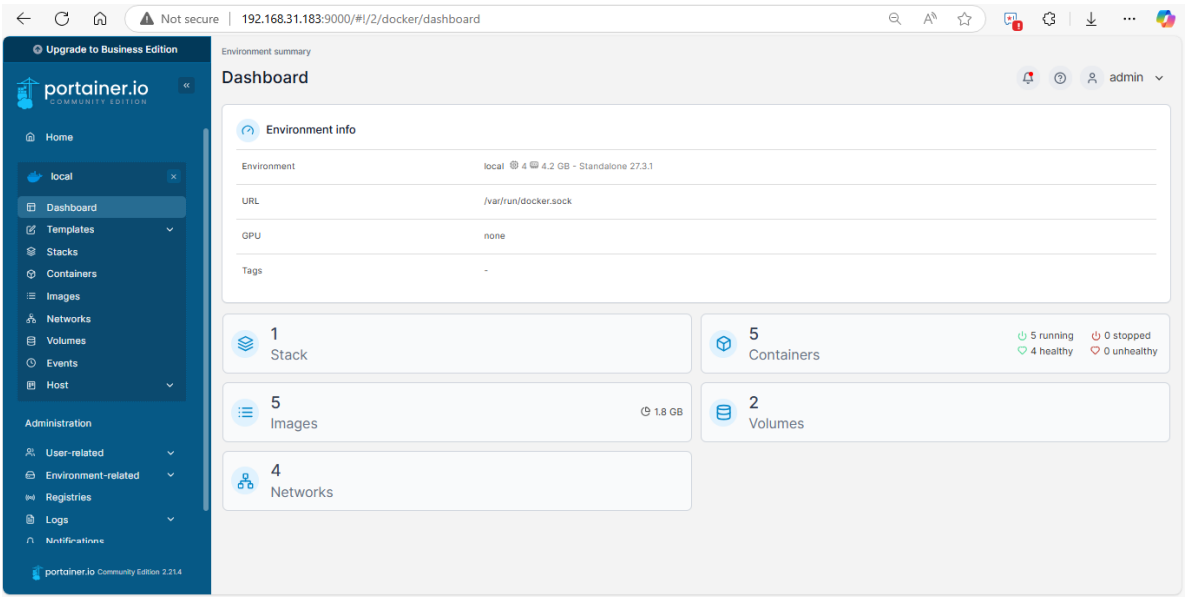

Photo 7. 2 Docker Dashboard Page

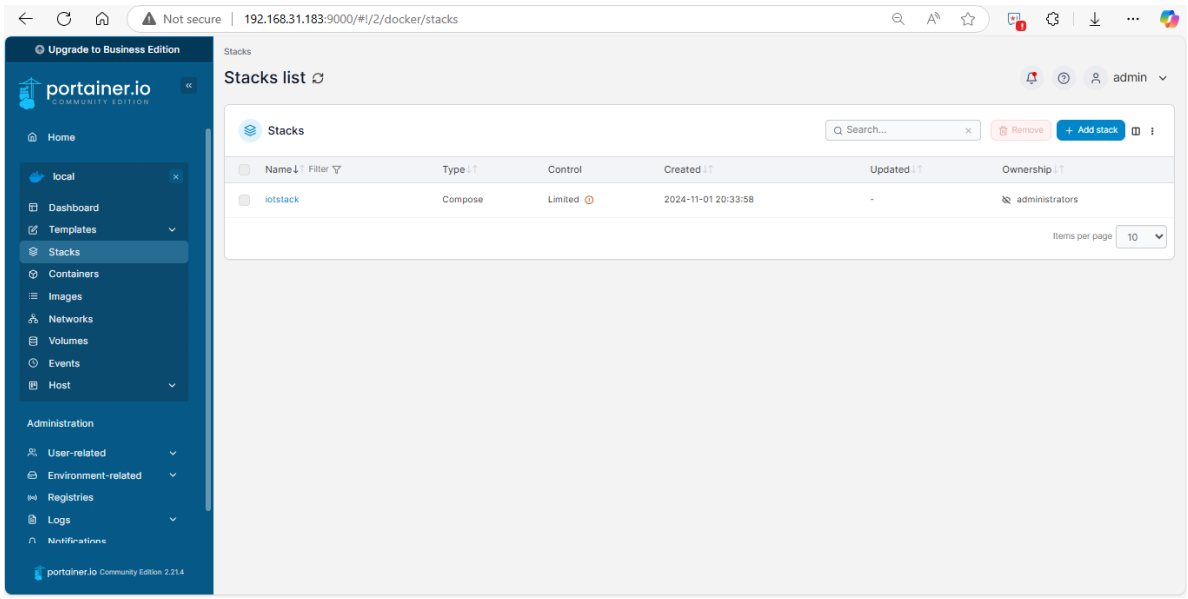

Photo 7. 3 Docker Stacks Page

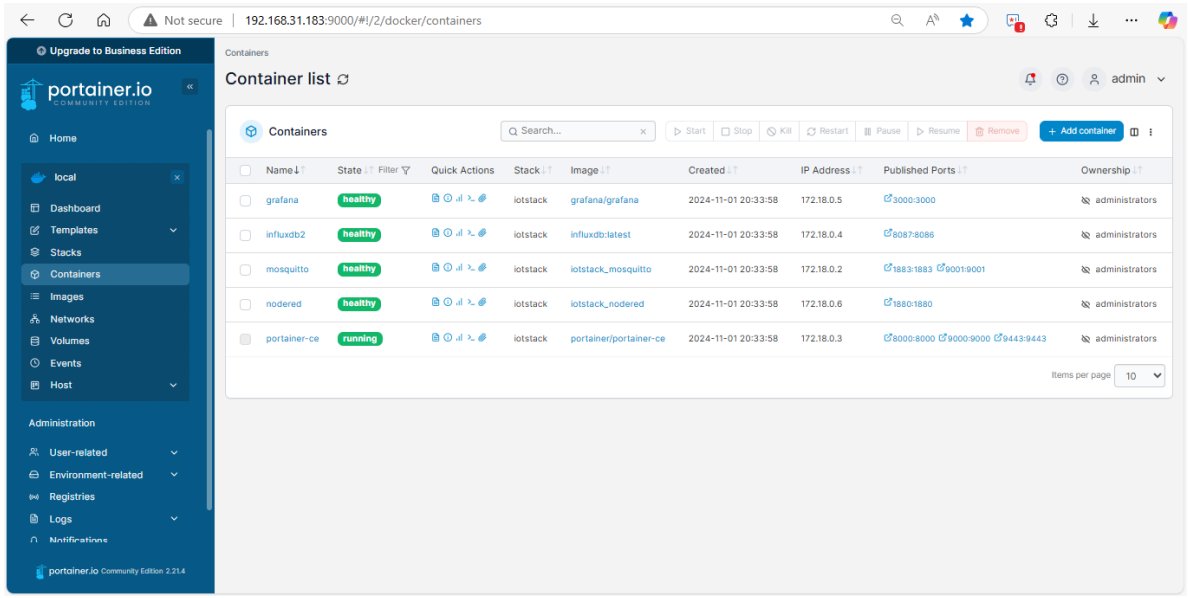

Photo 7. 4 Docker Containers Page

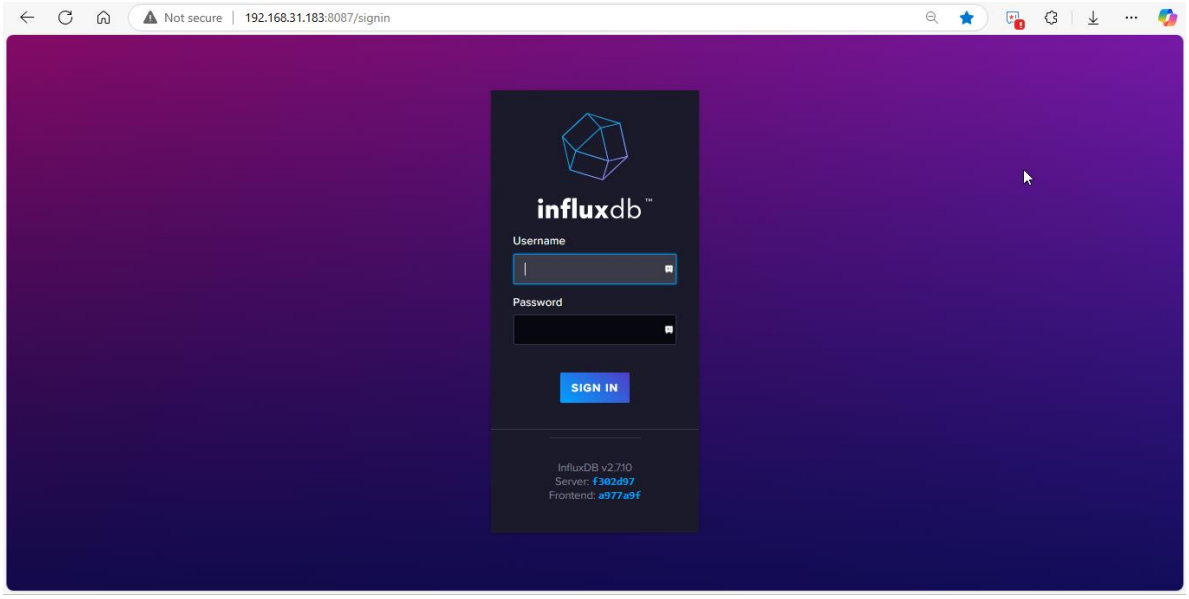

Photo 7. 5 Influx-DB Login Page

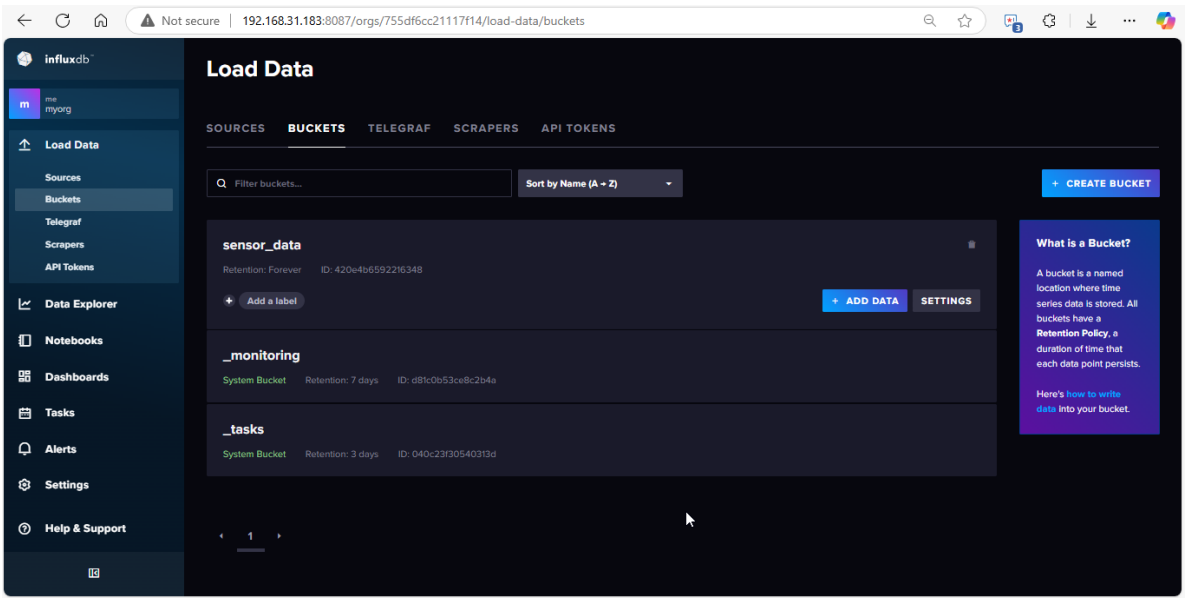

Photo 7. 6 Influx-DB Load Data Page

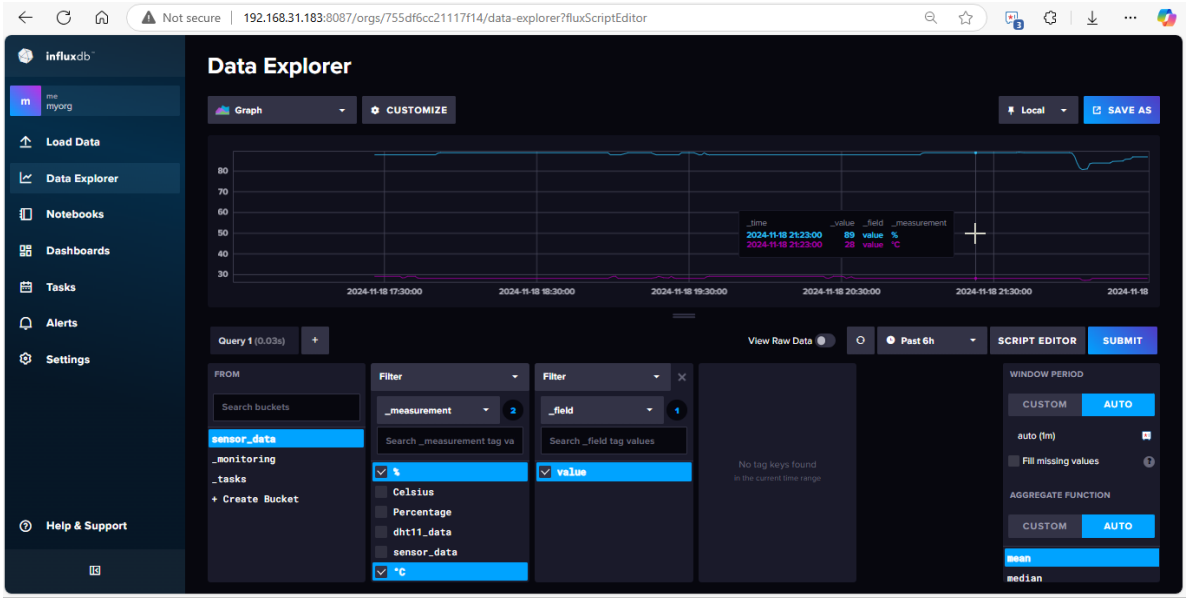

Photo 7. 7 Influx-DB Data Explorer Page

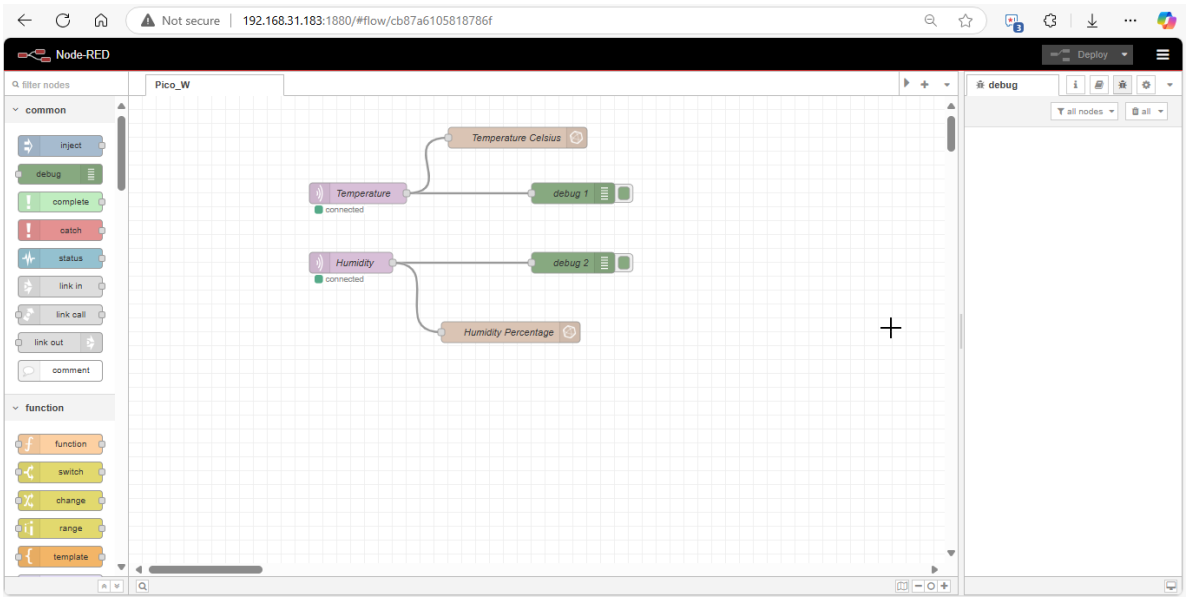

Photo 7. 8 Node RED Flowchart Page

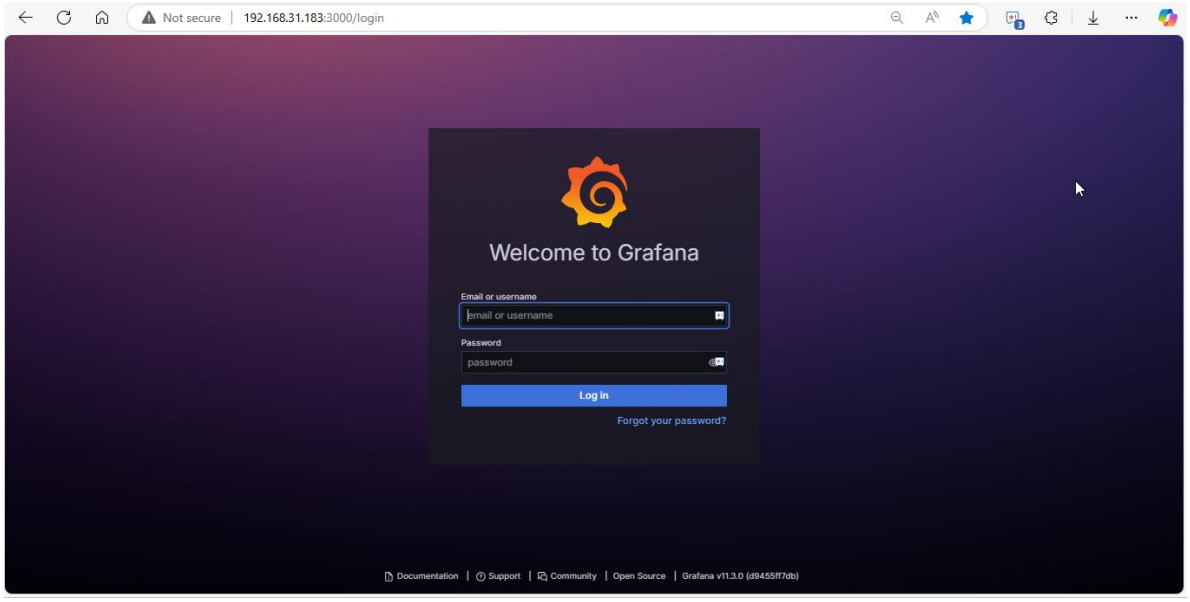

Photo 7. 9 Grafana Login Page

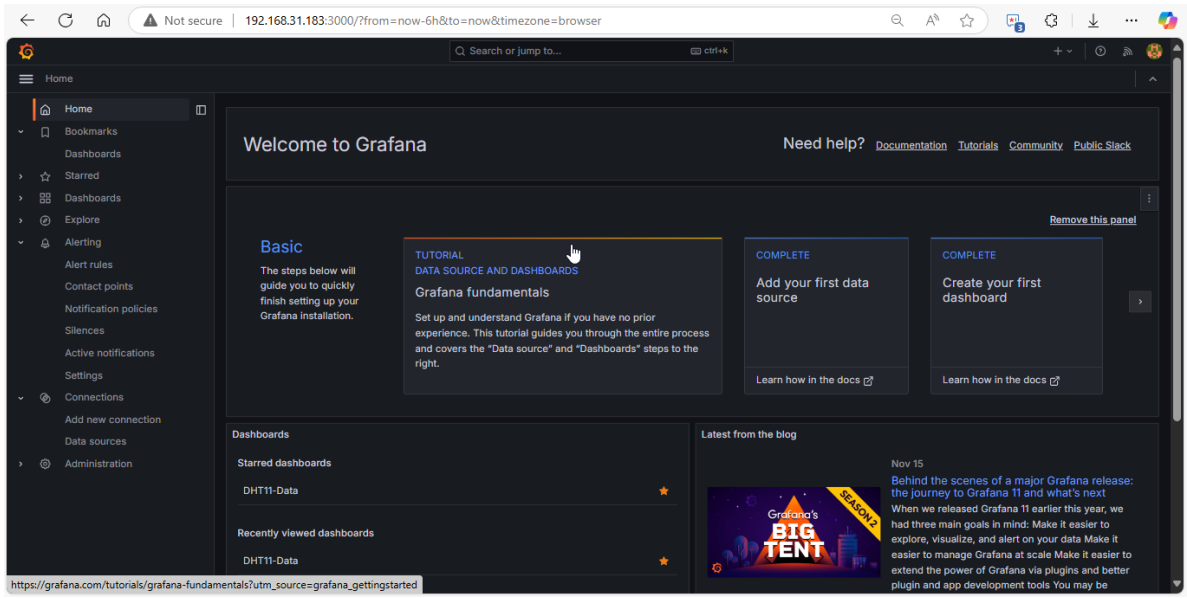

Photo 7. 10 Grafana Welcome Page

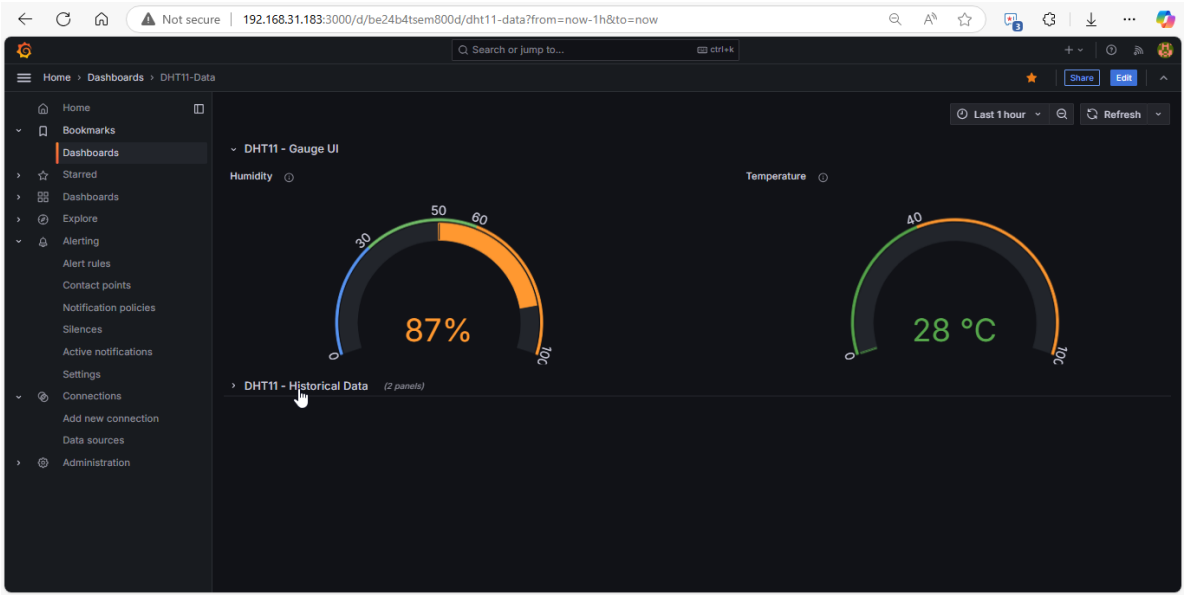

Photo 7. 11 Grafana Dashboard - Guage

Photo 7. 12 Grafana Dashboard - Historical Data View

CHAPTER 8

CONCLUSION

The IoT-Based Environmental Monitoring System using Raspberry Pi and Docker provides an efficient and scalable solution for real-time environmental data collection, processing, and visualization. Through the integration of components like the DHT11 sensor, Raspberry Pi Pico W, MQTT broker, Influx-DB, and Grafana, the system is capable of monitoring key environmental parameters such as temperature and humidity.

This project successfully demonstrates the potential of Internet of Things (IoT) technology in environmental monitoring, offering the ability to collect, store, and visualize data remotely. The system is designed for easy scalability, allowing the addition of more sensors or devices as needed. The use of Docker and microservices ensures that the system can be deployed and managed in a containerized environment, promoting ease of maintenance and flexibility.

In conclusion, this system represents a practical, cost-effective solution for monitoring environmental conditions, with applications in smart homes, agriculture, and industrial environments. It also highlights the importance of seamless integration between hardware and software to create a fully functional IoT ecosystem. Future work can include adding more sensors, improving system performance, and enhancing data visualization features.

CHAPTER 9

FURTHER ENHANCEMENTS

While the current IoT-Based Environmental Monitoring System is functional and provides valuable insights, several enhancements can be made to improve its capabilities, scalability, and usability. Some potential improvements include:

9.1 MULTI-SENSOR INTEGRATION

Expand the system to support additional environmental sensors such as gas sensors, air quality sensors, and light sensors. This would allow for more comprehensive monitoring of the environment.

9.2 REAL-TIME ALERTS AND NOTIFICATIONS

Integrate an alerting system that can notify users via email, SMS, or mobile apps when the monitored parameters exceed defined thresholds (e.g., high temperature or humidity). This would improve the system's ability to trigger immediate action in critical situations.

9.3 EDGE COMPUTING

Implement edge computing capabilities on the Raspberry Pi to process and analyze sensor data locally before sending it to the cloud. This would reduce latency and bandwidth usage while enabling faster decision-making at the edge.

9.4 MACHINE LEARNING INTEGRATION

Incorporate machine learning models to predict environmental trends based on historical data. For instance, the system could forecast temperature changes or humidity patterns, offering proactive measures for environmental control.

9.5 ENERGY EFFICIENCY

Improve the system's power consumption, especially for remote deployments, by optimizing energy usage or integrating low-power communication protocols (e.g., LoRaWAN) for long-range, low-energy data transmission.

9.6 MOBILE APPLICATION

Develop a mobile application to enable real-time monitoring and visualization of environmental data, providing users with easy access to system statistics and control settings from anywhere.

9.7 DATA SECURITY AND ENCRYPTION

Enhance security by implementing encryption for data transmission and storage, ensuring that sensitive environmental data is protected from unauthorized access or tampering.

9.8 CLOUD INTEGRATION

Expand the system to support integration with cloud platforms like AWS, Google Cloud, or Microsoft Azure for more robust data storage, processing, and analytics.

By incorporating these enhancements, the system can evolve into a more advanced and versatile solution suitable for a wide range of applications, from smart cities to industrial automation.

CHAPTER 10

REFERENCES

10.1 BOOKS AND ARTICLES

- **"Internet of Things: A Hands-On Approach"** by Arshdeep Bahga, Vijay Madisetti
    a. This book provides a comprehensive introduction to the Internet of Things (IoT) and includes hands-on examples using Raspberry Pi and other hardware. It's a great reference for anyone working on IoT-based projects.
- **"Raspberry Pi User Guide"** by Eben Upton and Gareth Halfacree
    a. A detailed guide to using the Raspberry Pi, including setup, configuration, and interfacing with sensors like the DHT11.
- **"Docker: Up & Running"** by Karl Matthias and Sean P. Kane
    a. This book covers Docker's functionality, which is crucial for containerizing your applications and managing them on the Raspberry Pi.

10.2 ONLINE RESOURCES AND DOCUMENTATION

- **Raspberry Pi Documentation**
    a. https://www.raspberrypi.org/documentation/
    b. The official Raspberry Pi documentation, including information on how to use the Raspberry Pi Pico W and connect sensors like the DHT11.
- **MicroPython Documentation**
    a. https://docs.micropython.org/
    b. Detailed documentation for MicroPython, which is used for programming the Raspberry Pi Pico W.
- **InfluxDB Documentation**
    a. https://www.influxdata.com/docs/
    b. InfluxDB's official documentation, which provides an overview of time-series data management and how to interact with it.
- **Grafana Documentation**
    a. https://grafana.com/docs/
    b. The official Grafana documentation for setting up dashboards and querying data from InfluxDB.
- **Eclipse Mosquitto MQTT Documentation**
    a. https://mosquitto.org/documentation/
    b. Eclipse Mosquitto’s official documentation on configuring and using the MQTT broker.
- **Docker Documentation**
    a. https://docs.docker.com/
    b. The official Docker documentation for managing containers and deploying applications on Docker.

- **Node-RED Documentation**
    a. https://nodered.org/docs/
    b. A guide for using Node-RED, a tool for wiring together hardware devices, APIs, and online services.

10.3 RESEARCH PAPERS AND ARTICLES

- **"IoT-Based Environmental Monitoring and Control System: A Case Study"** by R. S. K. Chandra, S. R. S. R. Kumar
    a. This paper discusses an IoT-based environmental monitoring system and includes examples of sensor integration, data collection, and cloud-based visualization.
- **"An IoT-based Smart Agriculture Monitoring System for Precision Farming"** by M. Hossain, A. A. Rahim, and A. M. S. Kazi
    a. Explores IoT-based environmental monitoring and its applications in smart agriculture.
- **"Data Security and Privacy in Internet of Things (IoT)"** by Mohammad Ali Hamade
    a. Discusses security and privacy concerns related to IoT, particularly in environmental monitoring systems.

10.4 WEBSITES AND BLOGS

- **Adafruit Learning System**
    a. https://learn.adafruit.com/
    b. A collection of tutorials related to Raspberry Pi, sensors, and other hardware components, including those used in environmental monitoring.
- **Hackster.io**
    a. https://www.hackster.io/
    b. A community-driven platform with numerous IoT-based projects, including environmental monitoring systems.
- **Medium - IoT Projects**
    a. https://medium.com/
    b. Various blogs and tutorials by IoT developers and enthusiasts, providing insights into building IoT projects with sensors and cloud platforms.

CHAPTER 11

APPENDICES

11.1 USER DOCUMENTATION

This section outlines the necessary steps to set up and use the **IoT-Based Environmental Monitoring System**.

11.1.1 INSTALLATION INSTRUCTIONS

PREREQUISITES:

- Raspberry Pi 5 (or similar version) with Raspbian OS installed.
- Raspberry Pi Pico W.
- DHT11 sensor.
- Docker, Portainer, and Node-RED installed.
- InfluxDB, Grafana, Eclipse Mosquitto MQTT broker, and MQTT client.
- Basic understanding of terminal commands and network configuration.

STEP 1: HARDWARE SETUP

1. **Connect the DHT11 Sensor to Raspberry Pi Pico W:**
    - Connect the **VCC** pin of the DHT11 to the **3V3** pin on the Raspberry Pi Pico W.
    - Connect the **GND** pin of the DHT11 to the **GND** pin on the Raspberry Pi Pico W.
    - Connect the **DATA** pin to **GPIO 15** on the Raspberry Pi Pico W.
2. **Connect the Raspberry Pi Pico W to your Raspberry Pi:**
    - Connect the Raspberry Pi Pico W via USB to the Raspberry Pi 5. The Raspberry Pi 5 will act as the central hub for data collection, MQTT communication, and dashboard hosting.

STEP 2: SOFTWARE INSTALLATION

1. **Install MicroPython on Raspberry Pi Pico W:**
    - Download and install the latest version of MicroPython from the official MicroPython website.
    - Follow the instructions for flashing the firmware onto your Raspberry Pi Pico W.
2. **Set Up the MQTT Broker (Eclipse Mosquitto):**
    - On your Raspberry Pi 5, install the Mosquitto MQTT broker by running:

```
Bash Command:
sudo apt-get update
sudo apt-get install mosquitto mosquitto-clients
```

- Configure Mosquitto by editing /etc/mosquitto/mosquitto.conf if needed.

3. **Install Docker & Portainer:**
   - Install Docker on the Raspberry Pi 5:

     Bash Command:
     ```
     curl -sSL https://get.docker.com | sh
     sudo usermod -aG docker pi
     sudo systemctl enable docker
     sudo systemctl start docker
     ```
   - Install Portainer (a GUI for Docker management):

     Bash Command:
     ```
     docker volume create portainer_data
     docker run -d -p 9000:9000 --name portainer --restart always \
       -v /var/run/docker.sock:/var/run/docker.sock -v
     portainer_data:/data portainer/portainer-ce
     ```
4. **Install InfluxDB and Grafana (for Data Storage & Visualization):**
   - Install InfluxDB:

     Bash Command:
     ```
     sudo apt-get install influxdb
     sudo systemctl start influxdb
     sudo systemctl enable influxdb
     ```
   - Install Grafana:

     Bash Command:
     ```
     sudo apt-get install -y software-properties-common
     sudo add-apt-repository "deb https://packages.grafana.com/oss/deb
     stable main"
     sudo apt-get update
     sudo apt-get install grafana
     sudo systemctl enable grafana-server
     sudo systemctl start grafana-server
     ```
5. **Install Node-RED for Automation (optional for workflows):**
   - Install Node-RED:

     Bash Command:
     ```
      <(curl -sL https://nodered.org/setup_20.x)
     ```
6. **Deploy Docker Containers for Grafana, InfluxDB, Mosquitto, and Node-RED (optional):**
   - Using Docker, you can easily deploy containers for InfluxDB, Grafana, and Mosquitto. For example, to run InfluxDB in Docker:

Bash Command:

docker run -d --name influxdb -p 8086:8086 influxdb:latest

- Similarly, deploy Grafana and Mosquitto containers.

STEP 3: CONFIGURE THE SYSTEM

1. **Configure MicroPython on Raspberry Pi Pico W:**
   - Write a script for your Raspberry Pi Pico W to read data from the DHT11 sensor, connect to Wi-Fi, and send the data via MQTT to the broker. Upload this script using a tool like **Thonny** (Python IDE).
2. **Set Up Grafana:**
   - Access Grafana through http://<RaspberryPi-IP>:3000 and configure it to connect to InfluxDB for data visualization.
   - Create dashboards to visualize real-time sensor data.
3. **Configure MQTT Topics:**
   - Define MQTT topics for sending and receiving sensor data. Configure the MQTT client on the Raspberry Pi Pico W to publish sensor readings to the MQTT broker.

11.2 README

This section provides the basic information about the project, including the system's purpose and usage.

Project Name: **IoT-Based Environmental Monitoring System**

Description:

This system collects temperature and humidity data from a DHT11 sensor using a Raspberry Pi Pico W. It sends the data to an MQTT broker, which forwards it to InfluxDB for storage. The data is then visualized in real-time using Grafana.

Components Used:

- Raspberry Pi 5
- Raspberry Pi Pico W
- DHT11 Temperature and Humidity Sensor
- Mosquitto MQTT Broker
- InfluxDB (Time-series Database)
- Grafana (Visualization)
- Docker and Portainer (For containerized services)
- Node-RED (For workflow automation, optional)

Key Features:

- Real-time environmental monitoring of temperature and humidity.
- Data storage in InfluxDB for efficient time-series data management.
- Visualization of sensor data using Grafana dashboards.
- MQTT-based communication for seamless data transmission.

11.2.1 CLIENT MODULE GUIDE

Client Module Overview:

The client module is responsible for reading sensor data, processing it, and sending it to the MQTT broker for transmission to the server. This module runs on the Raspberry Pi Pico W.

Steps for Client Module Setup:

1. **Sensor Data Collection**:
    - The DHT11 sensor collects temperature and humidity data at regular intervals.
2. **Wi-Fi Configuration**:
    - The Raspberry Pi Pico W connects to the local Wi-Fi network for data transmission. You need to input your SSID and password in the configuration script.
3. **Data Transmission via MQTT**:
    - The data collected by the sensor is packaged in JSON format and published to the MQTT broker.
    - The MQTT topic is defined as environment/data.
4. **Script Example**:
    - The MicroPython script running on the Raspberry Pi Pico W collects sensor data and sends it via MQTT. An example script is provided in the client.py file.

Running the Client Module:

1. Open the terminal on the Raspberry Pi.
2. Navigate to the directory where the client.py script is saved.
3. Run the script using:

Bash Command:

```
python client.py
```

Expected Output:

- The client module will begin collecting data from the DHT11 sensor and publishing it to the MQTT broker.
- The MQTT client will log messages confirming the transmission of sensor data.

Troubleshooting:

- Ensure that the Raspberry Pi Pico W is correctly connected to the Wi-Fi network.
- Verify the MQTT broker is running and the topic name is correctly configured in the client script.