

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-45b617
```

```
# 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

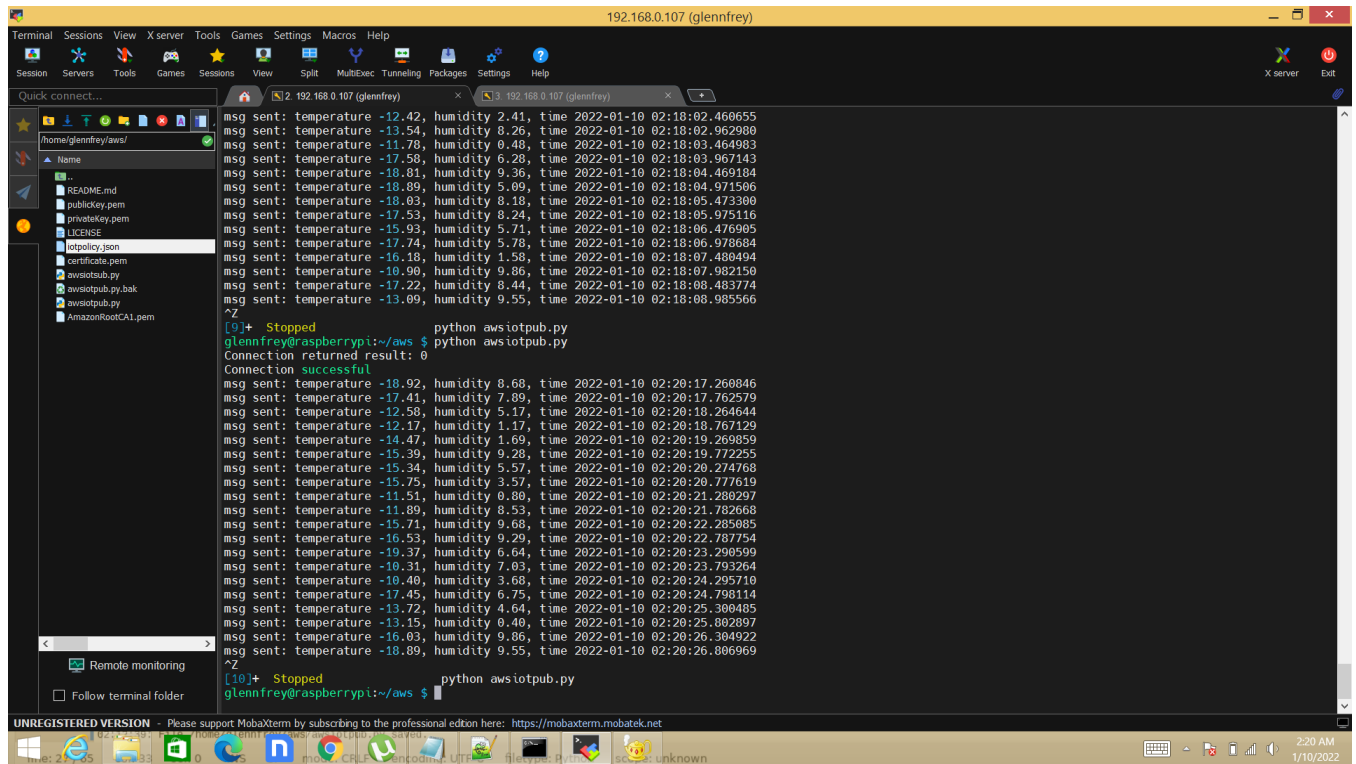
sqs = boto3.resource('sqs', region_name='us-east-1',
aws_access_key_id="AKIAR2THXZDUAAV2IN7E",
aws_secret_access_key="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")

# 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



```
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help

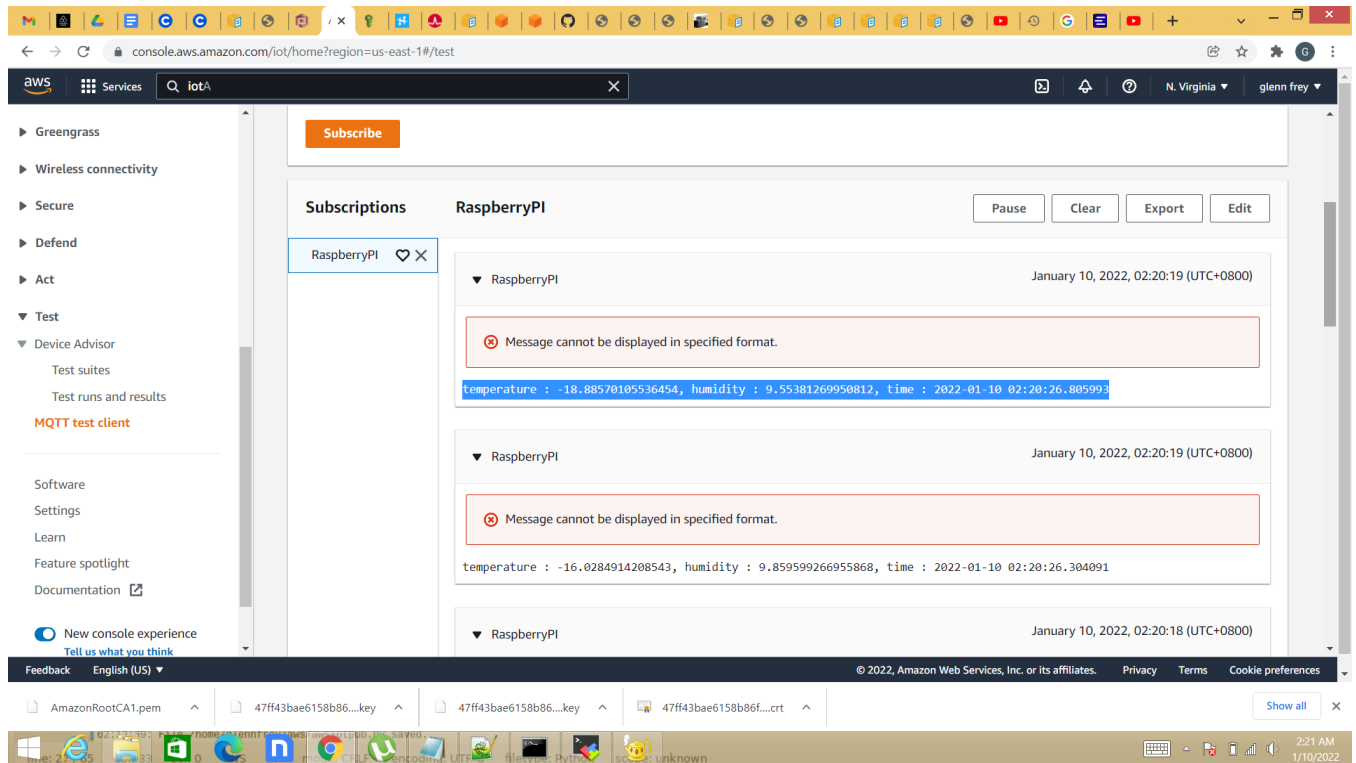
Quick connect...
[home/glennfrey/aws/]
Name
..
README.md
publickey.pem
privatekey.pem
LICENSE
iotpolicy.json
certificate.pem
awsiotpub.py
awsiotpub.py.bak
awsiotpub.py
AmazonRootCA1.pem

msg sent: temperature -12.42, humidity 2.41, time 2022-01-10 02:18:02.460655
msg sent: temperature -13.54, humidity 8.26, time 2022-01-10 02:18:02.962980
msg sent: temperature -11.78, humidity 0.48, time 2022-01-10 02:18:03.464983
msg sent: temperature -17.58, humidity 6.28, time 2022-01-10 02:18:03.967143
msg sent: temperature -18.81, humidity 9.36, time 2022-01-10 02:18:04.469184
msg sent: temperature -18.89, humidity 5.09, time 2022-01-10 02:18:04.971506
msg sent: temperature -18.03, humidity 8.18, time 2022-01-10 02:18:05.473300
msg sent: temperature -17.53, humidity 8.24, time 2022-01-10 02:18:05.975116
msg sent: temperature -15.93, humidity 5.71, time 2022-01-10 02:18:06.476905
msg sent: temperature -17.74, humidity 5.78, time 2022-01-10 02:18:06.978684
msg sent: temperature -16.18, humidity 1.58, time 2022-01-10 02:18:07.480494
msg sent: temperature -10.90, humidity 9.86, time 2022-01-10 02:18:07.982150
msg sent: temperature -17.22, humidity 8.44, time 2022-01-10 02:18:08.483774
msg sent: temperature -13.09, humidity 9.55, time 2022-01-10 02:18:08.985566
^Z
[10]+ Stopped python awsiotpub.py
glennfrey@raspberrypi:~/aws $ python awsiotpub.py
Connection returned result: 0
Connection successful
msg sent: temperature -18.92, humidity 8.68, time 2022-01-10 02:20:17.260846
msg sent: temperature -17.41, humidity 7.89, time 2022-01-10 02:20:17.762579
msg sent: temperature -12.58, humidity 5.17, time 2022-01-10 02:20:18.264644
msg sent: temperature -12.17, humidity 1.17, time 2022-01-10 02:20:18.767129
msg sent: temperature -14.47, humidity 1.69, time 2022-01-10 02:20:19.269859
msg sent: temperature -15.39, humidity 9.28, time 2022-01-10 02:20:19.772255
msg sent: temperature -15.34, humidity 5.57, time 2022-01-10 02:20:20.274768
msg sent: temperature -15.75, humidity 3.57, time 2022-01-10 02:20:20.777619
msg sent: temperature -11.51, humidity 0.80, time 2022-01-10 02:20:21.280297
msg sent: temperature -11.89, humidity 8.53, time 2022-01-10 02:20:21.782668
msg sent: temperature -15.71, humidity 9.68, time 2022-01-10 02:20:22.285085
msg sent: temperature -16.53, humidity 9.29, time 2022-01-10 02:20:22.787754
msg sent: temperature -19.37, humidity 6.64, time 2022-01-10 02:20:23.290599
msg sent: temperature -10.31, humidity 7.03, time 2022-01-10 02:20:23.793264
msg sent: temperature -10.40, humidity 3.68, time 2022-01-10 02:20:24.295714
msg sent: temperature -17.45, humidity 6.75, time 2022-01-10 02:20:24.798114
msg sent: temperature -13.72, humidity 4.64, time 2022-01-10 02:20:25.300485
msg sent: temperature -13.15, humidity 0.40, time 2022-01-10 02:20:25.802897
msg sent: temperature -16.03, humidity 9.86, time 2022-01-10 02:20:26.304922
msg sent: temperature -18.89, humidity 9.55, time 2022-01-10 02:20:26.806969
^Z
[10]+ Stopped python awsiotpub.py
glennfrey@raspberrypi:~/aws $
```

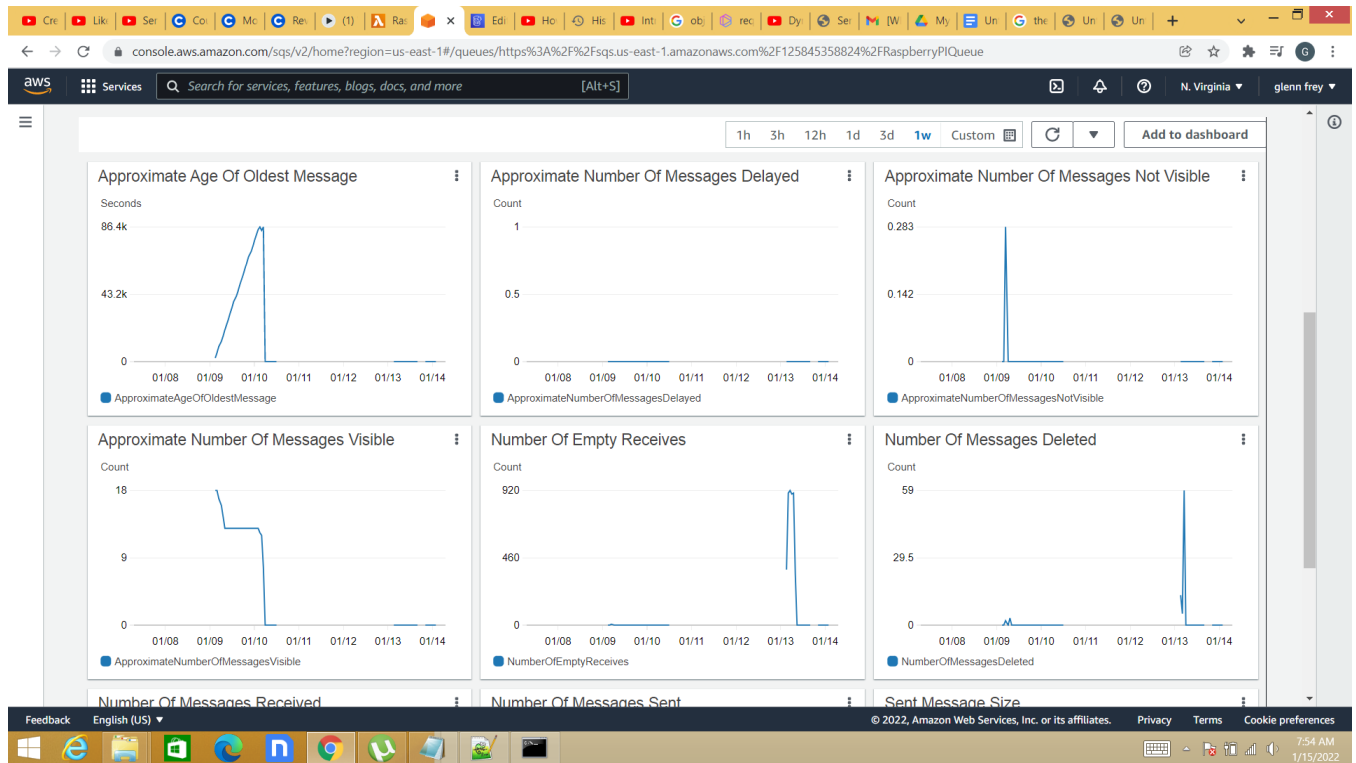
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

2:20 AM
1/10/2022

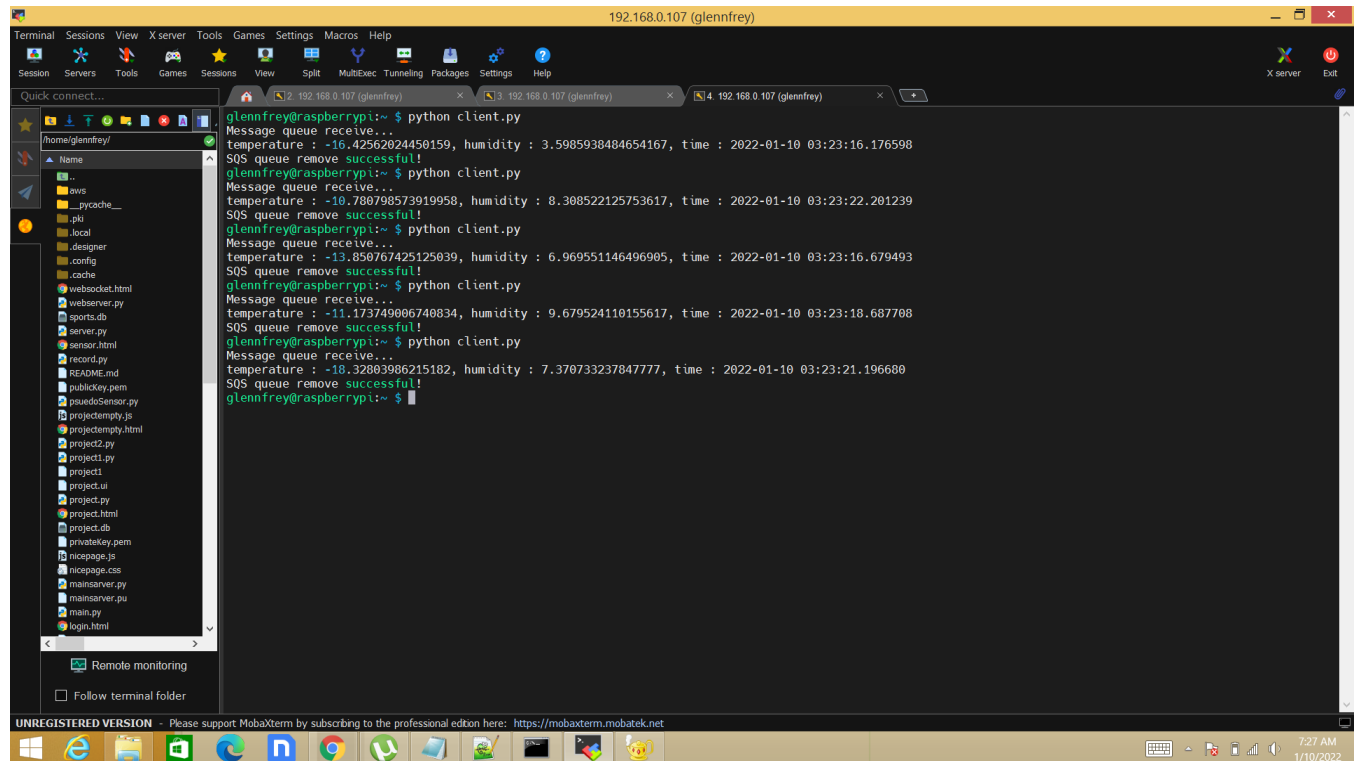
3.2 The data server sending messages (JSON timestamps, temperature, and humidity values) to AWS



3.3 The client connecting to AWS SQS



3.4 The client showing the timestamps of incoming SQS messages



```
glennfrey@raspberrypi:~$ python client.py
Message queue receive...
temperature : -16.42562024450159, humidity : 3.5985938484654167, time : 2022-01-10 03:23:16.176598
SQS queue remove successful!
glennfrey@raspberrypi:~$ python client.py
Message queue receive...
temperature : -10.780798573919958, humidity : 8.308522125753617, time : 2022-01-10 03:23:22.201239
SQS queue remove successful!
glennfrey@raspberrypi:~$ python client.py
Message queue receive...
temperature : -13.850767425125039, humidity : 6.969551146496905, time : 2022-01-10 03:23:16.679493
SQS queue remove successful!
glennfrey@raspberrypi:~$ python client.py
Message queue receive...
temperature : -11.173749006740834, humidity : 9.679524110155617, time : 2022-01-10 03:23:18.687788
SQS queue remove successful!
glennfrey@raspberrypi:~$ python client.py
Message queue receive...
temperature : -18.32803986215182, humidity : 7.370733237847777, time : 2022-01-10 03:23:21.196680
SQS queue remove successful!
glennfrey@raspberrypi:~$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

7:21 AM
1/10/2022