**Project 1 Part 1: Streaming Live Data to ThingsBoard**

A new folder named Project\_24\_Docker was created in a Terminal window. The docker-compose.yml file was downloaded and placed inside the Project\_24\_Docker folder. Within the Project\_24\_Docker folder, a folder named mosquitto was created. Three more subfolders titled config, data, and log were created inside the mosquitto folder. The mosquitto.config file was downloaded and placed inside the config folder.

A screenshot of a computer screen

Description automatically generated

The Project\_24\_Docker folder was navigated to in a Terminal window, and the Mosquitto container was initialized using the command **docker-compose up**.

A screenshot of a computer

Description automatically generated

The Paho MQTT Python client library was installed locally using the command **pip install paho-mqtt** in a local Terminal window.

A screenshot of a computer program

Description automatically generated

The home folder was navigated to in a Terminal window. Inside the home folder, two folders named .mytb-data and .mytb-logs were created.

A screenshot of a computer

Description automatically generated

Next to the location where the Project\_24\_Docker folder was created, a new folder titled Project\_24\_MQTT was created. Inside this folder, a new subfolder named ThingsBoard was created. The docker-compose.yml file was downloaded and placed inside the ThingsBoard folder. Additionally, two folders named .mytb-data and .mytb-logs were created inside the ThingsBoard folder.

A screenshot of a computer program

Description automatically generated

The ThingsBoard folder, created in Step 5, was navigated to in a Terminal window, and the ThingsBoard container was initialized using the command **docker-compose up**.

A screenshot of a computer

Description automatically generated

A new subfolder named paho-mqtt was created inside the Project\_24\_MQTT folder. The TBPublish.py file was downloaded and placed inside the paho-mqtt folder. This file was then opened in VS Code. The sensor\_data dictionary was modified by adding a new key, humidity, with a value set to 0. Within the while loop, a statement was added to generate random integer values between 50 and 100 for the humidity variable.

A screenshot of a computer program

Description automatically generated

In VS Code, a Terminal window was opened, and the TBPublish.py file was run.

A screenshot of a computer

Description automatically generated

<http://localhost:8080/> was navigated to in a browser window. ThingsBoard was accessed using the given credentials.

A screenshot of a computer

Description automatically generated

In ThingsBoard, the “Devices” option from the left menu was selected, revealing the DHT11 Demo Device. This device was opened, and the Latest Telemetry tab was navigated to.

A screenshot of a computer

Description automatically generated

The main page of Firebase was navigated to, and a new project named module24Project was added following the steps in Video 24.3.

A screenshot of a computer

Description automatically generated

A Realtime database in Firebase was created following the steps in Video 24.3. A field titled temperature was added to the database and initialized to zero.

A screenshot of a computer

Description automatically generated

In ThingsBoard, the Root Rule Chain was navigated to. A "rest API call" node named Firebase was added. The link to the Realtime database from Firebase followed by "/temperature.json" was pasted into the “Endpoint URL pattern” field. The “Firebase” node was connected to the “Message Type Switch” node, and “Post telemetry” was added as the link label.

A screenshot of a computer screen

Description automatically generated

Firebase was navigated to, and the temperature and humidity data were displayed by expanding the entries in the Realtime database.

A screenshot of a computer

Description automatically generated