

IOT Report

Name: Hamza Iqbal

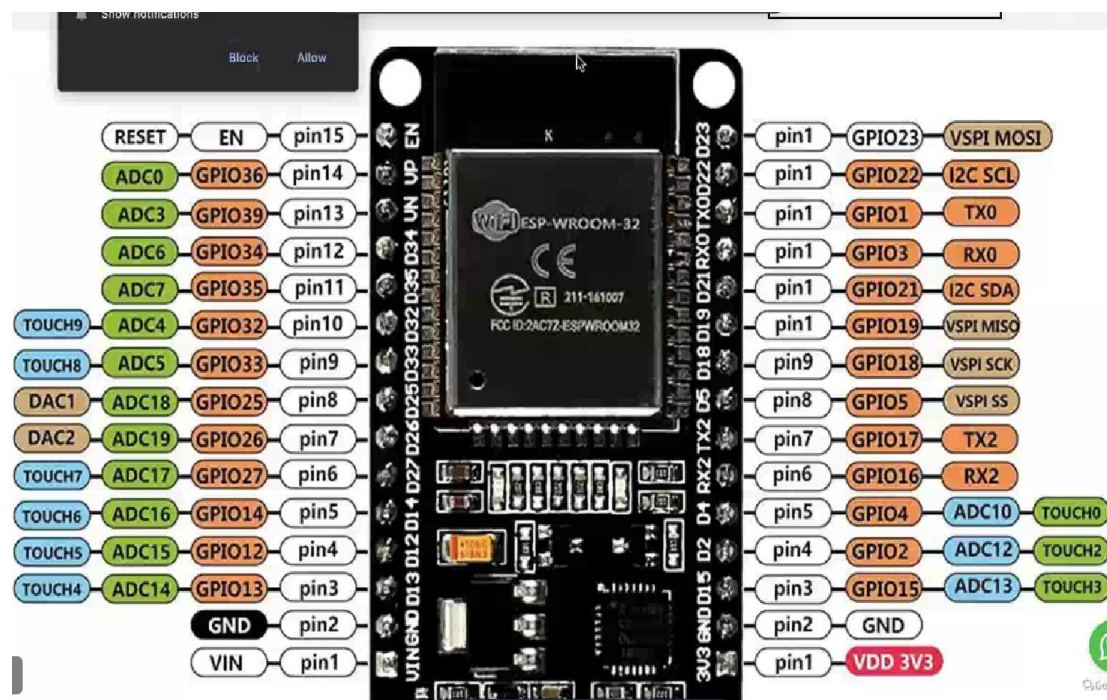
Date: 16-09-23

Objective:

The objective of this project is to establish an IoT system utilizing an ESP32 microcontroller to collect temperature, humidity, and distance data from sensors. This data is then transmitted to a Grafana dashboard via MQTT protocol. The project also involves the use of Docker for efficient container management on a Linux server. The data transmission is facilitated through a local hotspot.

Requirements:

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



Procedure:

➤ **Hardware:**

Connect the temperature and humidity sensor (DHT22) to the ESP32 as per 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.

➤ **Programming:**

Code : For ultrasonic sensor.

```
dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  int d=ultrasonic.getDistance();
  Serial.print("Distance:");
  Serial.print(d);
  Serial.println(" cm");
  delay(100);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  // Compute heat index in Fahrenheit (the default)
```

Code: For temperature and humidity

```
Serial.println("Sensor readings:");

if (isnan(temp_event.temperature)) {
  Serial.println("Error reading temperature!");
} else {
  Serial.print("Temperature: ");
  Serial.print(temp_event.temperature);
  Serial.println(" °C");
}

if (isnan(humidity_event.relative_humidity)) {
  Serial.println("Error reading humidity!");
} else {
  Serial.print("Humidity: ");
  Serial.print(humidity_event.relative_humidity);
  Serial.println(" %");
}
```

➤ **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.

➤ **Setting Up an MQTT-Based IoT Data Pipeline with Docker Containers:**

○ **Install and Configure Mosquitto MQTT Broker:**

- a. Create a Docker container for the Mosquitto MQTT broker.
- b. Configure MQTT broker settings and ports within the container.

○ **Install Grafana and Connect to MQTT Broker:**

- a. Create a Docker container for Grafana.
- b. Configure Grafana to connect to the MQTT broker.
- c. Set up Grafana's data source to communicate with InfluxDB.

○ **Create a Container for InfluxDB:**

- a. Set up a Docker container for InfluxDB to store MQTT data.
- b. Define InfluxDB configurations and expose necessary ports.

○ **Implement MQTT Data Relay Script:**

- a. Create a Python or Node.js script to receive MQTT data from ESP32.
- b. Configure the script to push the data into the InfluxDB container.

○ **Deploy Containers with Docker Compose:**

- a. Run the containers using Docker Compose with the `docker-compose up -d` command.
- b. Ensure all containers start without errors.

○ **Verify Container Status:**

- a. Confirm the status of all containers using `docker-compose ps`.

➤ **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.

➤ Visualizations:

```
pi@metapi:~/Downloads/grafana/docker_tot
docker_tot-grafana-1 | logger=context userId=1 orgId=1 uname=admin t=2023-09-01T11:26:05.998943991Z level=info msg="Update check succeeded" duration=704.96200ms
docker_tot-grafana-1 | logger=context userId=1 orgId=1 uname=admin t=2023-09-01T11:30:20.462216562Z level=info msg="Request Completed" method=GET path=
/public/ing/transformations/dark/formatLine.svg status=404 remote_addr=172.27.0.1 time_ms=22 duration=22.209678ms size=42220 referer="http://localhost:3
000/dashboard/new/editPanel=1&from=now-5m&orgId=1&tab=transformations" handler=public-assets
docker_tot-grafana-1 | logger=context userId=1 orgId=1 uname=admin t=2023-09-01T11:30:20.516798382Z level=info msg="Request Completed" method=GET path=
/public/ing/transformations/dark/formatLine.svg status=404 remote_addr=172.27.0.1 time_ms=10 duration=10.899727ms size=42220 referer="http://localhost:30
00/dashboard/new/editPanel=1&from=now-5m&orgId=1&tab=transformations" handler=public-assets
docker_tot-influxdb-1 | ts=2023-09-01T11:36:02.691531Z lvl=info msg="Retention policy deletion check (start)" log_id=0k04sGcG000 service=retention op_n
ame=retention_delete_check op_event=start
docker_tot-influxdb-1 | ts=2023-09-01T11:36:02.691531Z lvl=info msg="Retention policy deletion check (end)" log_id=0k04sGcG000 service=retention op_n
ame=retention_delete_check op_event=end op_elapsed=0.197ms
docker_tot-grafana-1 | logger=cleanup t=2023-09-01T11:36:04.694753645Z level=info msg="Completed cleanup jobs" duration=28.944226ms
docker_tot-grafana-1 | logger=plugins.update.checker t=2023-09-01T11:36:06.215575016Z level=info msg="Update check succeeded" duration=981.725336ms
docker_tot-grafana-1 | logger=grafana.update.checker t=2023-09-01T11:36:06.384044985Z level=info msg="Update check succeeded" duration=1.212248271s
docker_tot-grafana-1 | logger=infra.usagestats t=2023-09-01T11:37:38.691650537Z level=info msg="Usage stats are ready to report"
docker_tot-grafana-1 | ts=2023-09-01T11:39:00.863078Z lvl=info msg="Unauthorized log log_id=0k04sGcG000 error="token required"
docker_tot-grafana-1 | logger=cleanup t=2023-09-01T11:46:04.711419812Z level=info msg="Completed cleanup jobs" duration=45.118838ms
docker_tot-grafana-1 | logger=plugins.update.checker t=2023-09-01T11:46:05.767739657Z level=info msg="Update check succeeded" duration=533.576805ms
docker_tot-grafana-1 | logger=grafana.update.checker t=2023-09-01T11:46:05.768101819Z level=info msg="Update check succeeded" duration=596.41267ms
docker_tot-grafana-1 | logger=live t=2023-09-01T11:51:33.557331902Z level=info msg="Initialized channel handler" channel=grafana/dashboard/uid/f7f1cc9
3-7732-43c6-b24a-4ca1afe42392 address=grafana/dashboard/uid/f7f1cc93-7732-43c6-b24a-4ca1afe42392
docker_tot-grafana-1 | logger=cleanup t=2023-09-01T11:56:04.681061072Z level=info msg="Completed cleanup jobs" duration=15.542726ms
docker_tot-grafana-1 | logger=plugins.update.checker t=2023-09-01T11:56:05.849468485Z level=info msg="Update check succeeded" duration=615.396476ms
docker_tot-grafana-1 | logger=grafana.update.checker t=2023-09-01T11:56:05.849923489Z level=info msg="Update check succeeded" duration=678.183115ms
docker_tot-grafana-1 | logger=context userId=1 orgId=1 uname=admin t=2023-09-01T11:56:44.547161548Z level=info msg="Request Completed" method=GET path
=/api/plugins/grafana-oncall-app/settings status=404 remote_addr=172.27.0.1 time_ms=0 duration=814.214us size=64 referer="http://localhost:3000/alerting/n
otifications handler=/api/plugins/pluginId/settings
docker_tot-grafana-1 | logger=context userId=1 orgId=1 uname=admin t=2023-09-01T11:56:45.98499606Z level=info msg="Request Completed" method=GET path=
/api/plugins/grafana-oncall-app/settings status=404 remote_addr=172.27.0.1 time_ms=0 duration=941.682us size=64 referer="http://localhost:3000/alerting/r
outes handler=/api/plugins/pluginId/settings
docker_tot-grafana-1 | logger=context userId=1 orgId=1 uname=admin t=2023-09-01T11:56:45.987234656Z level=info msg="Request Completed" method=GET path
=/api/plugins/grafana-oncall-app/settings status=404 remote_addr=172.27.0.1 time_ms=0 duration=849.823us size=64 referer="http://localhost:3000/alerting/r
outes handler=/api/plugins/pluginId/settings
docker_tot-grafana-1 | logger=context userId=1 orgId=1 uname=admin t=2023-09-01T11:56:48.029809389Z level=info msg="Request Completed" method=GET path
=/api/plugins/grafana-oncall-app/settings status=404 remote_addr=172.27.0.1 time_ms=1 duration=1.483922ms size=64 referer="http://localhost:3000/alerting/
notifications handler=/api/plugins/pluginId/settings"
```

➤ Insights:

- Docker provides an efficient and scalable solution for managing containers, ensuring smooth deployment and operation of the IoT system.
- Mosquitto MQTT broker facilitates seamless communication between the ESP32 and the Grafana dashboard.
- InfluxDB serves as a reliable database for storing and retrieving sensor data.

➤ Conclusion:

The project has successfully established an IoT system, integrating sensors with an ESP32 microcontroller. The data collected (temperature, humidity, and distance) is transmitted to a Grafana dashboard via MQTT protocol. Docker containerization enhances the system's scalability and manageability. The utilization of a local hotspot for data transmission ensures remote monitoring and visualization on a Linux server. This comprehensive setup lays a solid foundation for further IoT applications and data-driven.