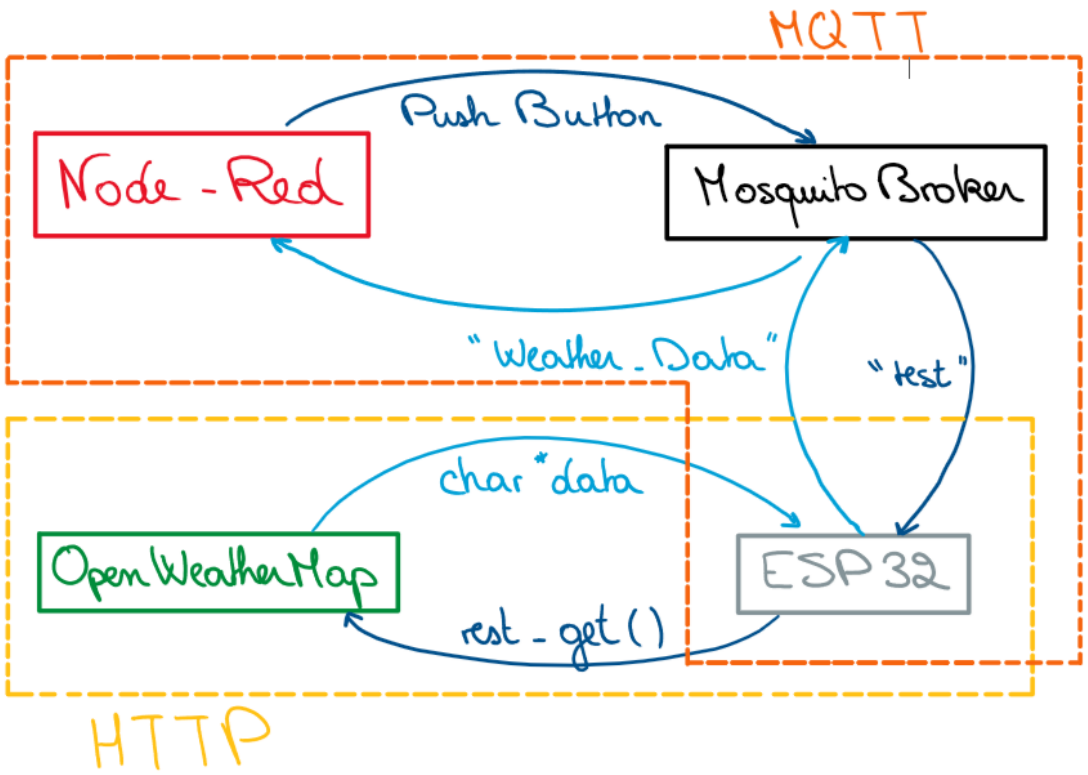
General Diagram



ESP32 HTTP Client

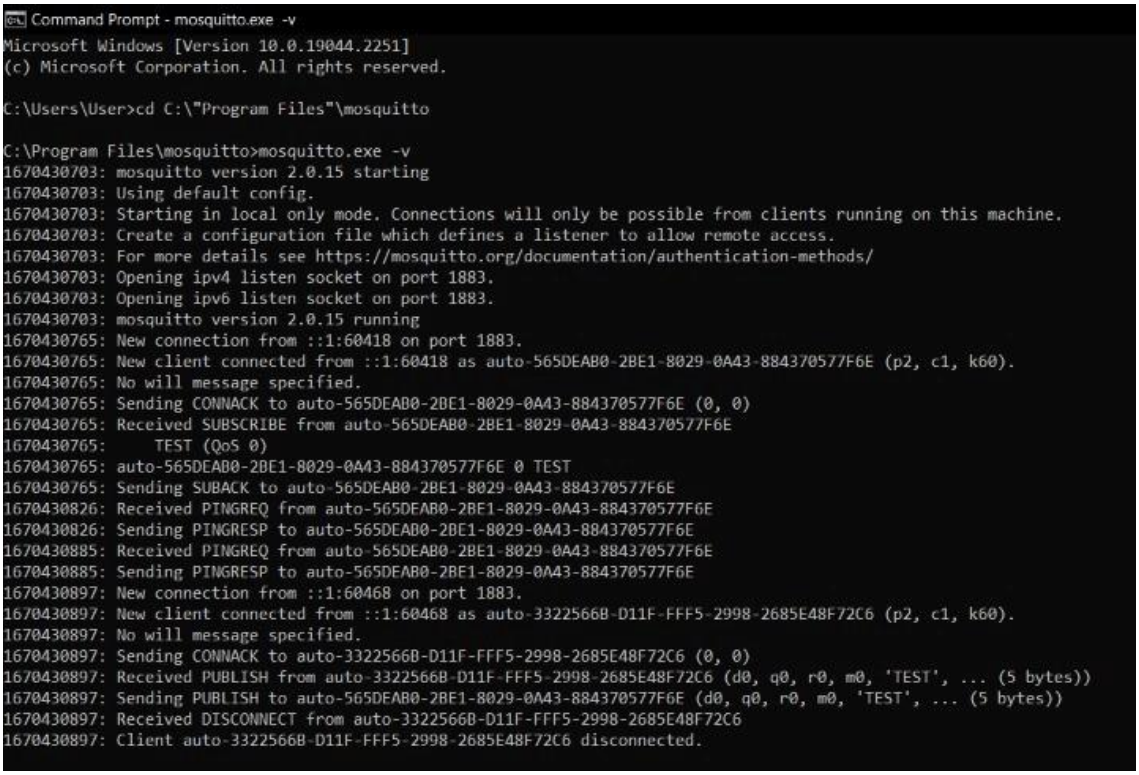
We first started with the HTTP part, the goal was to successfully retrieve the data from the OpenWeatherMap and print them on the terminal of VSCode in the format requested in the instructions.

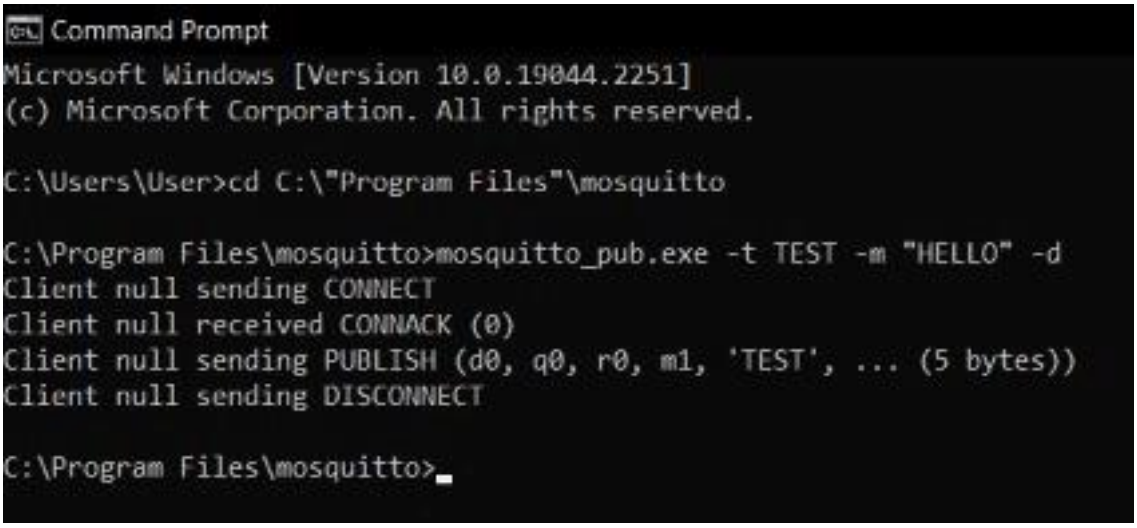
- The client_event_get_handler() function retrieves the data and stores it inside a variable called data. Seeing that the received data is going to be too large, it will be sent to the esp in two parts, so we used the strncat() function to concatenate those two parts together
- The rest_get() function is used to do the API call using the API KEY we got after signing up to the weather app
- We added the function print_data() which would be responsible for selecting the data we need from the "data" variable and displaying it in the right format.
    - We first used the cJSON_Parse() on the "data" variable to turn it into a cJSON recognizable variable called "json" in order to work on it with the functions specific to cJSON
    - We used the cJSON_GetObjectItem() and to cJSON_GetArrayItem() extract the information needed and printed it using the printf() function

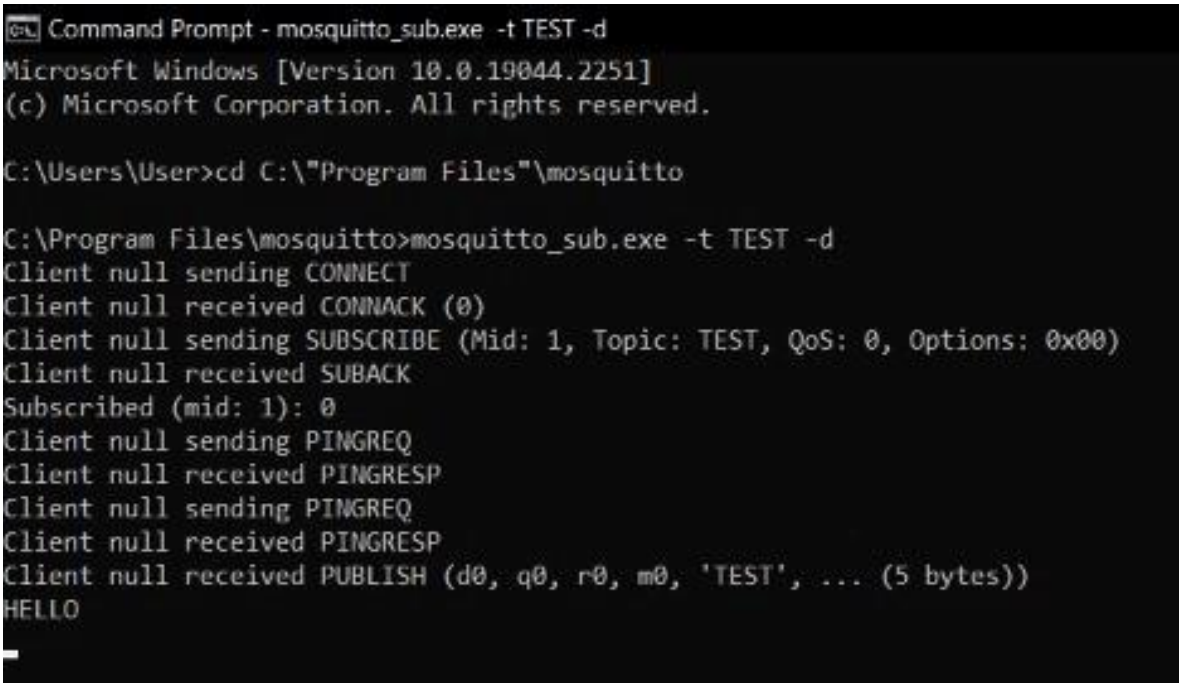
MQTT Mosquitto Broker

Installing and testing the MQTT Mosquitto broker was straight forward.

We used three terminals, the first one runs mosquitto in the background, the second one is the publisher whom sends a message through a specified topic, and the third one is the subscriber who subscribes to that topic and listens to the messages sent on that topic.





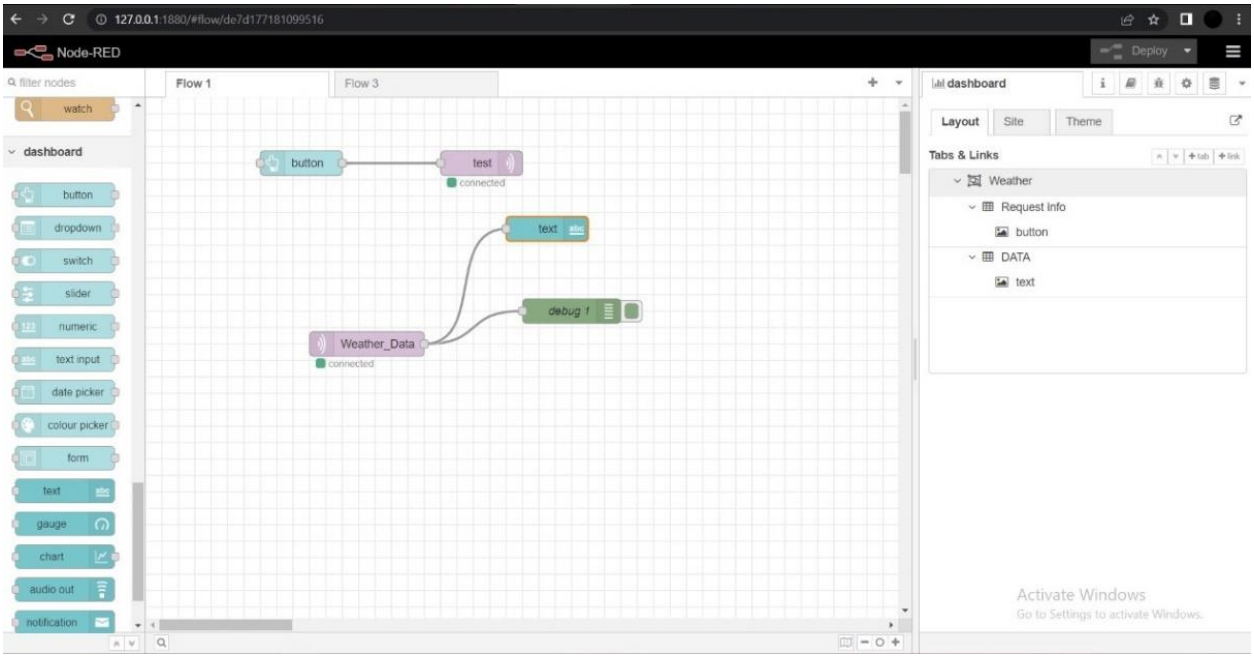


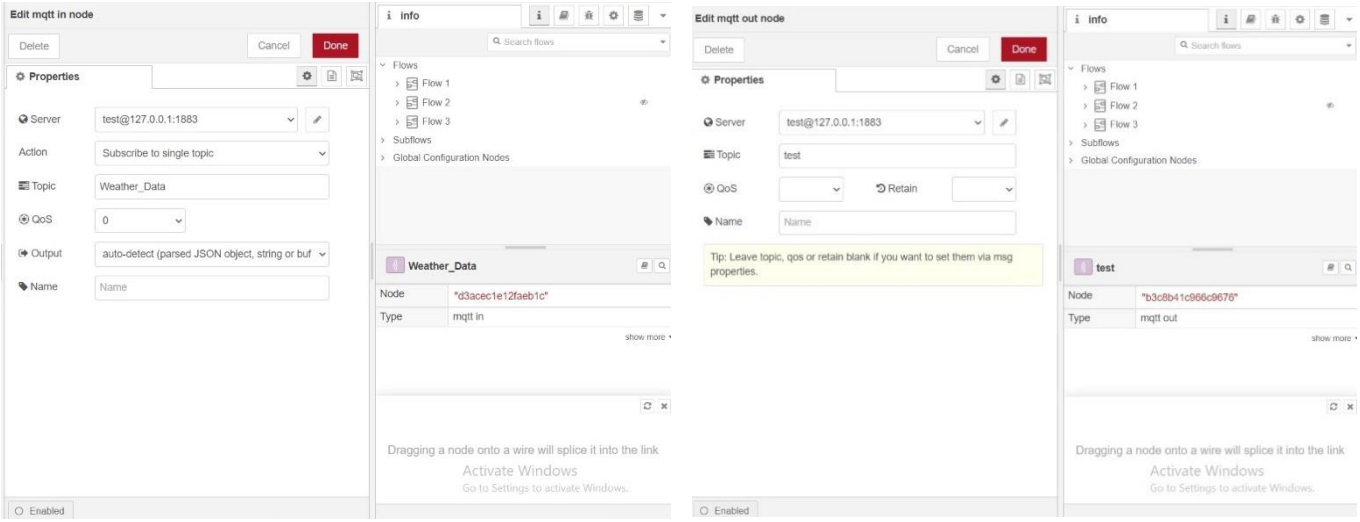
All of the above is done locally on the PC with no connection to the esp or Node-RED whatsoever.

For the purpose of our project we needed to allow the broker to communicate with the esp32 which was causing problems because the broker would only allow communication with the local host. To fix that problem we had to go to ProgramFiles\mosquitto and add a test.conf file which contained the two lines “listener 1883” and “allow_anonymous true”.

Node-RED MQTT Client and Dashboard

To set up Node-RED we connected a push button to an MQTT publish node named “test” below. It was assigned the topic “test”, the request for the data will be sent through it. On the other hand, there is the MQTT receive node which will be the one receiving the data through the topic “Weather_Data” and will display it using the “text” node on the page. We can also see below that both the servers for “test” and “Weather_Data” are the local host because they are communicating with the broker which is on the PC itself.





ESP32 MQTT Client

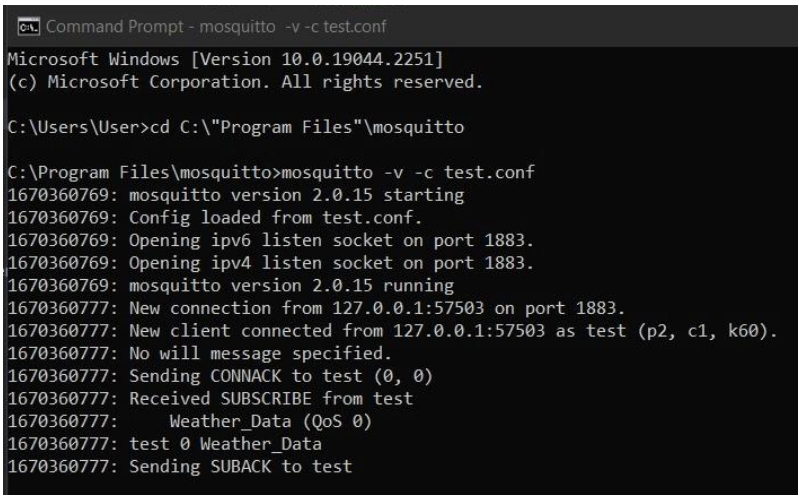
In the function mqtt_event_handler_cb(), in the case of MQTT_EVENT_CONNECTED, we put as a condition to check that the topic received is called “test” (the one we want to subscribe to) and in the case MQTT_EVENT_DATA, after making sure we are receiving a call through the topic we subscribed to, we will call the function rest_get() defined earlier to get the data from OpenWeatherMap and send it through the topic “Weather_Data” to the broker which will then send it back to Node_RED.

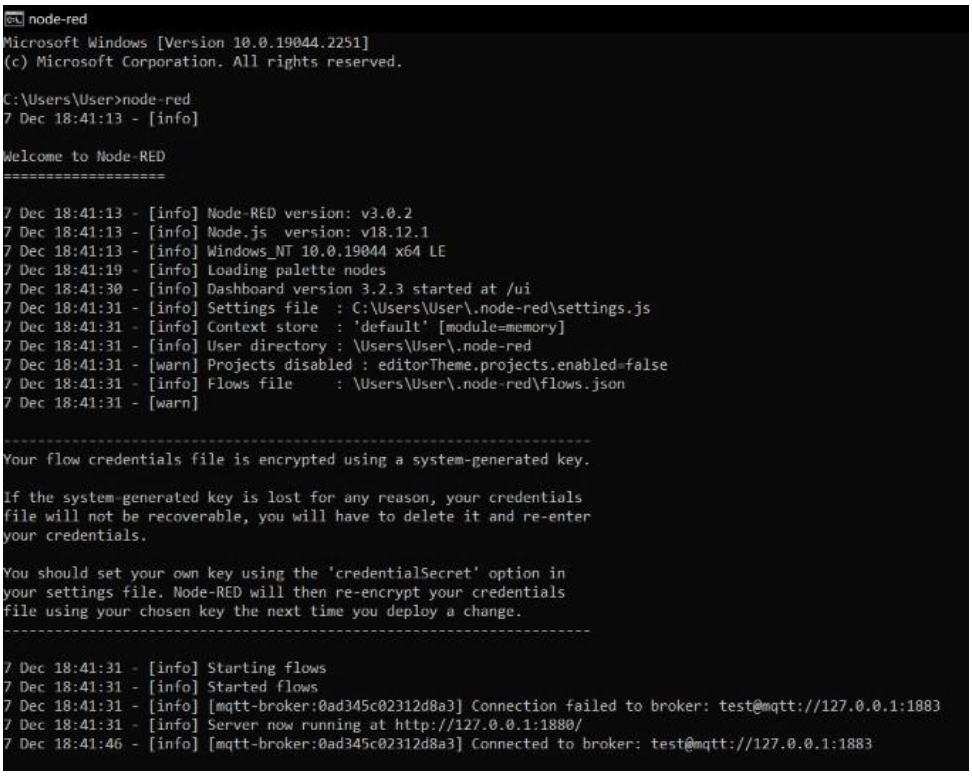
One more thing to note is the uri in the function mqtt_app_start() which needs to be modified every once in a while, seeing that the IP keeps changing “mqtt://MY_IP:1883”.

We merged this code with the one we wrote for the HTTP part, but we modified the function print_data() in the latter so that it concatenates all the data we want to send in order, adding icons and in the right format, puts them in a buffer variable and sends them in a variable called “send_data”.

Final Result

To run the program we just worked on, we need to have Node-RED and the Mosquitto broker running in the background through the command prompt as demonstrated below





We also need the esp32 to be connected, with the code uploaded on it.

Finally, all that’s left is to open the Node-RED page and press the button shown in the picture below, and a few seconds later the requested data will be displayed.