Simple AWS IoT Connection for Temperature & Humidity Sensor

ECEA5348: M2M & IoT Interface Design & Protocols for Embedded Systems

Glenn Frey Olamit

December 15, 2021

1. Implementation & Assumption Notes

I used Python and Psudosensor script to implement the data server and Paho MQTT library to send data to AWS IoT Thing. I used the Raspberry Pi as the host of a data server to simulate a real project.

I used Raspberry Pi model 3b+ for this project and installed the latest Raspbian OS for this embedded project as it fit the requirement. I installed Geany, a lightweight GUI text editor as resources must be conserved and MobaXterm to work remotely.

I used the AWS console in AWS IoT to generate the required certificate to connect the data server to AWS IoT Core namely the publicKey.pem, privateKey.pem, certificate.pem, AmazonRootCA1.pem.

I assume the values extracted from the psudosensor in Celsius and the Values I choose to display are in Celsius.

I assume the temperature and humidity reading and timestamp are sent as one JSON in the data server.

I assume the data received by the AWS IoT Thing is a JSON with a temperature and humidity reading and timestamp are in it. The data is then passed to the AWS SQS by adding a rule with a topic “RaspberryPI” as a filter for the JSON data.

I use the Boto3 SQS library to write a simple program in Python to act as a client that removes messages from the SQS queue and display them on the command line.

I assume the client program will show the connection is establish before displaying the SQS message.

2. Code

2.1 data server.py

Breakline

```
#!/usr/bin/python

# this source is part of my Hackster.io project:
https://www.hackster.io/mariocannistra/radio-astronomy-with-rtl-sdr-raspberrypi-and-amazon-aws-iot-4
5b617

# use this program to test the AWS IoT certificates received by the author
# to participate to the spectrogram sharing initiative on AWS cloud

# this program will publish test mqtt messages using the AWS IoT hub
# to test this program you have to run first its companion awsiotsub.py
# that will subscribe and show all the messages sent by this program

import paho.mqtt.client as paho
import os
import socket
import ssl
from time import sleep
#from random import uniform
from psuedoSensor import PseudoSensor
import datetime

connflag = False

def on_connect(client, userdata, flags, rc):
    global connflag
    connflag = True
    print("Connection returned result: " + str(rc) )
    print("Connection successful")

def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

#def on_log(client, userdata, level, buf):
```

```
#    print(msg.topic+" "+str(msg.payload))

mqttc = paho.Client()
mqttc.on_connect = on_connect
mqttc.on_message = on_message
#mqttc.on_log = on_log

awshost = "a1u1xwv7g7dxk1-ats.iot.us-east-1.amazonaws.com"
awsport = 8883
clientId = "RaspberryPI"
thingName = "RaspberryPI"
caPath = "/home/glennfrey/aws/AmazonRootCA1.pem"
certPath = "/home/glennfrey/aws/certificate.pem"
keyPath = "/home/glennfrey/aws/privateKey.pem"

mqttc.tls_set(caPath, certfile=certPath, keyfile=keyPath, cert_reqs=ssl.CERT_REQUIRED,
tls_version=ssl.PROTOCOL_TLSv1_2, ciphers=None)

mqttc.connect(awshost, awsport, keepalive=60)

mqttc.loop_start()

while 1==1:
    sleep(0.5)
    if connflag == True:
#        tempreading = uniform(20.0,25.0)
        ps = PseudoSensor()
        humidity,temperature = ps.generate_values()
#        mqttc.publish("temperature", temperature, "humidity", humidity, "time", datetime.datetime.now(),
qos=1)
        mqttc.publish("RaspberryPI", f"temperature : {temperature}" + f", humidity : {humidity}" + f",
time : {datetime.datetime.now()}", qos=1)
        print("msg sent: temperature " + "%.2f" % temperature + ", humidity " + "%.2f" % humidity + ",
time " + "%s" % datetime.datetime.now() )
    else:
        print("waiting for connection...")
```

2.2 psudoSensor.py

Breaklines

```
import random

class PseudoSensor:

    h_range = [0, 20, 20, 40, 40, 60, 60, 80, 80, 90, 70, 70, 50, 50, 30, 30, 10, 10]

    t_range = [-20, -10, 0, 10, 30, 50, 70, 80, 90, 80, 60, 40, 20, 10, 0, -10]

    h_range_index = 0

    t_range_index = 0

    humVal = 0

    tempVal = 0


    def __init__(self):

        self.humVal = self.h_range[self.h_range_index]

        self.tempVal = self.t_range[self.t_range_index]


    def generate_values(self):

        self.humVal = self.h_range[self.h_range_index] + random.uniform(0, 10);

        self.tempVal = self.t_range[self.t_range_index] + random.uniform(0, 10);

        self.h_range_index += 1

        if self.h_range_index > len(self.h_range) - 1:

            self.h_range_index = 0
```

```
        self.t_range_index += 1

        if self.t_range_index > len(self.t_range) - 1:

            self.t_range_index = 0

        return self.humVal, self.tempVal
```

2.3 client.py

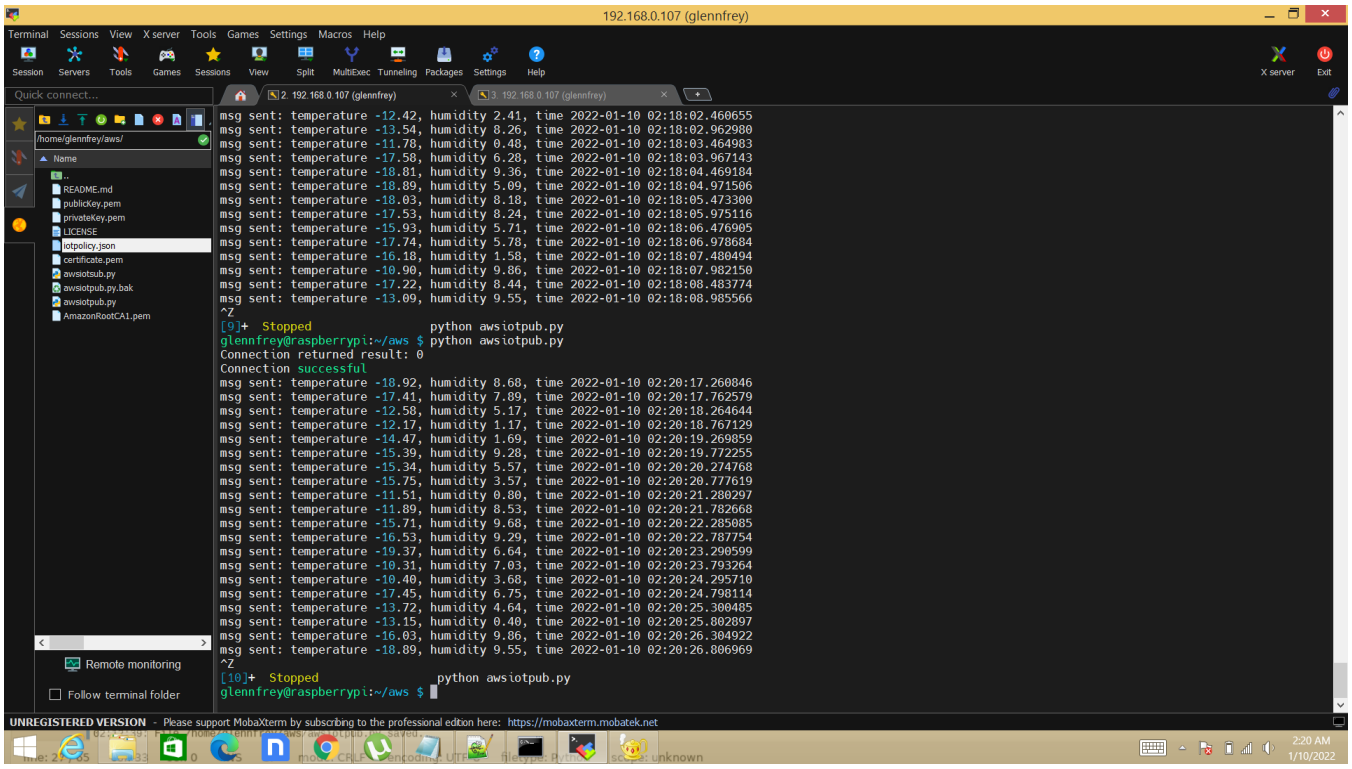
Breaklines

```
import boto3


sqs = boto3.resource('sqs', region_name='us-east-1',
aws_access_key_id="AKIAR2THXZDUAAV2IN7E",
aws_secret_access_key="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")

# Get the queue
queue = sqs.get_queue_by_name(QueueName='RaspberryPIQueue')

# Process messages by printing out body and optional author name
for message in queue.receive_messages():
    # Get the custom author message attribute if it was set
    print("Message queue receive...")
    print(message.body)
    print("SQS queue remove successful!")
    message.delete()
```

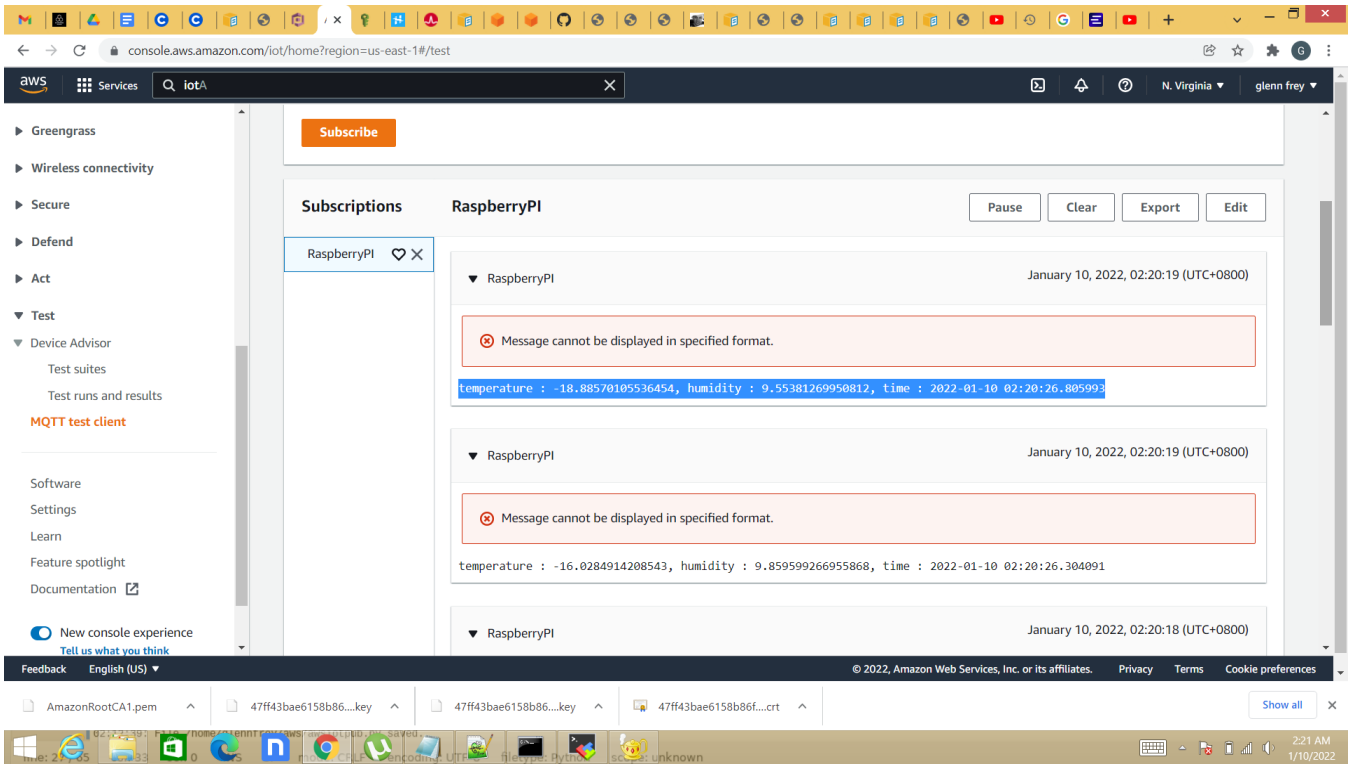
---

3. Screen Capture

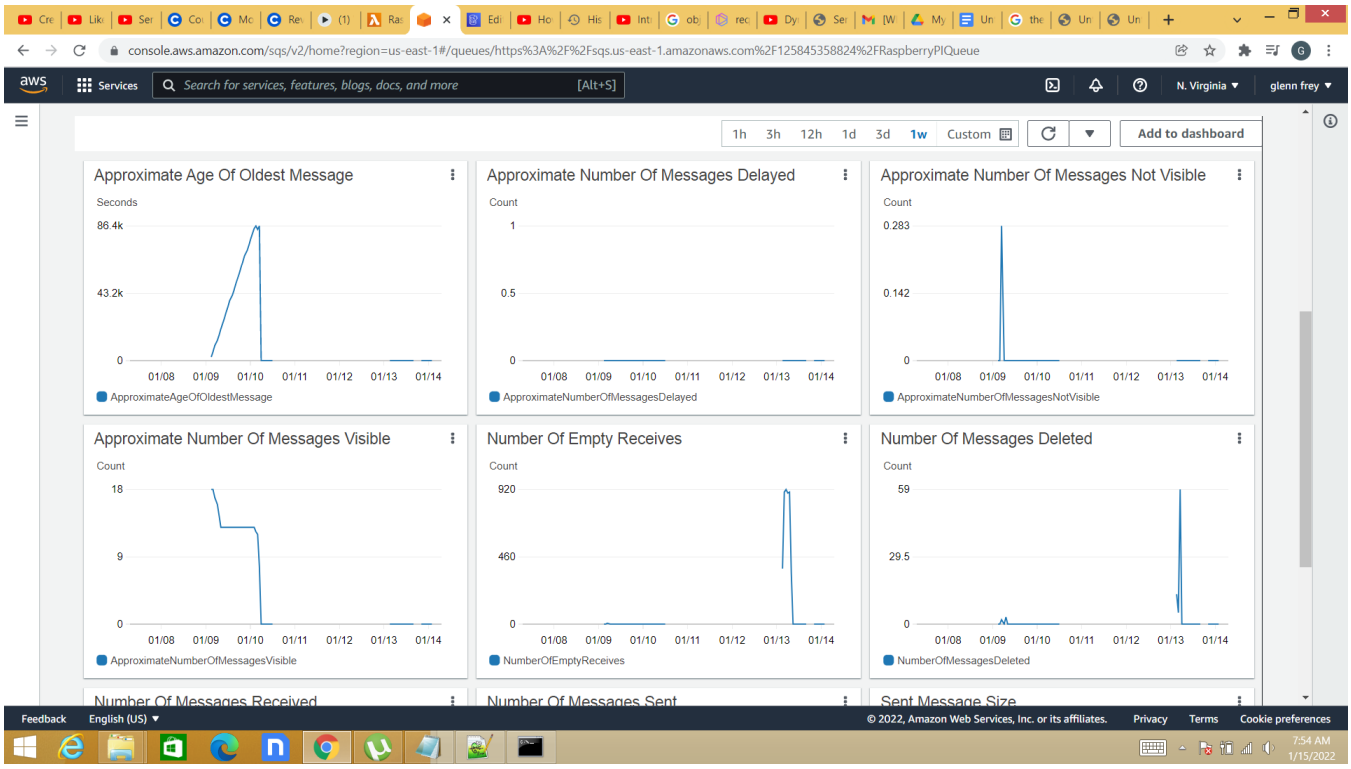
3.1 The data server tracing it's connection to AWS



3.2 The data server sending messages (JSON timestamps, temperature, and humidity values) to AWS



3.3 The client connecting to AWS SQS



3.4 The client showing the timestamps of incoming SQS messages