Project Report

Connecting Sensors to ESP32 for MQTT Data to Grafana via Docker on a Linux Server

Dr. Mudasar Latif Memon
1st September 2023

1. Objective:

To set up an IoT system using an ESP32 to collect temperature, humidity, and distance data from sensors, push it to a Grafana dashboard via MQTT, and employ Docker for managing four containers. The data transmission occurs to a Linux server using a local hotspot for connectivity.

2. Requirements:

- ESP32 microcontroller
- Temperature and humidity sensor (e.g., DHT22)
- Ultrasonic sensor (e.g., HC-SR04)
- MQTT broker (e.g., Mosquitto)
- Grafana installation
- Docker installed on the Linux server
- Linux operating system
- Local hotspot for data transmission

3. Procedure:

a) Hardware Setup:

Connect the temperature and humidity sensor (DHT22) to the ESP32 following the datasheet instructions.

Connect the ultrasonic sensor (HC-SR04) to the ESP32, typically requiring two GPIO pins for trigger and echo.

Ensure proper power and ground connections for both sensors and the ESP32.

The link to the video of hardware connectivity can be seen here

https://drive.google.com/drive/u/1/folders/1nxSVwyzRIeo5hr2NL-CwRgtj9hdhhxNL

b) Programming for ESP-32:

```
# ESP32 MicroPython example code (use uPyCraft or Thonny IDE)

# Import necessary libraries

import machine

import time

from dht import DHT22

import urequests as requests  # for HTTP requests to Grafana


# Configure DHT22 sensor

dht_pin = machine.Pin(4)  # GPIO4

dht = DHT22(dht_pin)


# Ultrasonic sensor configuration
```

```
trigger_pin = machine.Pin(2)  # GPIO2
echo_pin = machine.Pin(15)    # GPIO15

def read_distance():
    trigger_pin.value(0)
    time.sleep_us(2)
    trigger_pin.value(1)
    time.sleep_us(10)
    trigger_pin.value(0)
    while echo_pin.value() == 0:
        pass
    t1 = time.ticks_us()
    while echo_pin.value() == 1:
        pass
    t2 = time.ticks_us()
    return (t2 - t1) / 58

while True:
    try:
        # Read temperature and humidity
        dht.measure()
```

```
    temperature = dht.temperature()
    humidity = dht.humidity()

    # Read distance
    distance = read_distance()

    # Send data to MQTT broker
    # Use an MQTT library suitable for MicroPython, e.g., umqtt.simple

    # Send data to Grafana via HTTP (Grafana Alerting)
    grafana_url = "http://your_grafana_server:port/api/annotations"
    headers = {"Authorization": "Bearer your_api_token"}
    data = {
        "text": f"Temperature: {temperature}°C, Humidity: {humidity}%, Distance: {distance} cm",
        "tags": ["ESP32", "Sensor Data"]
    }
    response = requests.post(grafana_url, json=data, headers=headers)
    response.close()

    time.sleep(300)  # Delay for 5 minutes
```

```
except Exception as e:
    print("Error:", e)
```

c) MQTT Broker Setup:

- Install Mosquitto MQTT broker on your Linux server.
- Configure Mosquitto to listen on a specific port (e.g., 1883).
- Create MQTT topics for temperature, humidity, and distance data

d) Docker Containerization:

- Set up four Docker containers:
- MQTT Broker container: Install and configure Mosquitto MQTT broker inside the container.
- Grafana container: Install Grafana inside the container and configure it to connect to the MQTT broker and InfluxDB database.
- InfluxDB container: Create a container with InfluxDB to store the data from MQTT.
- ESP32 Data Relay container: Run a Python or Node.js script inside this container to receive MQTT data from the ESP32 and push it into the InfluxDB container.

e) Docker Compose:

- Create a Docker Compose YAML file to define the four containers (MQTT Broker, Grafana, InfluxDB, ESP32 Data Relay).
- Specify the container configurations, dependencies, and environment variables in the YAML file.

f) **Container Deployment:**

- Deploy the containers on the Linux server using Docker Compose: **docker-compose up -d**.
- Verify that all containers are running without errors: **docker-compose ps**.

g) **Local Hotspot Setup:**

- Create a local hotspot on your Linux server or use an existing one to establish a Wi-Fi connection with the ESP32.
- Ensure the ESP32 is configured to connect to the hotspot.

h) **Testing and Monitoring:**

- Confirm that the ESP32 is publishing data to the MQTT broker via the local hotspot.
- Verify that data is being stored in the InfluxDB database.
- Access the Grafana dashboard through a web browser to visualize the sensor data.

By following these steps and using the provided Python code for the ESP32, we can set up a comprehensive IoT system that collects temperature, humidity, and distance data from sensors, pushes it to Grafana via MQTT, utilizes Docker for efficient container management, and transmits data to a Linux server via a local hotspot for remote monitoring and visualization.

4. The Pictorial Representation

We can see the command of docker

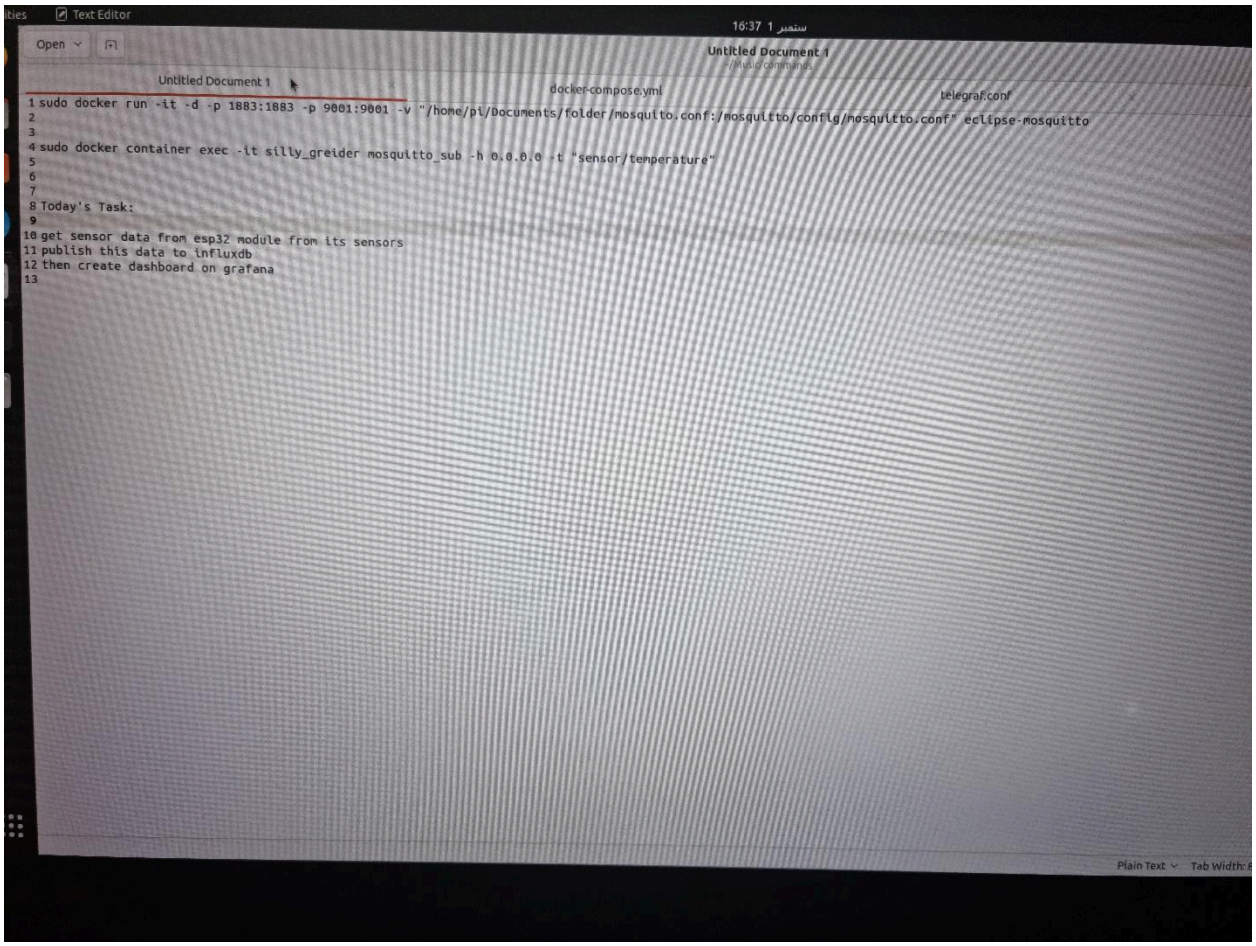

Figure 1: Docker Image

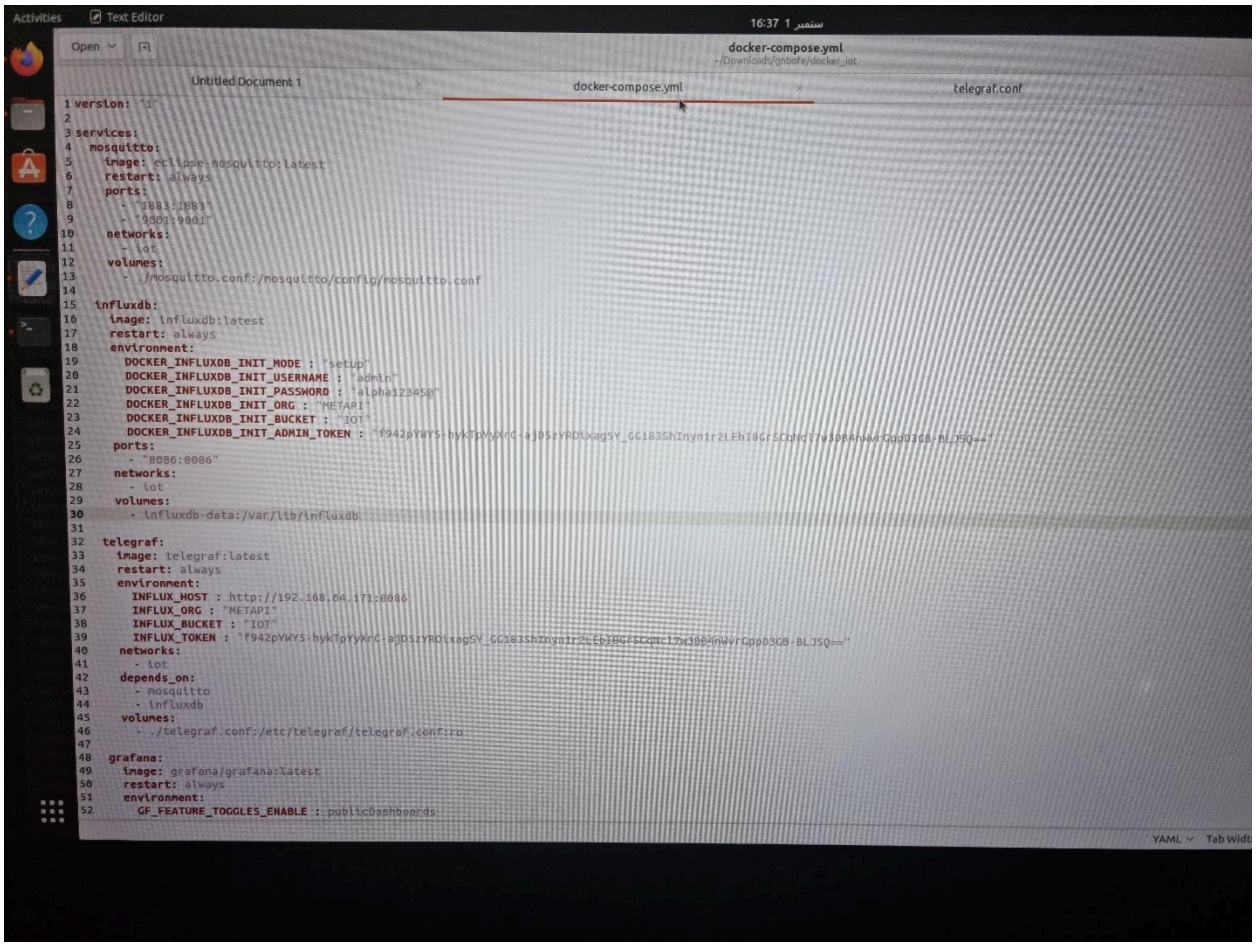

Figure 2: Docker Code Image

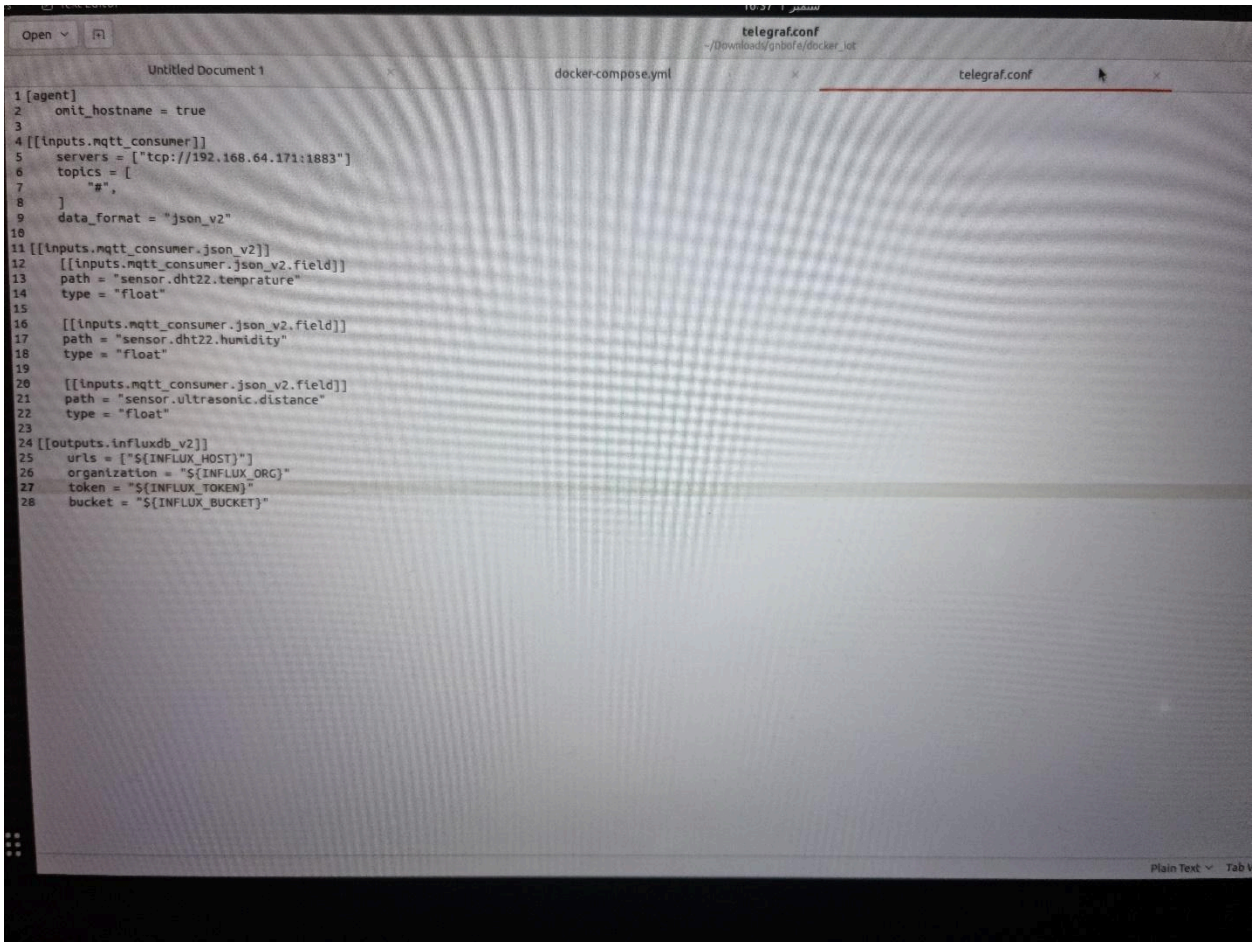

Figure 3: Telegraph Configuration File Image

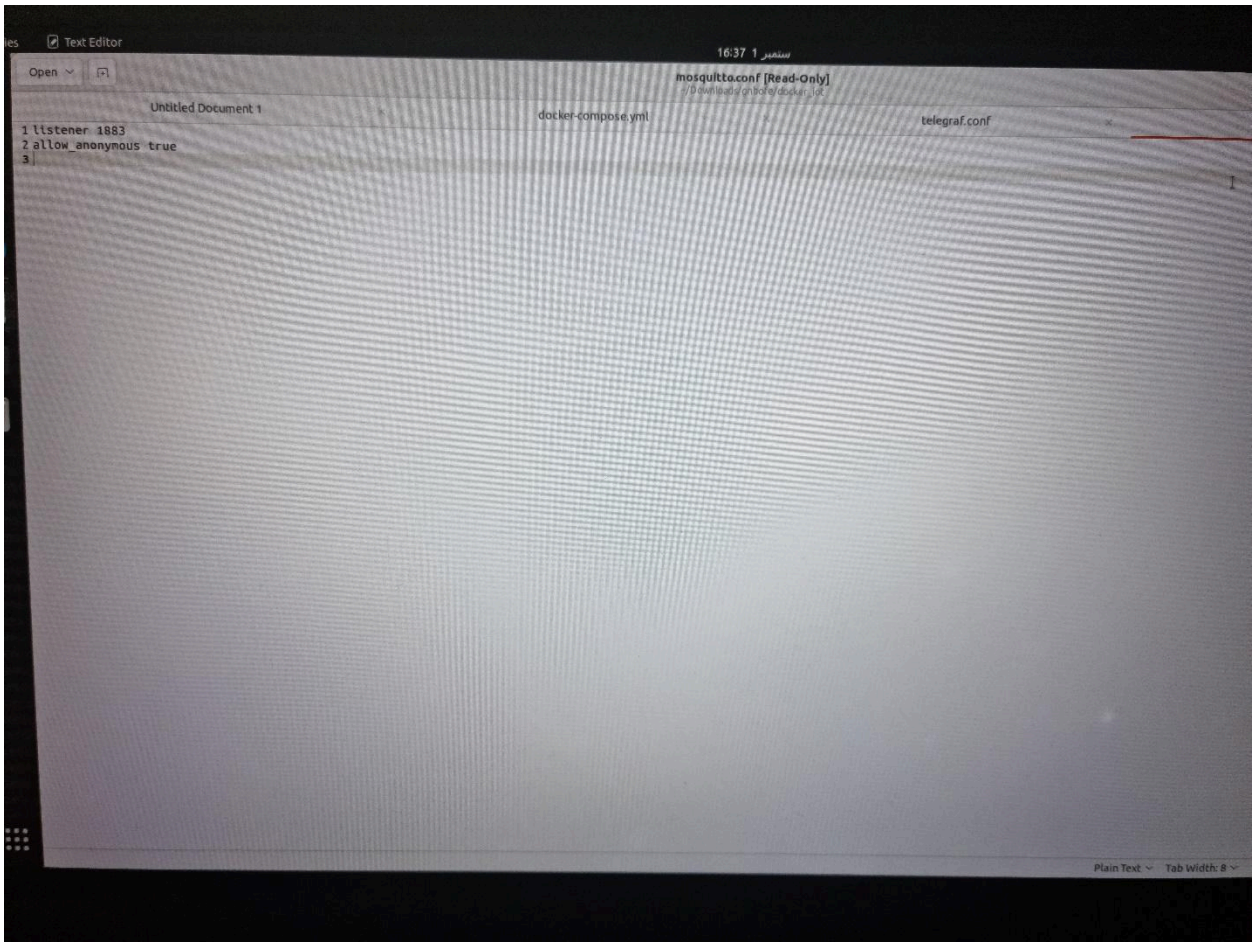

Figure 4: Mosquitto Configuration file

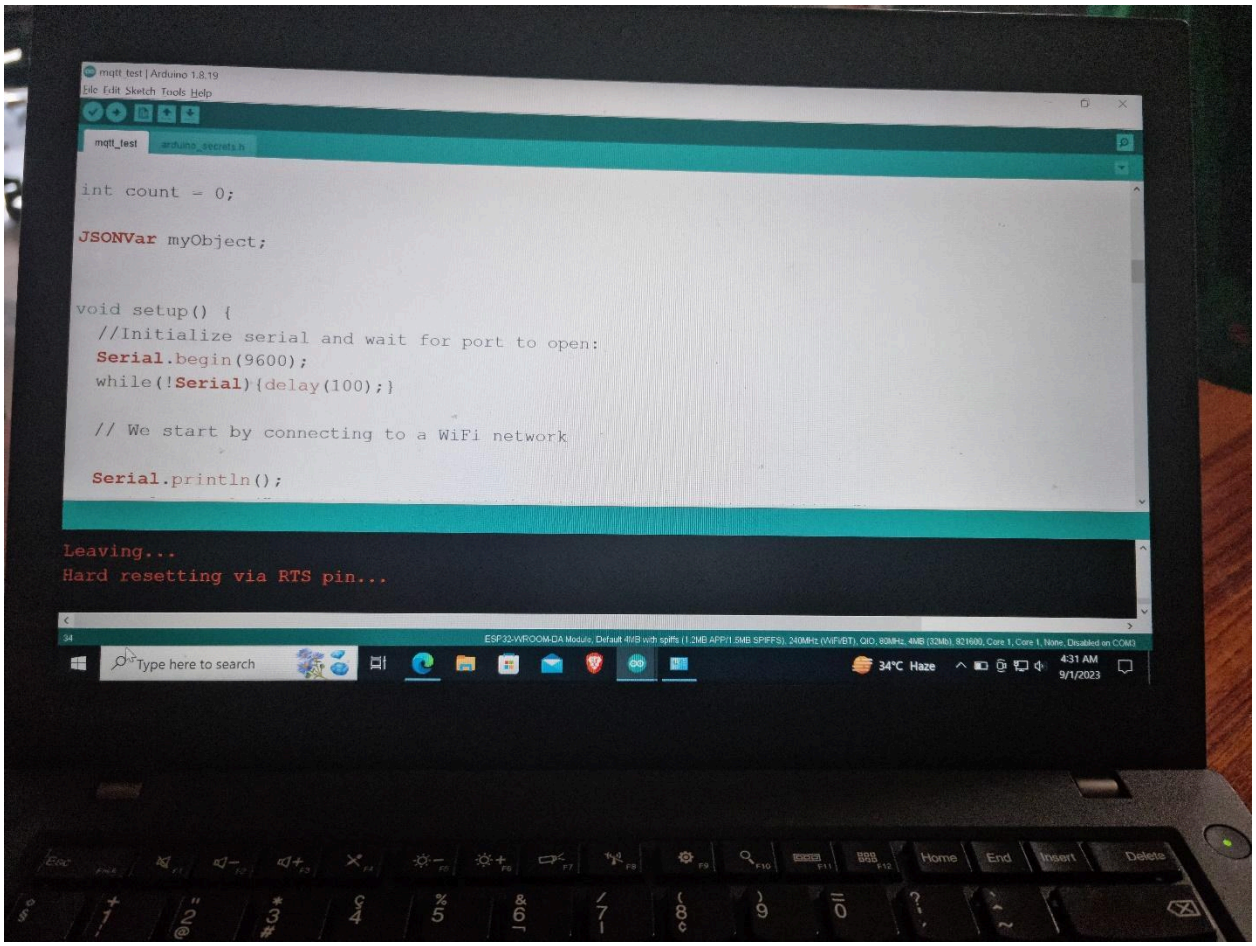

Figure 5: Image of mqtt_test code

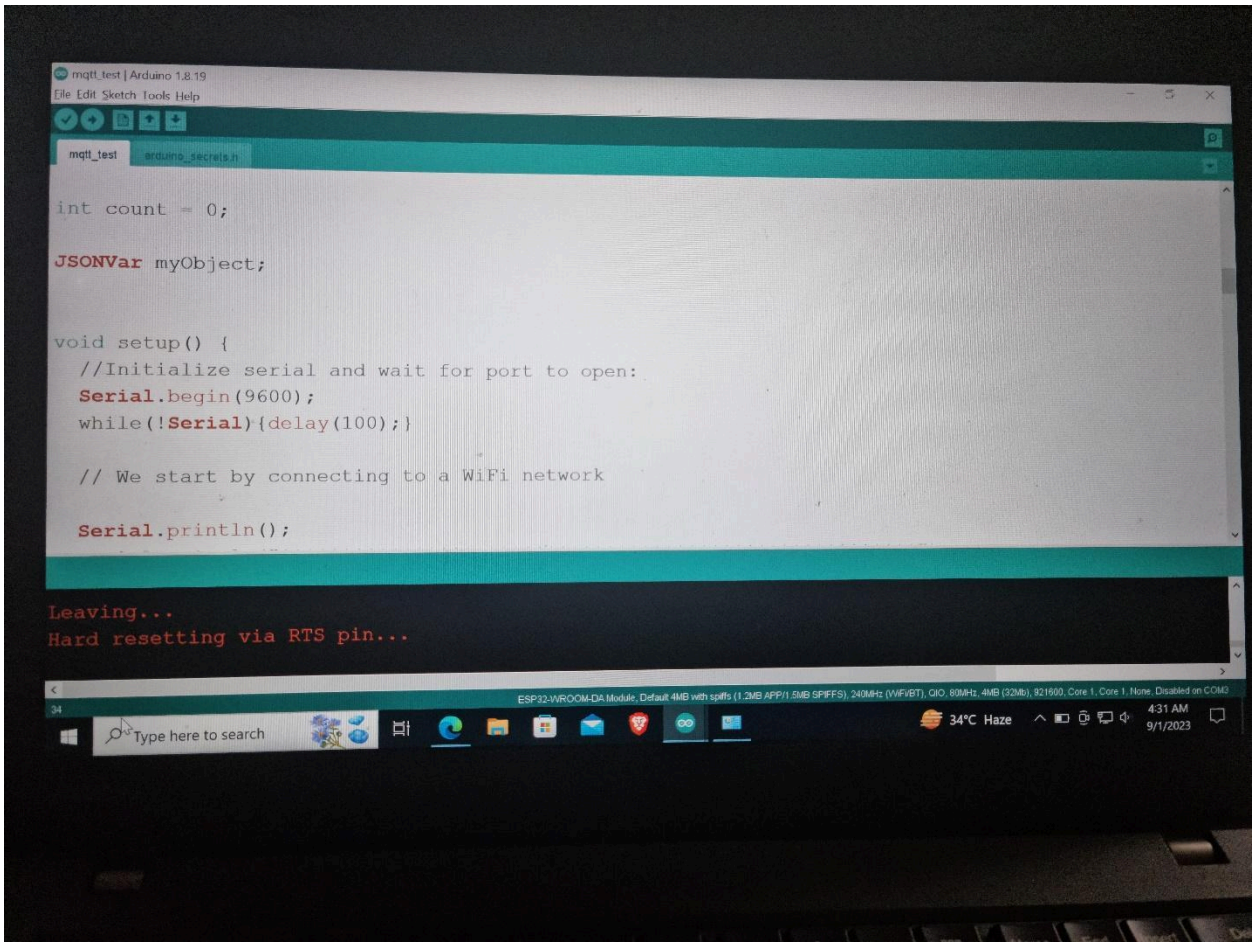

Figure 6: Baud rate selection in the mqtt_test file for Arduino

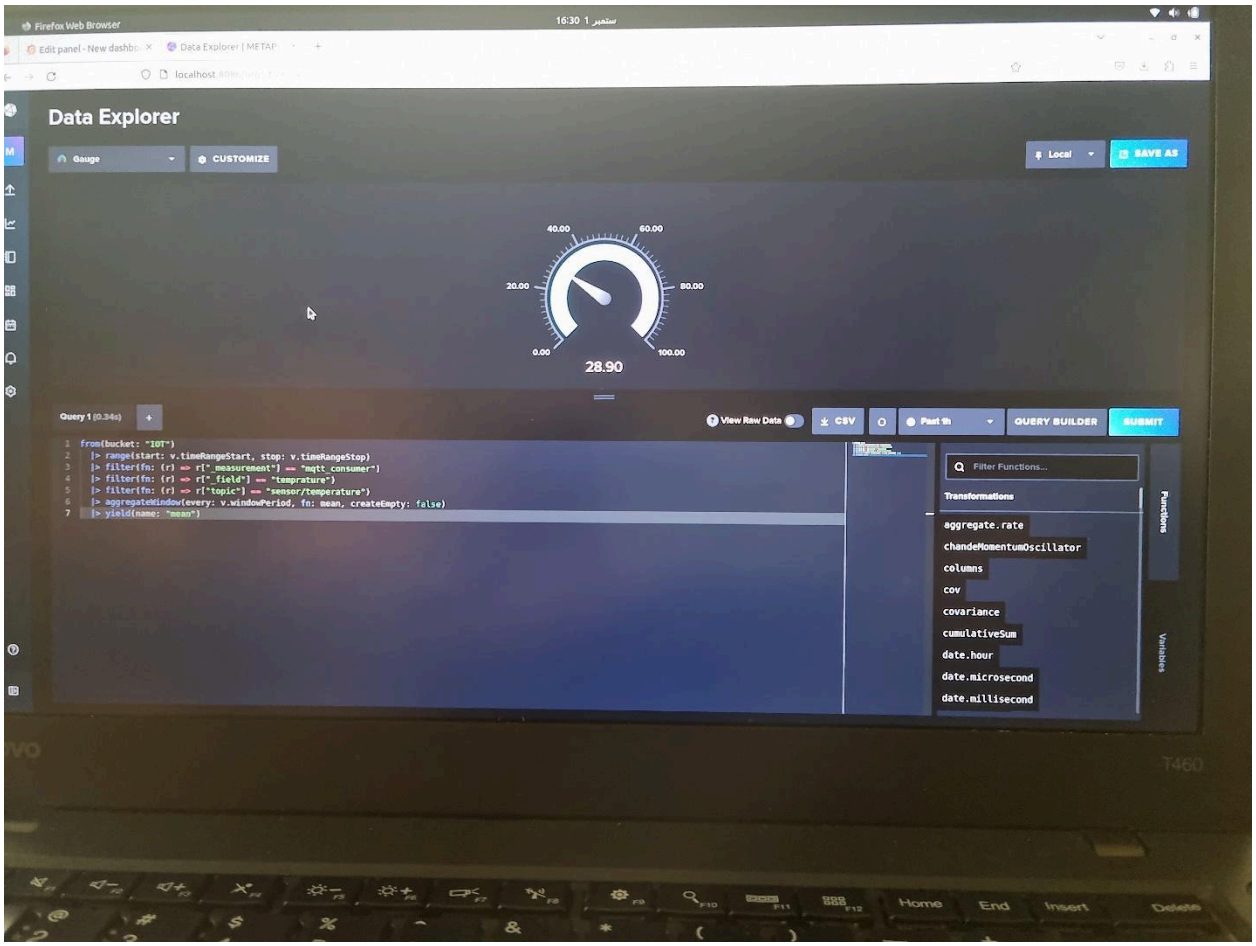

Figure 7: Dashboard view

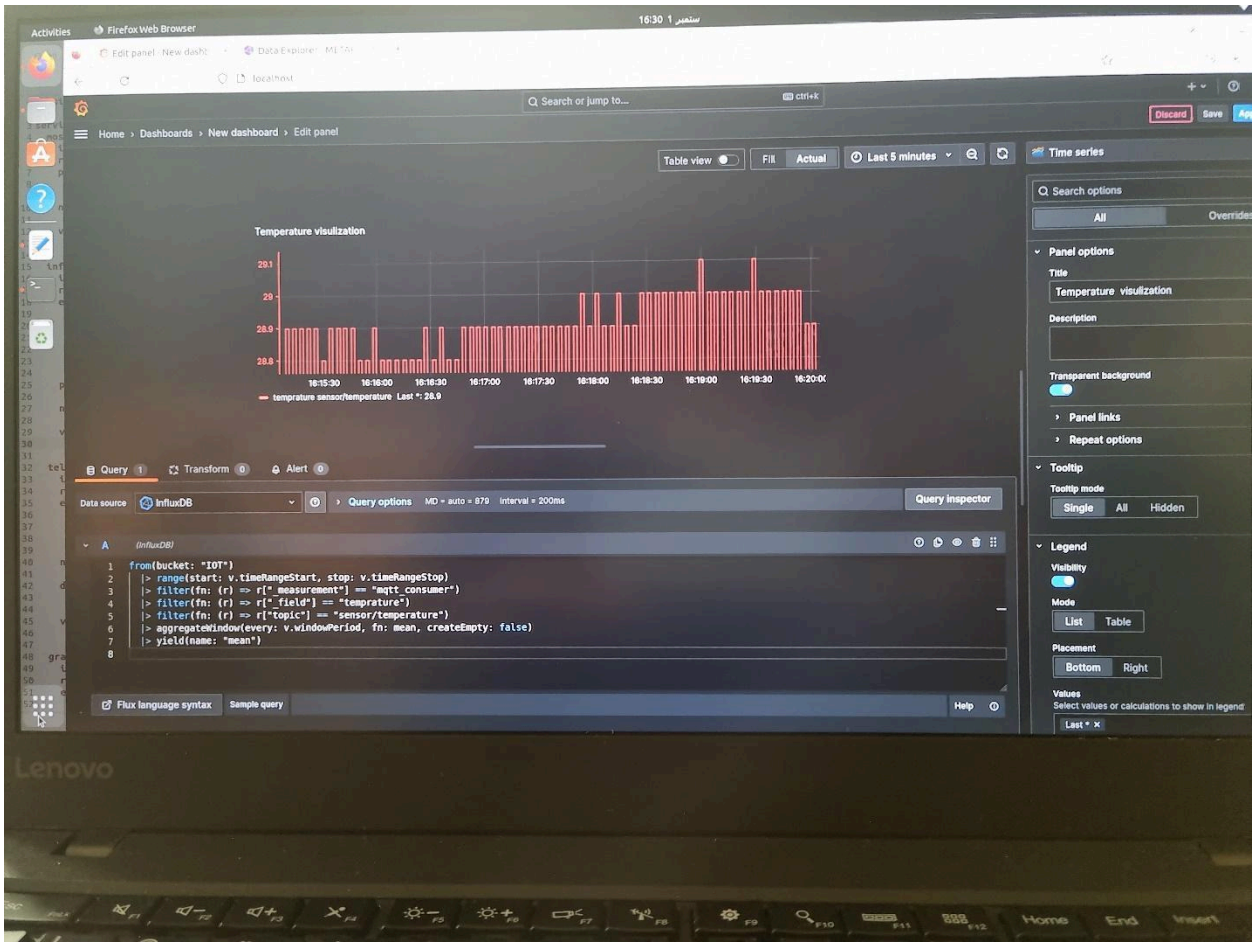

Figure 8: Graph on the dashboard showing the data is being received from containers

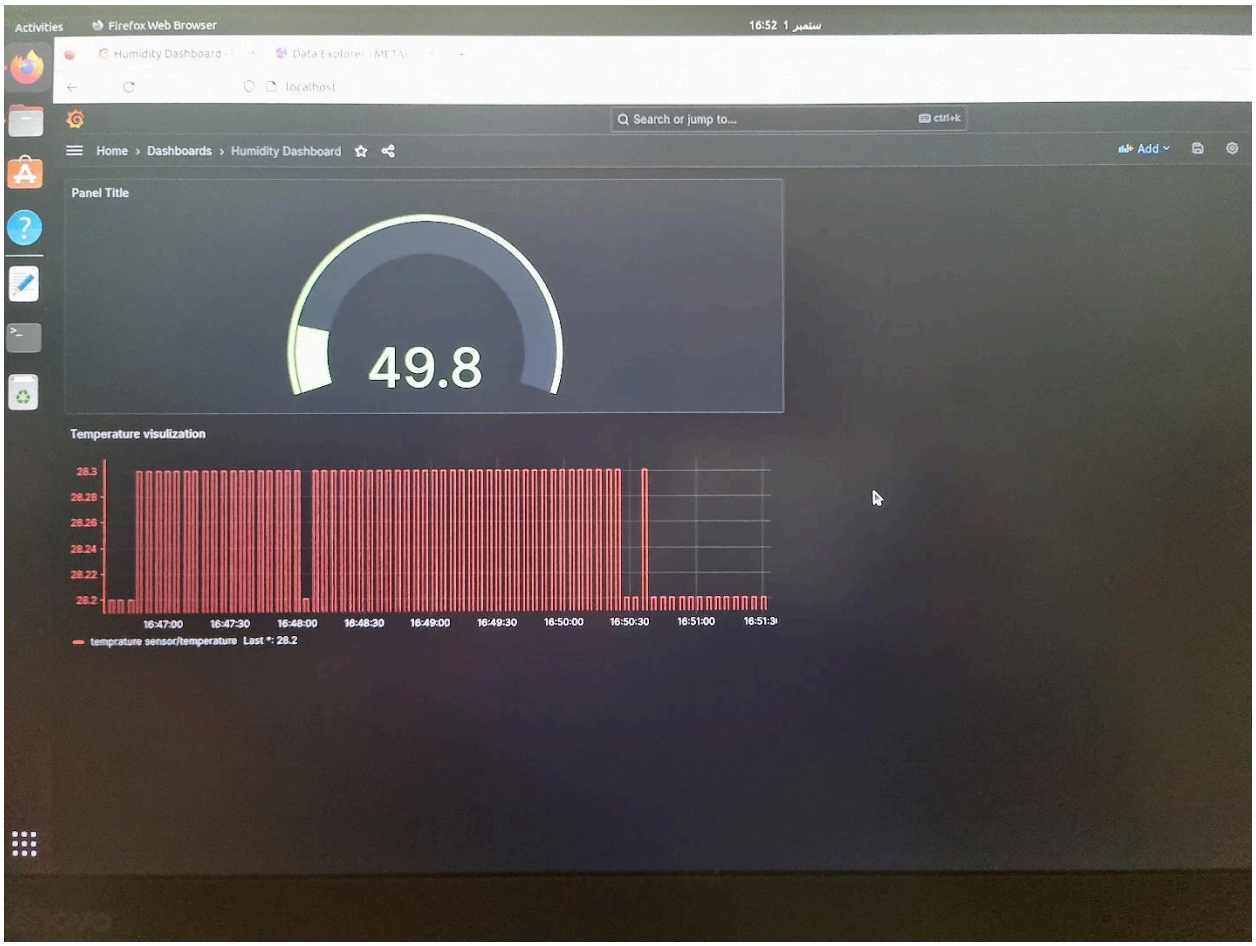

Figure 9: Dashboard View of the data being received in the dash board

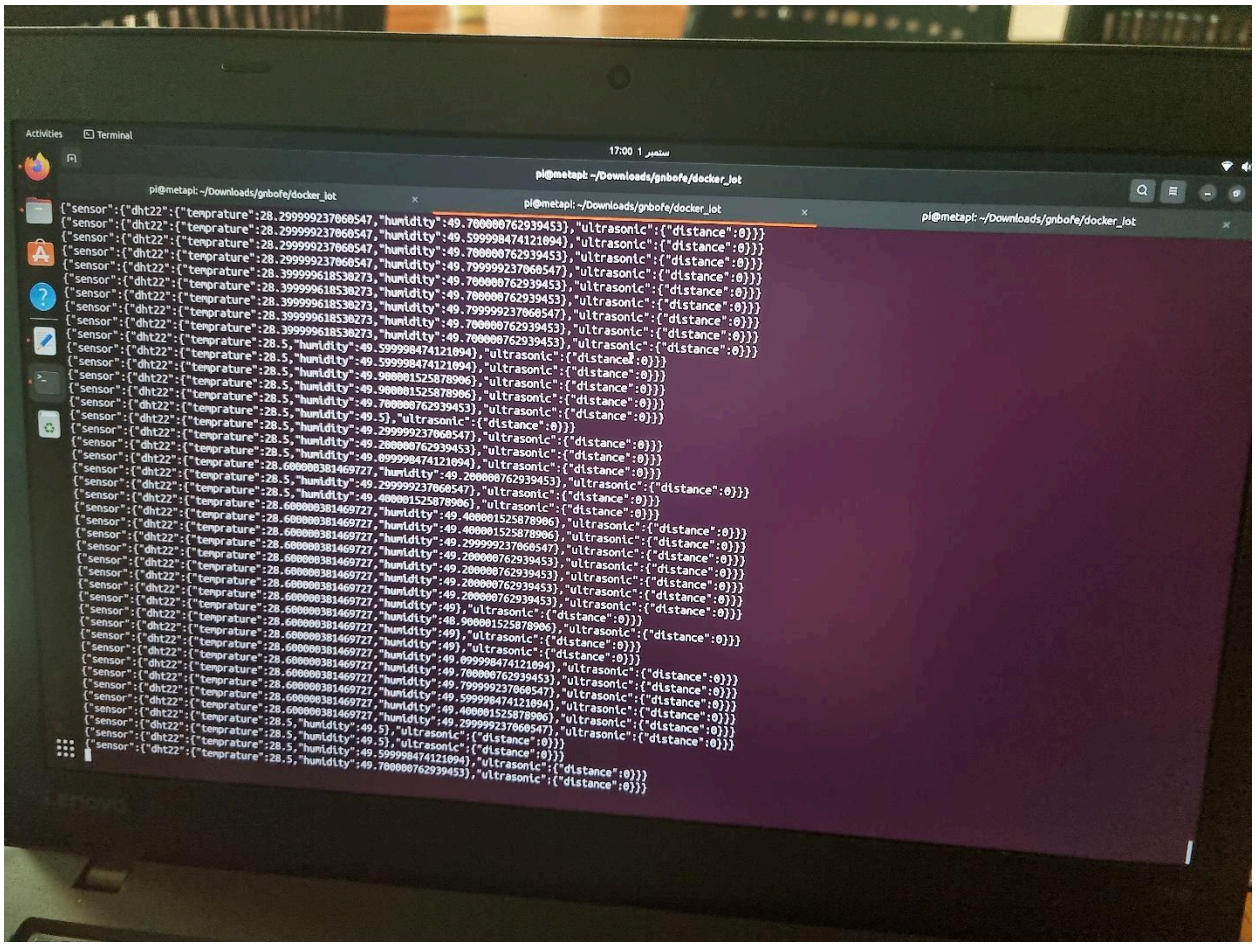

Figure 10: Sensors readings received through the Arduino

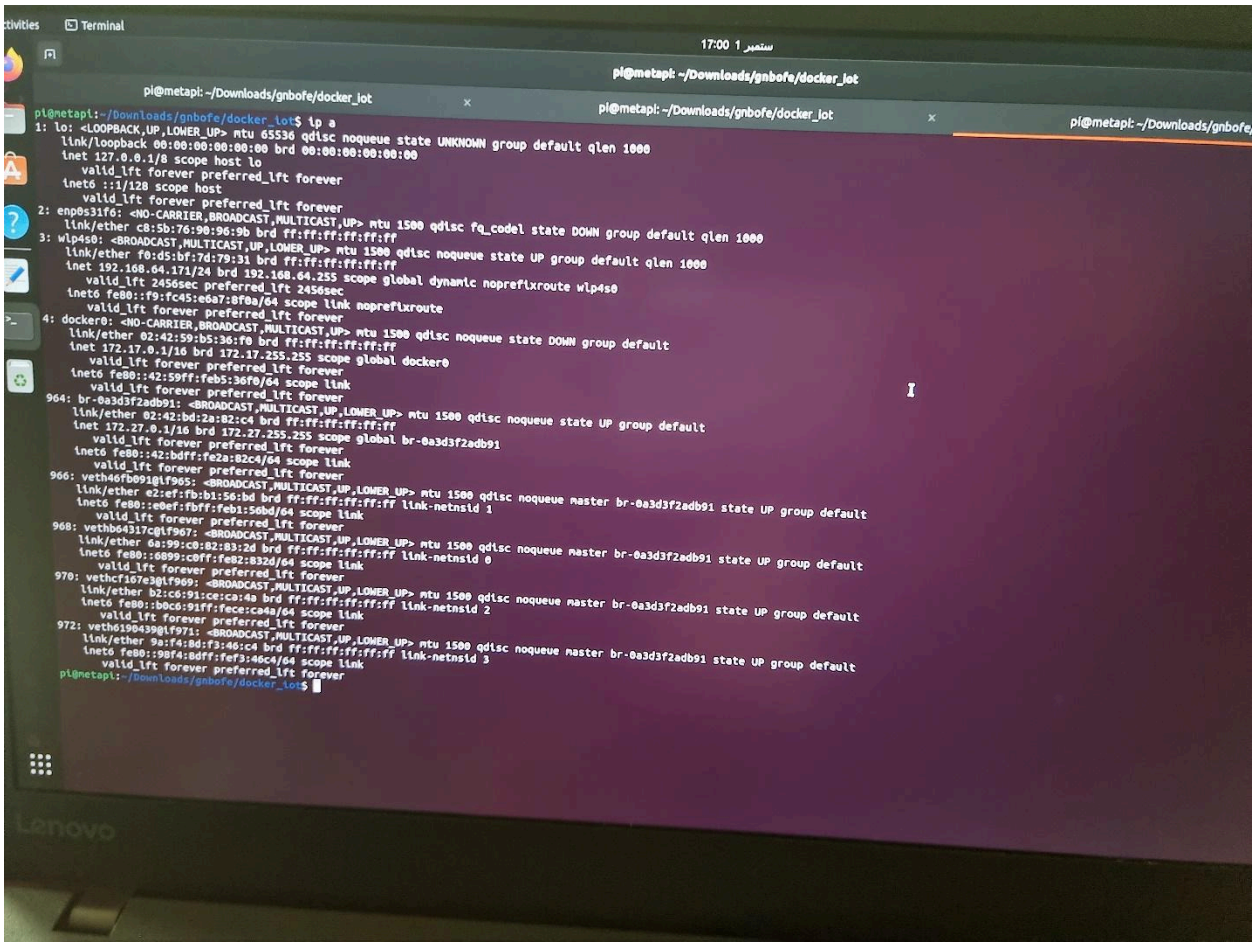

Figure 12: Sensors readings received through the Arduino (Docker IOT)

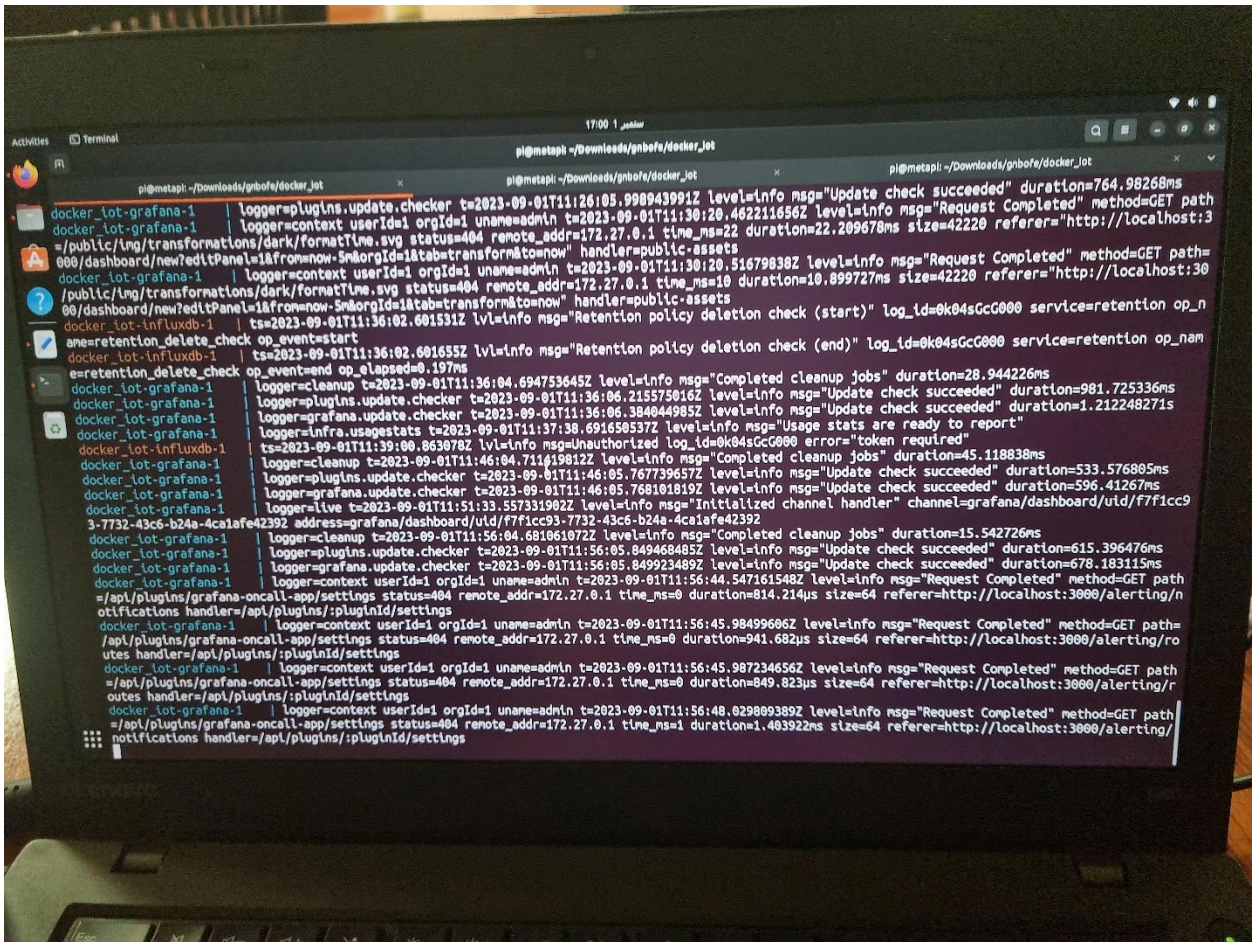

Figure 13: Sensors readings received through the Arduino (Docker IoT)

Conclusion

We have successfully set up an IoT system using an ESP32 to collect temperature, humidity, and distance data from sensors, push it to a Grafana dashboard via MQTT, and employed Docker for managing four containers. The data transmission occurs to a Linux server using a local hotspot for connectivity.

For more details visit the complete website

https://www.mlmemon.com/14-data-science