

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/1nxSVwyzRleo5hr2NL-CwRgtj9hdh_hxNL

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

```
Text Editor 16:37 1
Untitled Document 1
docker-compose.yml telegraf.conf
1 sudo docker run -it -d -p 1883:1883 -p 9001:9001 -v "/home/pi/Documents/folder/mosquitto.conf:/mosquitto.conf" eclipse-mosquitto
2
3
4 sudo docker container exec -it silly_greider mosquitto_sub -h 0.0.0.0 -t "sensor/temperature"
5
6
7
8 Today's Task:
9
10 get sensor data from esp32 module from its sensors
11 publish this data to influxdb
12 then create dashboard on grafana
13
Plain Text Tab Width: 8
```

Figure 1: Docker Image

```
1 version: '3'
2
3 services:
4   mosquitto:
5     image: eclipse/mosquitto:latest
6     restart: always
7     ports:
8       - '1883:1883'
9       - '9001:9001'
10    networks:
11      - lot
12    volumes:
13      - ./mosquitto.conf:/mosquitto/config/mosquitto.conf
14
15  influxdb:
16    image: influxdb:latest
17    restart: always
18    environment:
19      DOCKER_INFLUXDB_INIT_MODE : 'setup'
20      DOCKER_INFLUXDB_INIT_USERNAME : 'admin'
21      DOCKER_INFLUXDB_INIT_PASSWORD : 'alpha123456'
22      DOCKER_INFLUXDB_INIT_ORG : 'METAPI'
23      DOCKER_INFLUXDB_INIT_BUCKET : 'IOT'
24      DOCKER_INFLUXDB_INIT_ADMIN_TOKEN : 'f942pVWYS-hyKtPvYKcIaj05zYR0LxagSY_GG180ShInym1rZLEb18GrScqNcL7w308AhWvGpp03GB-BLJ9Q=='
25    ports:
26      - '8086:8086'
27    networks:
28      - lot
29    volumes:
30      - influxdb-data:/var/lib/influxdb
31
32  telegraf:
33    image: telegraf:latest
34    restart: always
35    environment:
36      INFLUX_HOST : http://192.168.04.171:8086
37      INFLUX_ORG : 'METAPI'
38      INFLUX_BUCKET : 'IOT'
39      INFLUX_TOKEN : 'f942pVWYS-hyKtPvYKcIaj05zYR0LxagSY_GG180ShInym1rZLEb18GrScqNcL7w308AhWvGpp03GB-BLJ9Q=='
40    networks:
41      - lot
42    depends_on:
43      - mosquitto
44      - influxdb
45    volumes:
46      - ./telegraf.conf:/etc/telegraf/telegraf.conf:ro
47
48  grafana:
49    image: grafana/grafana:latest
50    restart: always
51    environment:
52      GF_FEATURE_TOGGLES_ENABLE : publicdashboards
```

Figure 2: Docker Code Image

```
1 [agent]
2   onit_hostname = true
3
4 [[inputs.mqtt_consumer]]
5   servers = ["tcp://192.168.64.171:1883"]
6   topics = [
7     "#",
8   ]
9   data_format = "json_v2"
10
11 [[inputs.mqtt_consumer.json_v2]]
12 [[inputs.mqtt_consumer.json_v2.field]]
13   path = "sensor.dht22.temprature"
14   type = "float"
15
16 [[inputs.mqtt_consumer.json_v2.field]]
17   path = "sensor.dht22.humidity"
18   type = "float"
19
20 [[inputs.mqtt_consumer.json_v2.field]]
21   path = "sensor.ultrasonic.distance"
22   type = "float"
23
24 [[outputs.influxdb_v2]]
25   urls = ["${INFLUX_HOST}"]
26   organization = "${INFLUX_ORG}"
27   token = "${INFLUX_TOKEN}"
28   bucket = "${INFLUX_BUCKET}"
```

Figure 3: Telegraph Configuration File Image

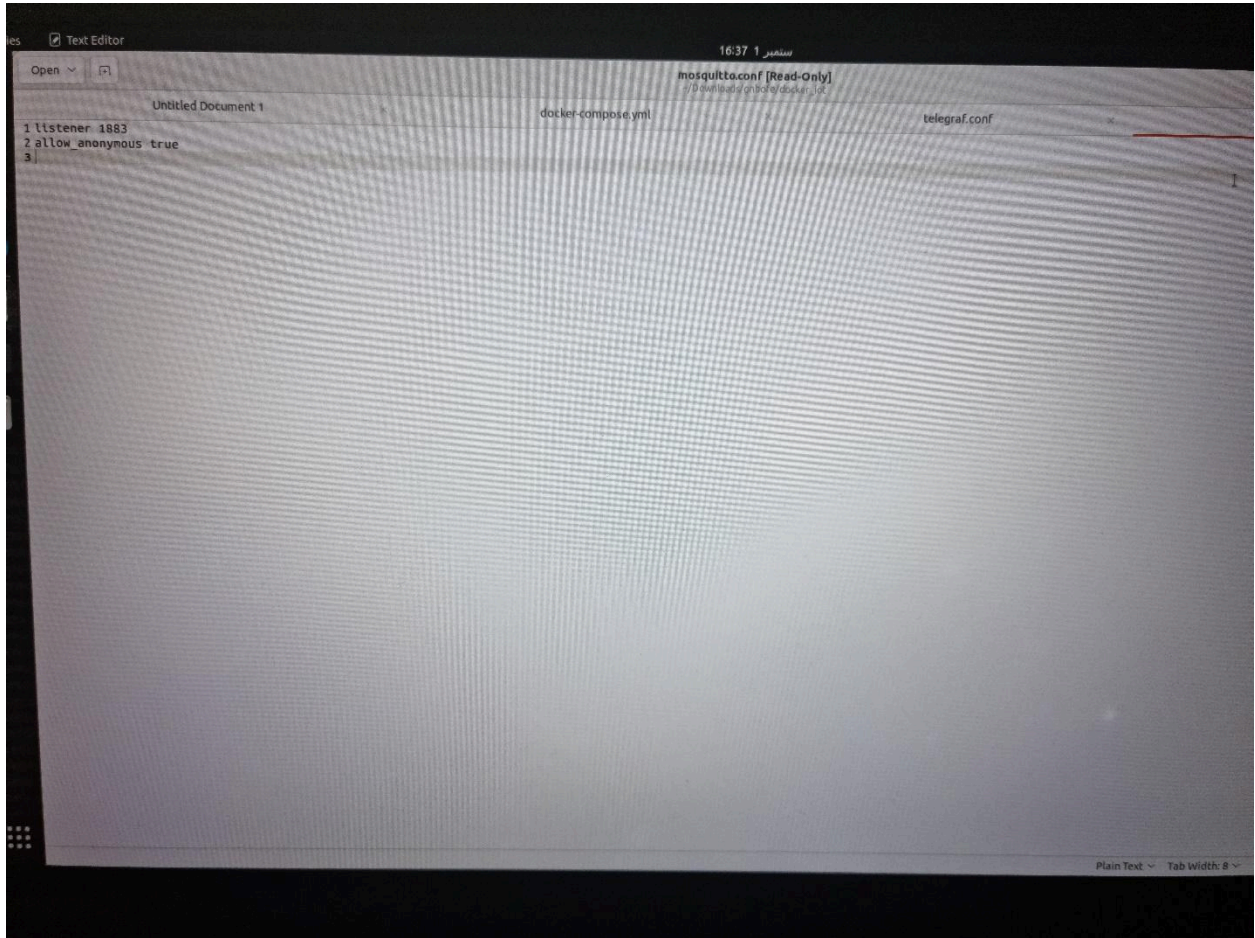


Figure 4: Mosquitto Configuration file

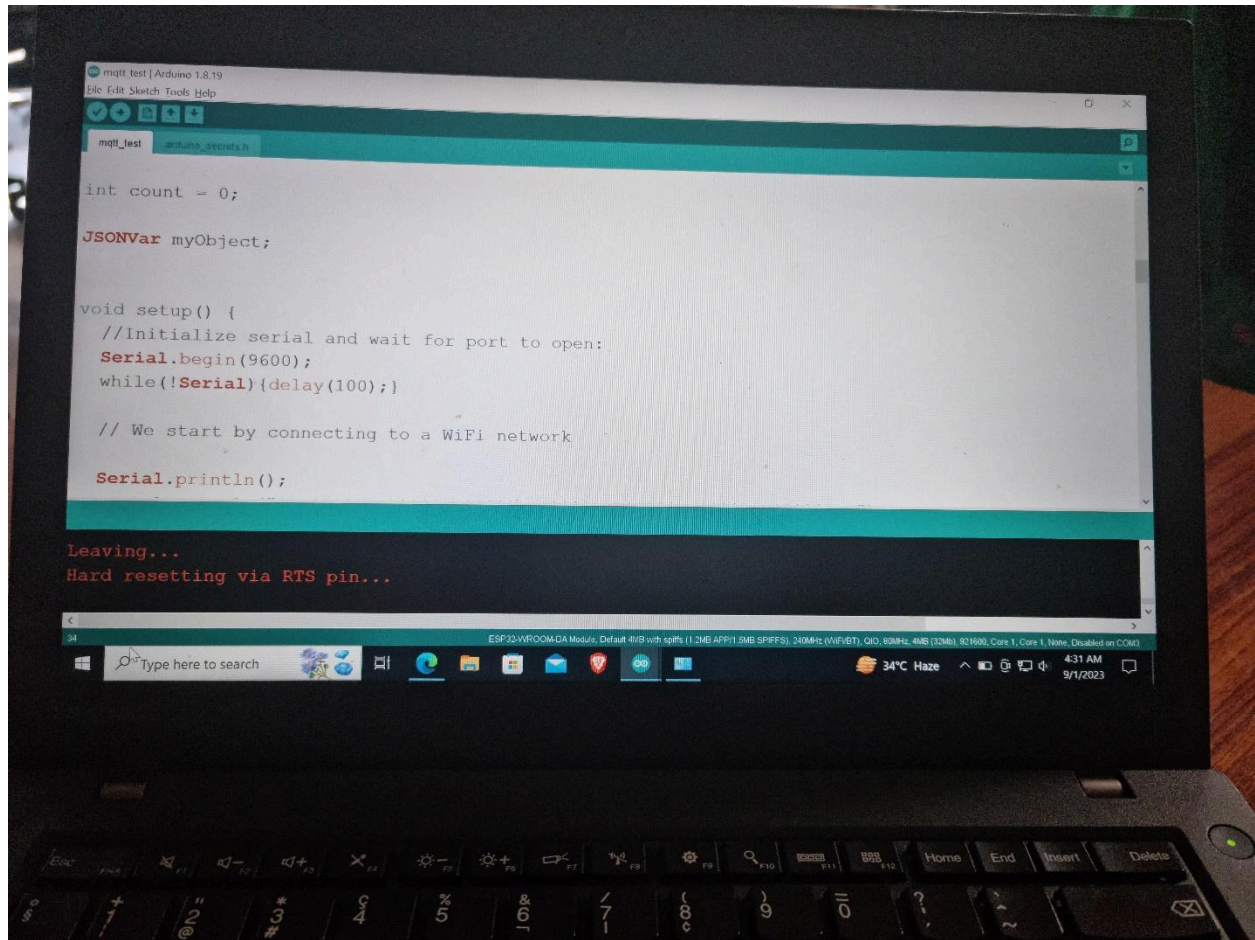


Figure 5: Image of mqtt_test code

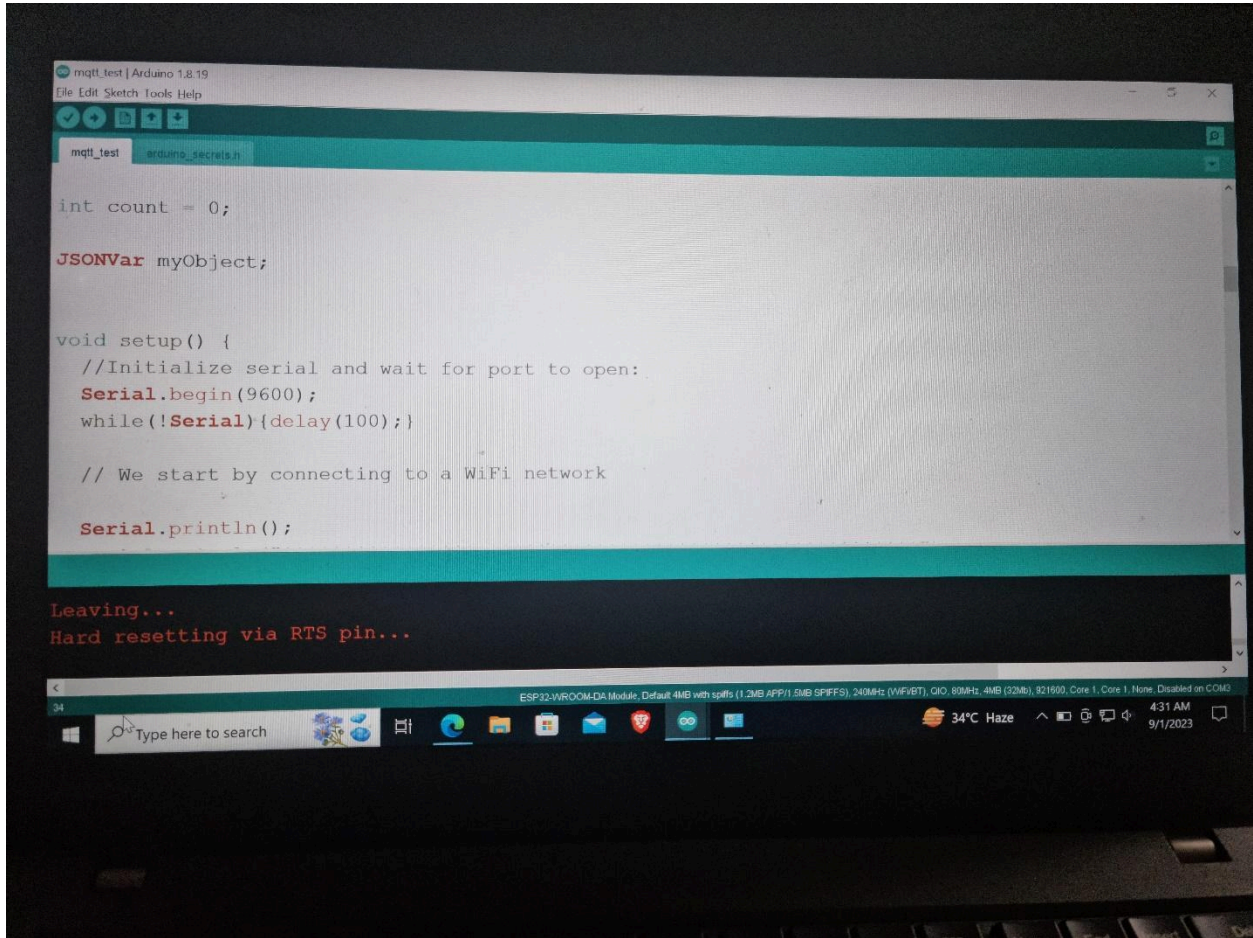


Figure 6: Baud rate selection in the mqtt_test file for Arduino

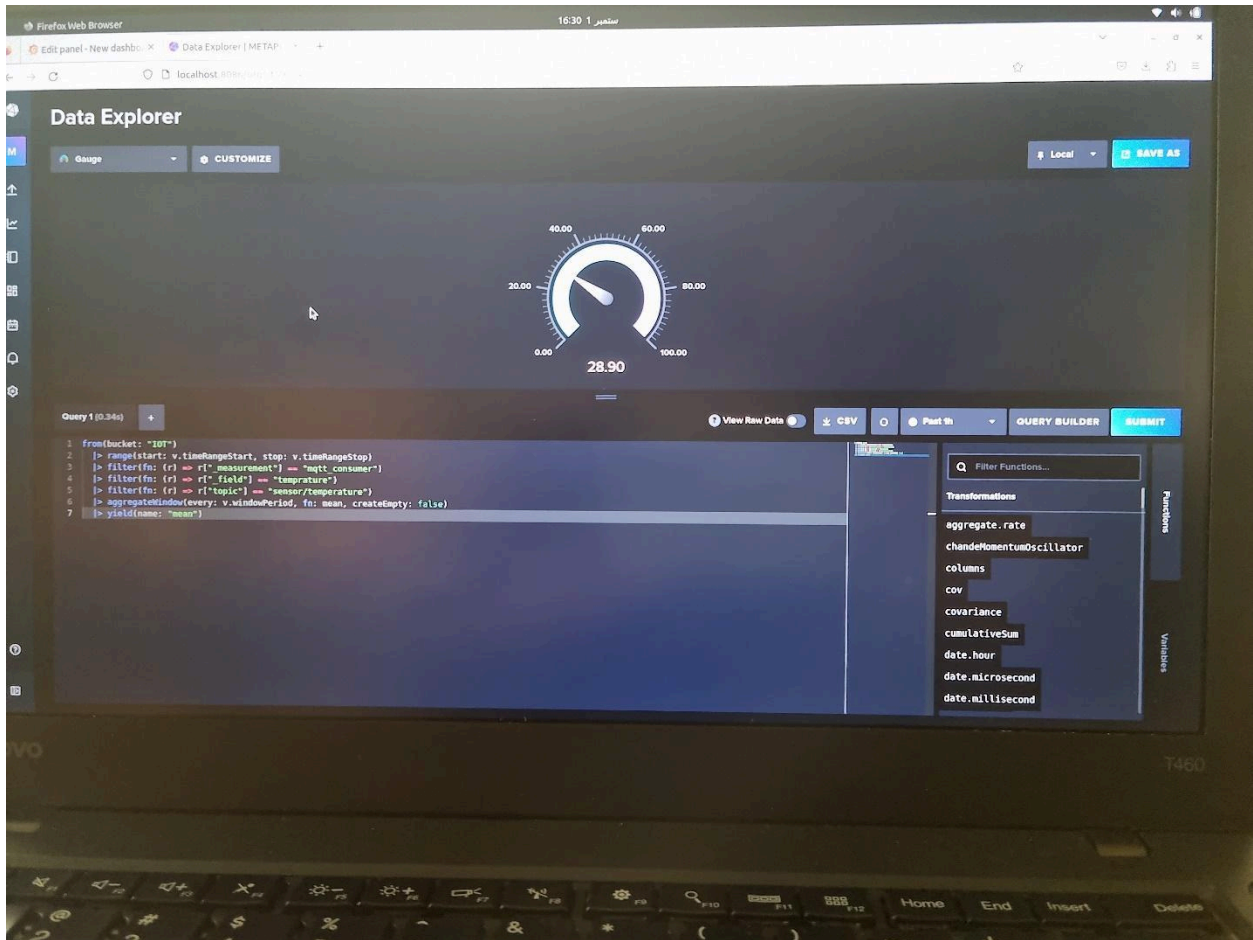


Figure 7: Dashboard view

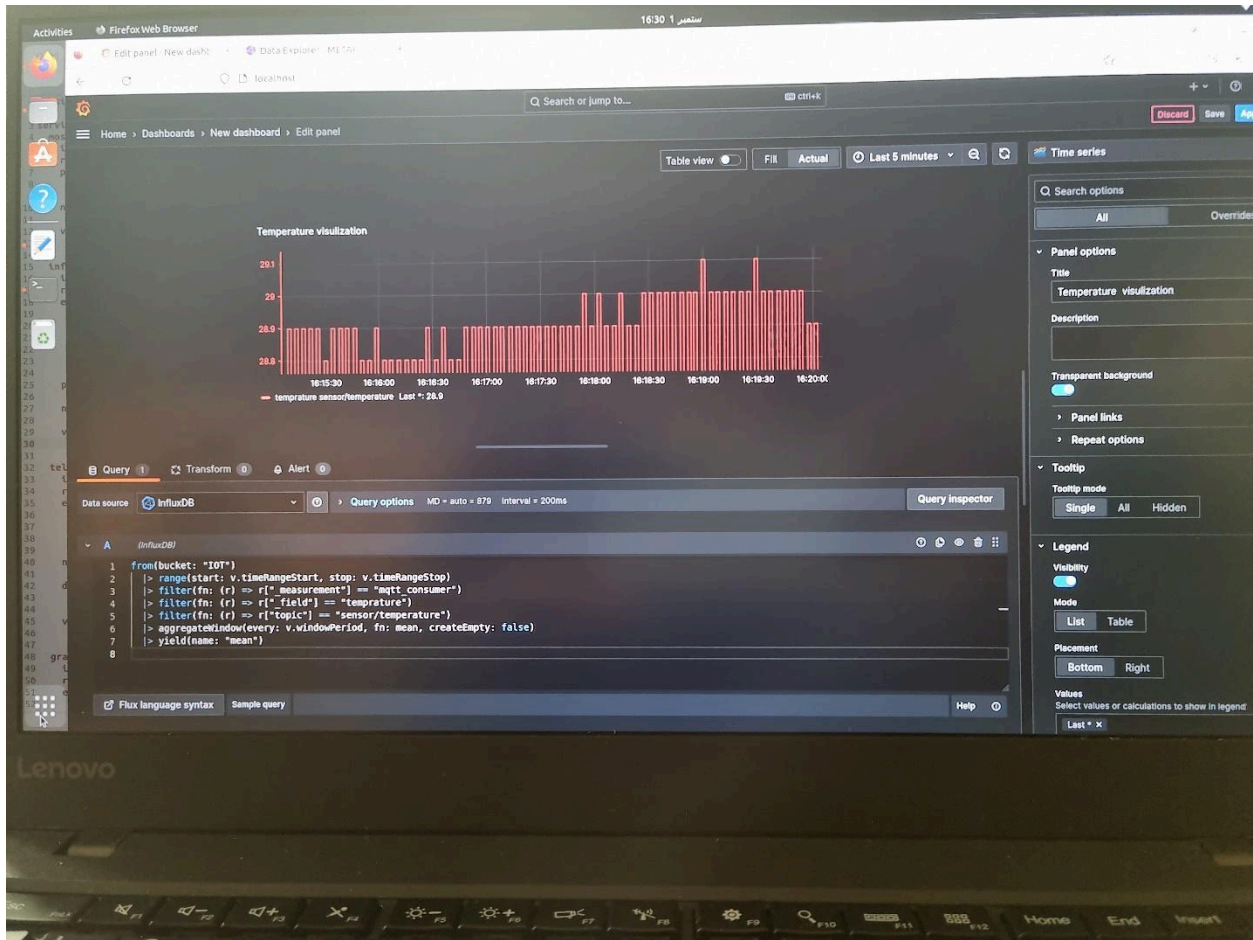


Figure 8: Graph on the dashboard showing the data is being received from containers

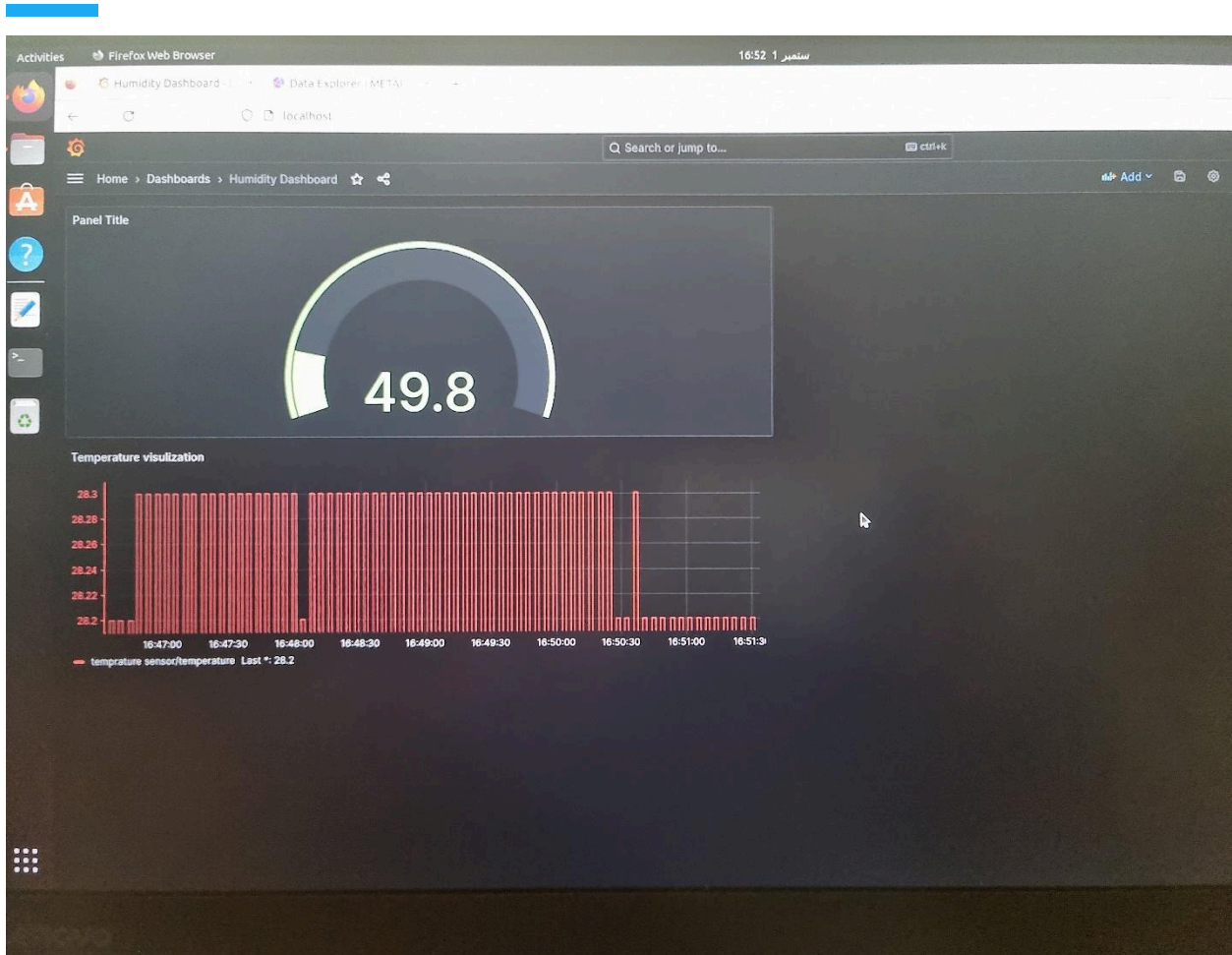


Figure 9: Dashboard View of the data being received in the dash board


```
activities Terminal 17:00 1 سبتمبر
pi@metapt: ~/Downloads/gnbofe/docker_lot
pi@metapt: ~/Downloads/gnbofe/docker_lot
pi@metapt: ~/Downloads/gnbofe/docker_lot$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s31f6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
    link/ether c8:5b:76:90:96:9b brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether f0:d5:bf:7d:79:31 brd ff:ff:ff:ff:ff:ff
    inet 192.168.64.171/24 brd 192.168.64.255 scope global dynamic noprefixroute wlan0
        valid_lft 2450sec preferred_lft 2450sec
    inet6 fe80::f9:fc45:e6a7:8f0a/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:59:b5:36:f0 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:59ff:feb5:36f0/64 scope link
        valid_lft forever preferred_lft forever
964: br-0a3d3f2adb91: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:59:b5:36:f0 brd ff:ff:ff:ff:ff:ff
    inet 172.27.0.1/16 brd 172.27.255.255 scope global br-0a3d3f2adb91
        valid_lft forever preferred_lft forever
    inet6 fe80::42:bdff:fe2a:82c4/64 scope link
        valid_lft forever preferred_lft forever
966: veth0efb091f1995: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-0a3d3f2adb91 state UP group default
    link/ether a2:ef:fbff:febf:febf:febf brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::e0ef:fbff:febf:febf/64 scope link
        valid_lft forever preferred_lft forever
968: veth0d317c91f967: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-0a3d3f2adb91 state UP group default
    link/ether 6a:99:cd:02:b3:2d brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::6a99:cd02:b32d/64 scope link
        valid_lft forever preferred_lft forever
970: vethc107e391f969: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-0a3d3f2adb91 state UP group default
    link/ether 02:42:59:b5:36:f0 brd ff:ff:ff:ff:ff:ff link-netnsid 2
    inet6 fe80::02:42:59:b5:36:f0/64 scope link
        valid_lft forever preferred_lft forever
972: veth019043991f971: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-0a3d3f2adb91 state UP group default
    link/ether 9a:f4:8d:f3:46:c4 brd ff:ff:ff:ff:ff:ff link-netnsid 3
    inet6 fe80::9a:f4:8d:f3:46:c4/64 scope link
        valid_lft forever preferred_lft forever
pi@metapt: ~/Downloads/gnbofe/docker_lot
```

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

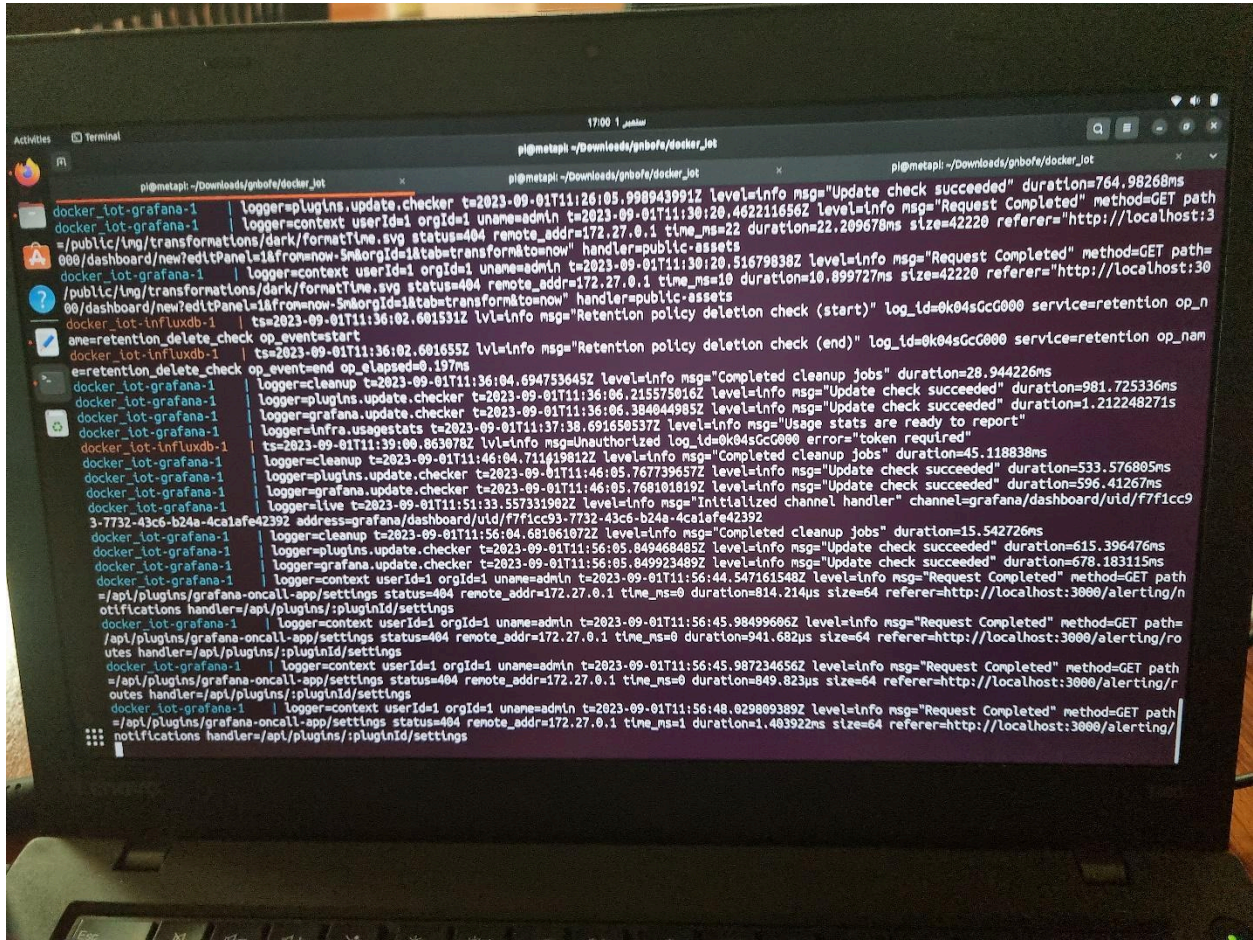


Figure 13: Sensors readings received through the Arduino (Docker IoT)



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>