# Extended AWS IoT Connections 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 used AWS SQS as a trigger to Lambda Function to pass the data to dynamoDB. I created a role and policy in AWS IAM accordingly in order to facilitate the transfer of data in each stage.

I created two Lambda Function. The first one is named RaspberryPISQStoDB written in Nodejs to transfer data from AWS SQS to dynamoDB. The other one is named RaspberryPIReadMessage written also in Nodejs to used API Gateway to retrieve data in dynamoDB.

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

I assume the client program will show the connection is established first before displaying the data retrieved in the database.

I used my laptop as a host for the client program running in Windows 8.2 as this would closely represent a real world on premise monitoring device.

## 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 client2.py

Breaklines

```
import boto3
from boto3.dynamodb.condition import Key, Attr
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Message')

response = table.scan(
        FilterExpression=Attr('messageid').gte(0)
)

print("Connecting to AWS API Gateway...")
print("data from dynamoDB receive...")

for x in response["Items"]:
        print(x)
        table.delete_item(x)

print("db data received successfully remove!")
```

---

## 2.4 Lambda Function RaspberryPISQStoDB written in Nodejs

Breaklines

```
// Loads in the AWS SDK
const AWS = require('aws-sdk');

// Creates the document client specifing the region
// The tutorial's table is 'in us-east-1'
const ddb = new AWS.DynamoDB.DocumentClient({region: 'us-east-1'});

exports.handler = async (event, context, callback) => {
   // Captures the requestId from the context message
   const requestId = context.awsRequestId;

   // Handle promise fulfilled/rejected states
   await createMessage(requestId, event).then(() => {
      callback(null, {
         statusCode: 201,
         body: JSON.stringify(event['RaspberryPI']),
         headers: {
            'Access-Control-Allow-Origin' : '*'
         }
      });
   }).catch((err) => {
      console.error(err)
   })
};

// Function createMessage
// Writes message to DynamoDb table Message
function createMessage(requestId, event) {
   const params = {
      TableName: 'Message',
      Item: {
         'messageid' : requestId,
         'message' : event['RaspberryPI']
      }
   }
   return ddb.put(params).promise();
}
```

## 2.5 Lambda Function RaspberryPIReadMessage written in Nodejs

Breaklines

```
// Loads in the AWS SDK
const AWS = require('aws-sdk');

// Creates the document client specifing the region
// The tutorial's table is 'in us-east-1'
const ddb = new AWS.DynamoDB.DocumentClient({region: 'us-east-1'});

exports.handler = async (event, context, callback) => {
   // Handle promise fulfilled/rejected states
   await readMessage().then(data => {
      data.Items.forEach(function(item) {
         console.log(item.message)
      });
      callback(null, {
         // If success return 200, and items
         statusCode: 200,
         body: data.Items,
         headers: {
            'Access-Control-Allow-Origin': '*',
         },
      })
   }).catch((err) => {
      // If an error occurs write to the console
      console.error(err);
   })
};

// Function readMessage
// Reads 10 messages from the DynamoDb table Message
// Returns promise
function readMessage() {
   const params = {
      TableName: 'Message',
      Limit: 10
   }
   return ddb.scan(params).promise();
```

```
}
```

---

## 2.6 indexaws.html

Breaklines

```
<!DOCTYPE html>

<html>
<head>

   <title>Sensor</title>
   <style>
   .progress {
   position: relative;
   width: 100%;
   height: 60px;
   background: #9cbab4;
   border-radius: 5px;
   overflow: hidden;
   }

   .progress__fill {
   width: 0%;
   height: 100%;
   background: #009579;
   transition: all 0.2s;
   }

   .progress__text {
   position: absolute;
   top: 50%;
   right: 5px;
   transform: translateY(-50%);
   font: bold 14px "Quicksand", sans-serif;
   color: #ffffff;
   }
   table {
      font-family: arial, sans-serif;
      border-collapse: collapse;
      width:100%;
```

```
      padding:4px
    }
    td, th {
      border: 1px solid #CCCCCC;
      padding: 8px;
    }
    th {
      font-weight: bold;
      text-transform: uppercase;
    }
    .wrapper {
     display:grid;
     grid-template-columns: 70% 30%;
     grid-gap:1em;
    }
    #readData {
     background:#079992;
     text-align:center;
     padding:4px;
    }
    #analyzeData {
     background:#78e08f;
     text-align:center;
     padding:4px;
    }

    #progressBar {
     background:#6a89cc;
     text-align:center;
     padding:4px;
    }

    </style>
</head>

<body>
    <div class = "wrapper">
    <div>
    <p style="background-color:#079992;color:#00FF00;border-radius:5px; font-size:1em; padding:4px">Click
the "Read from DB" button to call API gateway and display result from AWS database below!</p>
    <p><div id="myDiv"></div></p>
    <button style="background-color:#333333;color:#00FF00;border-radius:5px; font-size:1em; padding:4px"
onclick="callAwsLambdaFunction()">Read from DB</button>
```

```
    <table >
      <thead>
      <tr >
        <th>ID</th>
        <th>Temperature</th>
        <th>Humidity</th>
        <th>Time</th>
      </tr>
      </thead>
      <tbody id = "tableData"></tbody>
    </table>
    <button style="background-color:#333333;color:#00FF00;border-radius:5px; font-size:1em; padding:4px" id =
tableDisplay >Display</button>

    </div>
    <div id = "analyzeData" >
    <button style="background-color:#333333;color:#00FF00;border-radius:5px; font-size:1em" id = analyzeData
>Analyze</button>
    <p>Average Temperature</p>
    <p id="avetempdisplay">data</p>
    <p>Average Humidity</p>
    <p id="avehumiditydisplay">data</p>
    <p>Minimum Temperature</p>
    <p id="mintempdisplay">data</p>
    <p>Minimum Humidity</p>
    <p id="minhumiditydisplay">data</p>
    <p>Maximum Temperature</p>
    <p id="maxtempdisplay">data</p>
    <p>Maximum Humidity</p>
    <p id="maxhumiditydisplay">data</p>

     </div>

    <div id = "progressBar" >
    <button style="background-color:#333333;color:#00FF00;border-radius:5px; font-size:1em; padding:4px
text-align:center" id = "recordData" >Record 10 Values</button>
    <p id="recordingData">data</p>
    <div class="progress">
    <div class="progress__fill"></div>
    <span class="progress__text">0%</span>
    </div>
    </div>

    <div id = "readData" >
```

```
   <button style="background-color:#333333;color:#00FF00;border-radius:5px; padding:4px; font-size:1em" id =
"readData" >Read Data</button>
   <p id="tempdisplay">data</p>
   <p id="humiditydisplay">data</p>

   <p id = "tempRange">Temperature in normal range</p>
   <p id = "humidityRange">Humidity in normal range</p>

   <button onclick="return window_close_onclick();"
style="background-color:#333333;color:#00FF00;border-radius:5px; padding:4px; font-size:1em"
>EXIT</button>
   </div>
   </div>

   <script>

   readData = document.getElementById("readData")
   tempRangeColor = document.getElementById("tempRange")
   humidityRangeColor = document.getElementById("humidityRange")


   readData.addEventListener("click", e => {
            fetch('http://localhost:8888/readData')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("tempdisplay").innerHTML = jsonResponse[1].toString();
            document.getElementById("humiditydisplay").innerHTML = jsonResponse[0].toString();
            if (jsonResponse[1]>30 && jsonResponse[1]<40){
            document.getElementById("tempRange").innerHTML = "Temperature in normal range";
            tempRangeColor.style.color = 'black';
}           else {
            document.getElementById("tempRange").innerHTML = "Warning!!! temperature beyond normal
condition";
            tempRangeColor.style.color = 'red';
}
            if (jsonResponse[0]>20 && jsonResponse[0]<50){
            document.getElementById("humidityRange").innerHTML = "Humidity in normal range";
            humidityRangeColor.style.color = 'black';
}           else {
            document.getElementById("humidityRange").innerHTML = "Warning!!! humidity beyond normal
condition";
            humidityRangeColor.style.color = 'red';
}
})
```

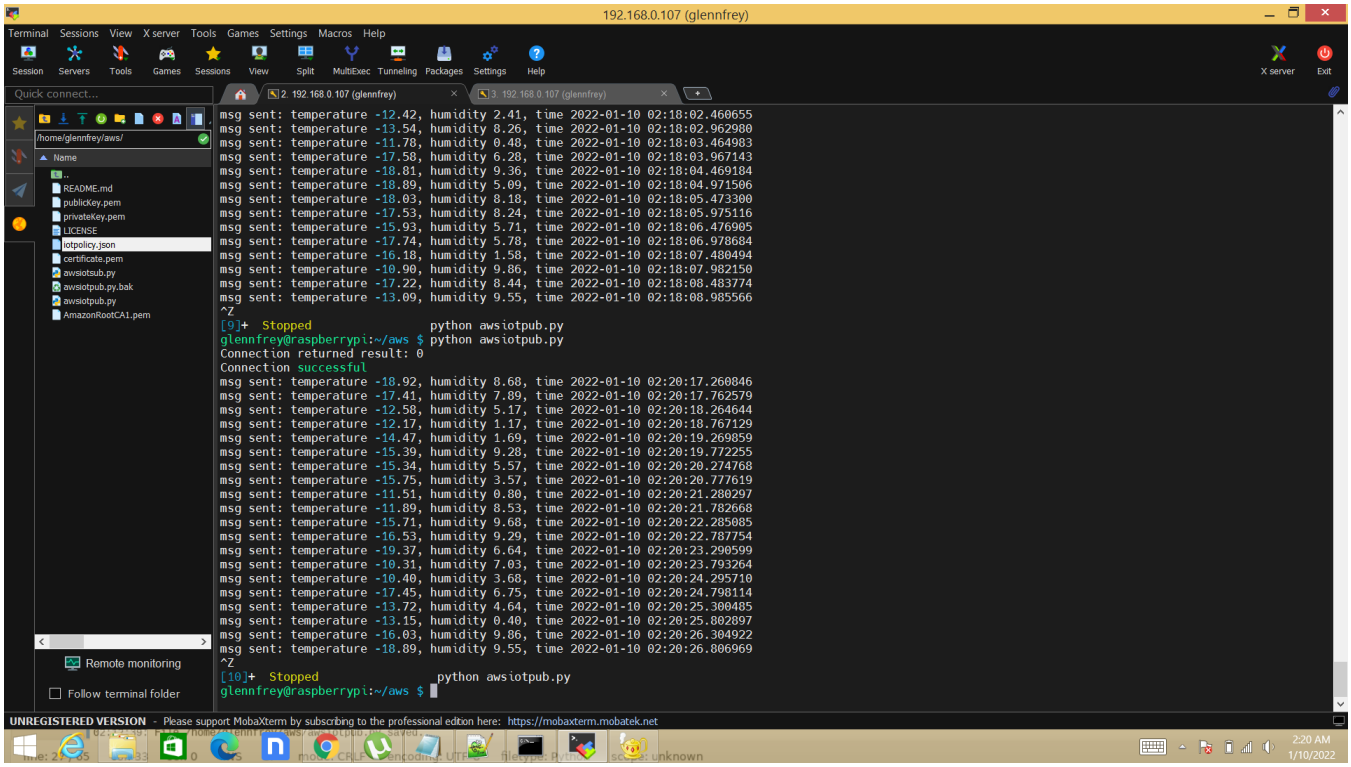
```
})

    function updateProgressBar(progressBar, value) {
            value = Math.round(value);
            progressBar.querySelector(".progress__fill").style.width = `${value}%`;
            progressBar.querySelector(".progress__text").textContent = `${value}%`;
}
    myProgressBar = document.querySelector(".progress")
    bar = 0

    recordData = document.getElementById("recordData")

    recordData.addEventListener("click", e => {
            for (let i = 0; i < 10; i++) {

            fetch('http://localhost:8888/recordData', {"method" : "POST"})
            .then(response => response.json()
            .then(jsonResponse => {
            document.getElementById("recordingData").innerHTML = bar + " " + jsonResponse.message;
            setTimeout( console.log(bar + " " + jsonResponse.message+ new Date()), 3000 * i)
            updateProgressBar(myProgressBar, bar*10)
            bar =+ i + 1;
}))
})))
    analyzeData = document.getElementById("analyzeData")

    analyzeData.addEventListener("click", e => {
            fetch('http://localhost:8888/analyzeDataAveTemp')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("avetempdisplay").innerHTML = jsonResponse.toString();
})
            fetch('http://localhost:8888/analyzeDataAveHumidity')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("avehumiditydisplay").innerHTML = jsonResponse.toString();
})
            fetch('http://localhost:8888/analyzeDataMinTemp')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("mintempdisplay").innerHTML = jsonResponse.toString();
})
            fetch('http://localhost:8888/analyzeDataMinHumidity')
            .then(response => response.json())
```

```
        .then(jsonResponse => { console.log(jsonResponse)
        document.getElementById("minhumiditydisplay").innerHTML = jsonResponse.toString();
})
        fetch('http://localhost:8888/analyzeDataMaxTemp')
        .then(response => response.json())
        .then(jsonResponse => { console.log(jsonResponse)
        document.getElementById("maxtempdisplay").innerHTML = jsonResponse.toString();
})
        fetch('http://localhost:8888/analyzeDataMaxHumidity')
        .then(response => response.json())
        .then(jsonResponse => { console.log(jsonResponse)
        document.getElementById("maxhumiditydisplay").innerHTML = jsonResponse.toString();
})
})

   tableDisplay = document.getElementById("tableDisplay")

   tableDisplay.addEventListener("click", e => {
        fetch('http://localhost:8888/tableDisplay')
        .then(response => response.json())
        .then(jsonResponse => { console.log(jsonResponse)
        console.log(typeof jsonResponse)
        console.log(jsonResponse[0][1])
        console.log(jsonResponse[0])
        loadTableData(jsonResponse);
})
})
        function loadTableData(jsonResponse){
        tableBody = document.getElementById('tableData')
        dataHtml = " ";
        for (let i = 0; i < 10; i++) {
           dataHtml +=
`<tr><td>${jsonResponse[i][0]}</td><td>${jsonResponse[i][1]}</td><td>${jsonResponse[i][2]}</td><td>${jso
nResponse[i][3]}</td></tr>`;
        }
        console.log(dataHtml)
        tableBody.innerHTML = dataHtml;
        }


   function window_close_onclick(){
        if(confirm("Do you want to exit?")){
        fetch('http://localhost:8888/exit')
```

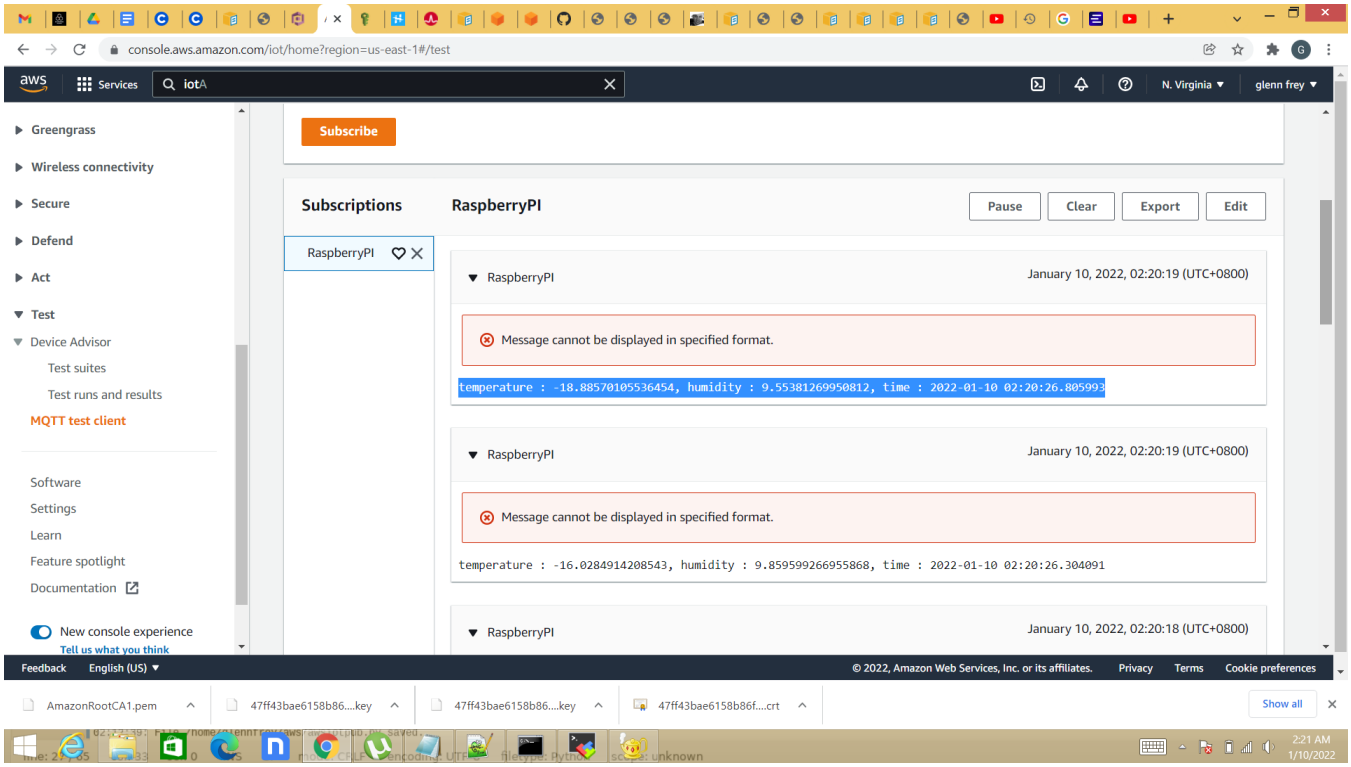
```
            let new_window =
            open(location, '_self');

            new_window.close();

            return false;
            }
}
                async function callAwsLambdaFunction() {
                        fetch( 'https://we1zdt5152.execute-api.us-east-1.amazonaws.com/production',  {
                                method: 'GET'
                         })
                        .then(response => response.json())
                        .then((response) => {
                                console.log(response.body);
                                response.body.forEach(element => {
                                document.getElementById("myDiv").innerHTML +=
"<p>"+element.message+"</p>";
                        })
                        });
      }


   </script>
</body>
</html>
```

# 3. Screen Capture

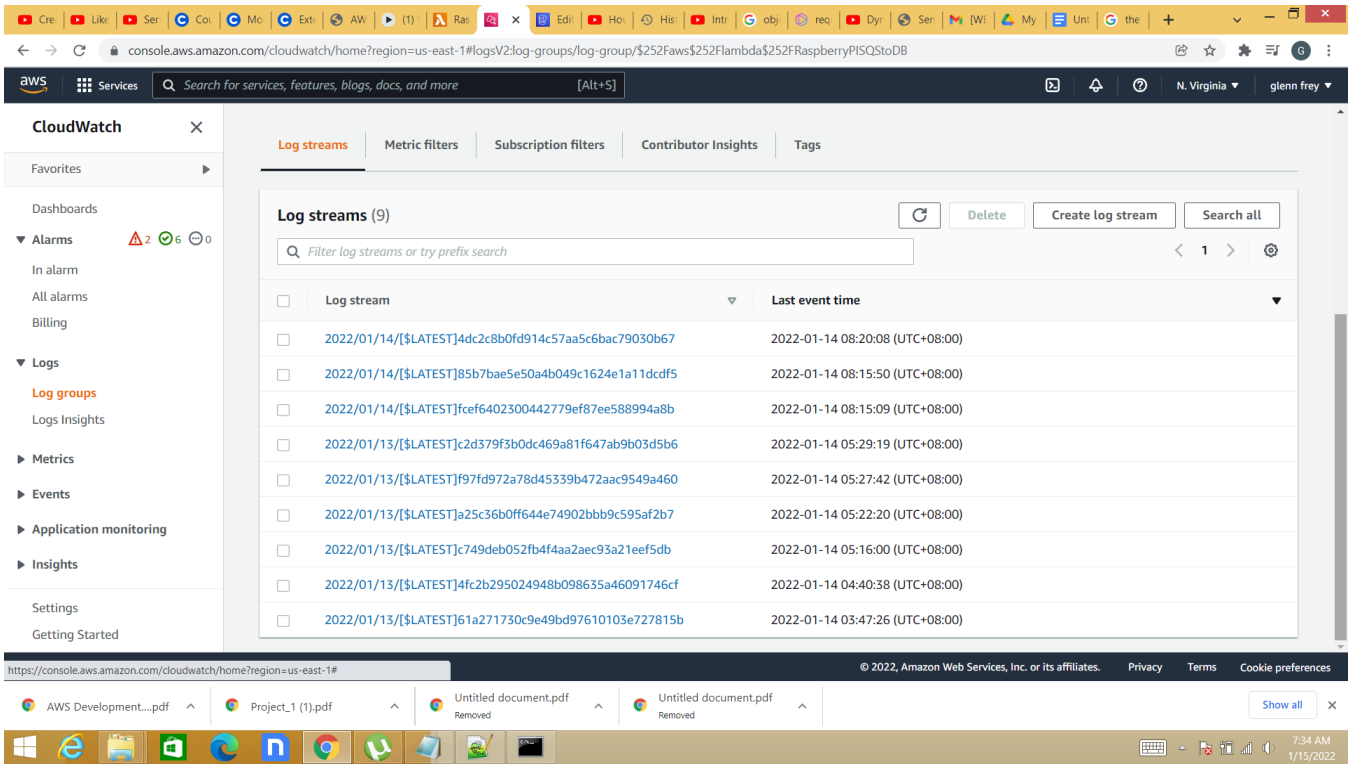
## 3.1 The data server tracing it's connection to AWS



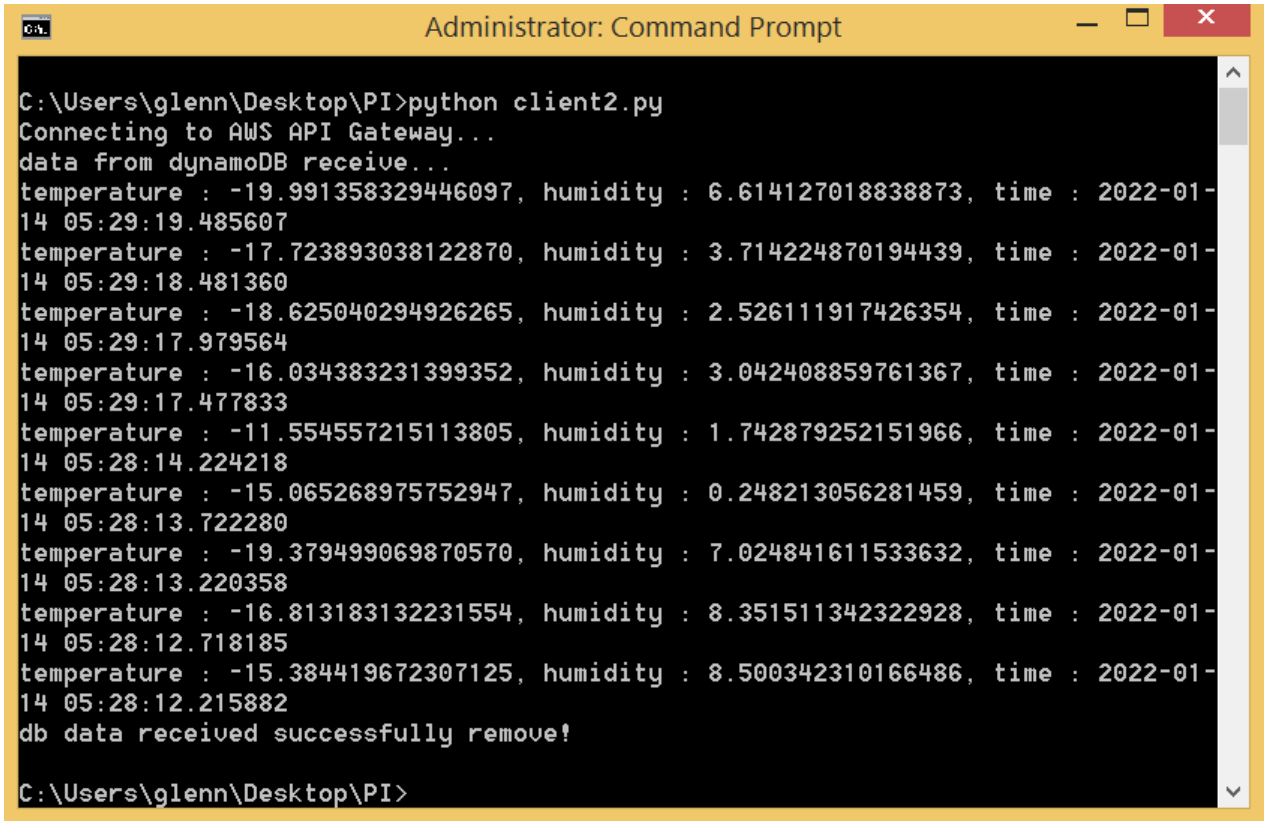
3.2 The data server sending messages (JSON timestamps, temperature, and humidity values) to AWS



3.3 The client connecting to the REST API



3.4 The client showing the retrieved messages from API transactions



3.5 (Optional if you attempted it) Use of the previously developed HTML UI as an alternate client.