

Azure Research

By: Johnson Domacasse (#4471709)

17 April 2024

Contents

1. Introduction:	3
2. Methods:.....	3
2.1 setting up our environment:.....	3
2.2 Running the code:	3
2.3 Monitoring on Azure:.....	4
3. Discussion:	5
3.1 Node-RED:.....	5
4. Bibliography:	6

1. Introduction:

The previous research was based on the cloud platform: Amazon Web Services (AWS). This research we will explore a different cloud platform known as Azure. Its one of the if not the most used cloud platforms in the industry. This research will be a guide as to which steps were taken to receive data on our cloud dashboard.

2. Methods:

What was used for this research were the following:

- Azure Stream Analytics Job
- Azure IoT hub
- ESP32
- Arduino IDE
- Azure SDK for C

For this research we will use the IoT hub for setting up our environment and then use the stream analytics tab to monitor the received data on our dashboard. From there we can export this data into separate files.

2.1 setting up our environment:

We begin by logging into azure. We use our FontysICT account for this contrary to the previous normal student account. From there we go to our the "dashboard" tab. Here we see both the analytics stream and the IoT hub already created for us. Once we are in the IoT hub, we go into the devices tab and here we add a device. Nothing special needs to be added here so we simply create the device making sure that everything is autogenerated for us. What is important here is that we have access to our connection strings and keys.

From there we go to our Arduino Ide and here we have to download the "Azure SDK for C" library. From there, we can traverse to this library, then into examples, then into the Azure IoT hub for esp32 folder. In this folder, we have to traverse into the IoT configs file to configure some details. The MQTT details you can fill out based on your network. From there under azure IoT we change some of the variables. For the FQDN variable you use the following format: `iotHub-{I-PCN}.azure.devices.net`. the device id is the name you have given your device. Finally the device key is the primary key that can be found in your device.

2.2 Running the code:

From there we compile the example and upload this onto our ESP32. If all went well, your monitor should look like figure 1.

```




1970/1/1 00:00:00 [INFO] Setting time using SNTP
.
2024/4/17 14:58:59 [INFO] Time initialized!
2024/4/17 14:58:59 [INFO] Client ID: johnson_device
2024/4/17 14:58:59 [INFO] Username: iotHub-i483564.azure-devices.net/johnson_device/?ap
2024/4/17 2024/4/17 14:58:5914 [INFO] :MQTT event MQTT_EVENT_BEFORE_CONNECT
58:59 [INFO] MQTT client started
2024/4/17 14:58:59 [INFO] Sending telemetry ...
2024/4/17 14:59:01 [INFO] MQTT event MQTT_EVENT_CONNECTED
2024/4/17 14:59:01 [INFO] Subscribed for cloud-to-device messages; message id:21315
2024/4/17 14:59:01 [INFO] Message published successfully
2024/4/17 14:59:01 [INFO] MQTT event MQTT_EVENT_SUBSCRIBED

```

Figure 1: Code uploaded successfully.

2.3 Monitoring on Azure:

Now that our code is uploaded correct, we can go to our dashboard to monitor if the data is being received. When we go back to the dashboard tab in azure, we go into the stream analytics tab rather than the IoT hub this time. Here we go into the “query” tab. This is where we will run the query in figure 2, to get all of the data received in our IoT hub.

 Test query
  Save query
  Discard changes

```

1  SELECT
2  |    *
3  INTO
4  |    OUTPUT
5  FROM
6  |    INPUT

```

Figure 2: Data acquisition query.

We run the query and if all is configured well, the data shall be input into the OUPUTs. See figure 3.

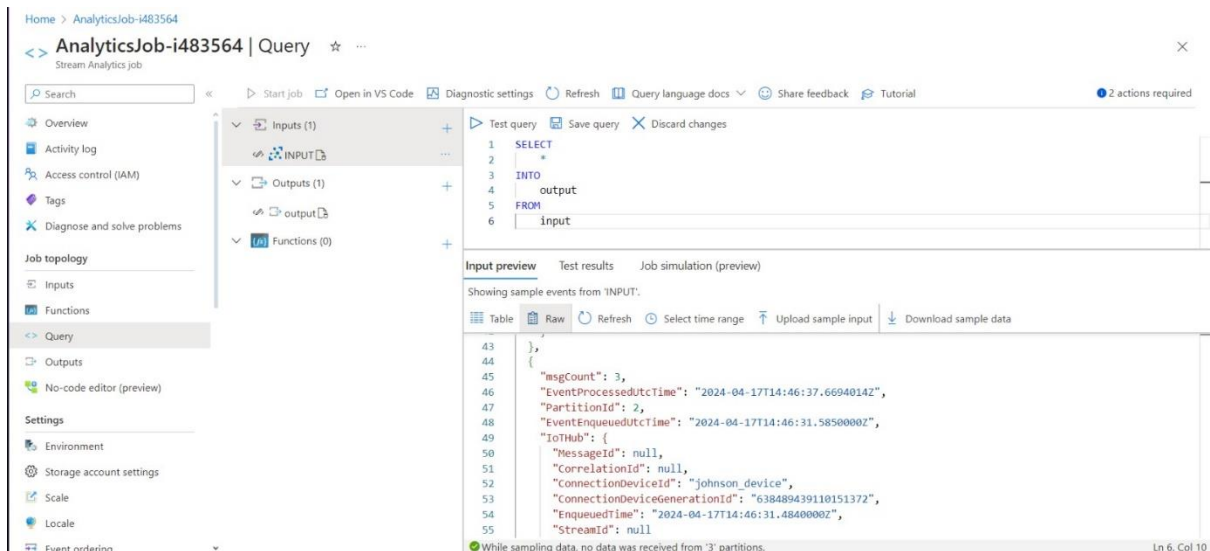


Figure 3: data being received successfully.

3. Discussion:

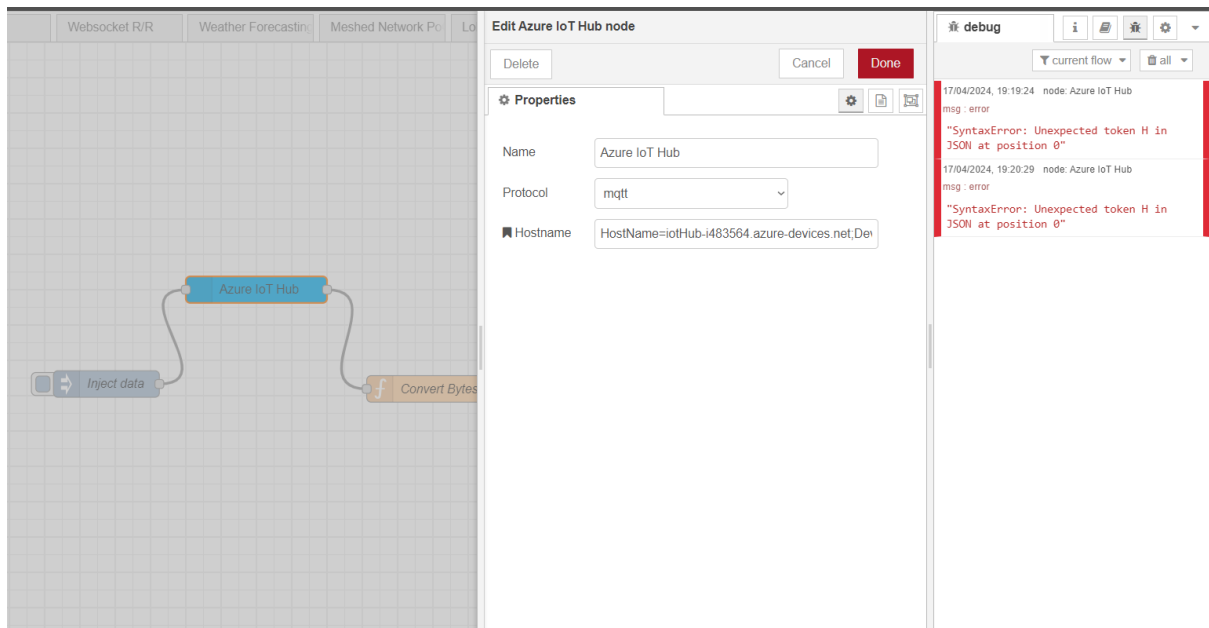
This section will dictate anything else I have tried doing within this research that didn't seem to function correctly.

3.1 Node-RED:

I attempted to connect this implementation with node-red in two ways. One using the MQTT node to connect it as I connected it with AWS. This didn't work because due to the service being a cloud service, configuring the MQTT would require me to enable TLS. In order to do so, I needed the a username and password. This didn't work. I tried using the connection string provided to me and the same URL used in the config file. As for username, I have tried using the username I use in azure. I used a source that had me use the CLI window within the azure portal, however when attempted, it said I don't have access to do this so I assumed I needed a higher authority and/or it would result in extra costs.[1]

The second method I tried was using the built in Azure IoT hub node. This didn't work I think due to being outdated. However this was the closest I have gotten to getting it to work. Although it was connecting to it, this was the error messages I was getting. See figure 3. In this figure I am using a flow from a tutorial online. The same flow is used by my teacher Renata. For her, when injecting, the flow runs fine, for me it gives me the error messages seen in figure 3. Despite this being the exact same flows, with the only difference being the connection strings. [2][3]

From here we agreed that I would write what I have found so far for this challenge and work it in this document.



4. Bibliography:

- [1] - Kinkar, N. (2021, February 13). *Connecting node-red to azure IOT hub using MQTT nodes*. Medium. <https://medium.com/@nikhilkinkar/connecting-node-red-to-azure-iot-hub-using-mqtt-nodes-6e9160549348>
- [2] - *Red-contrib-azure-IOT-hub*. node. (2017, October). <https://flows.nodered.org/node/node-red-contrib-azure-iot-hub>
- [3] - *Red-Contrib-azureiothubnode-Param*. node. (n.d.). <https://flows.nodered.org/node/node-red-contrib-azureiothubnode-param>