ECEA 5348- Project 1

Implementation/Assumption Notes

Objective- Create an Amazon Web Services connection that will:

1. Send data to an AWS IoT Thing that will display temperature, humidity, and timestamps read in from a pseudo sensor.

2. Display communications status (steps of connection, messages sent) on the command line, and capture a successful connection session.

3. Contain a rule to the AWS IoT Thing to send incoming data to an SQS Queue.

4. Have viewable data in the AWS SQS Queue in the AWS console to see messages arriving from the connected Python data server.

5. Use the Boto3 SQS library to act as a client that removes messages from the SQS queue and displays them on the command line.

6. Data from the pseudo sensor is humidity between 0 and 100%, and temperature between -20 and 100 degrees Fahrenheit.

Additional Notes

1. IDE is set up with: AWS Cloud9, Win10, Python, MQTT.

2. PDF submission will include:

    a. The data server tracing it's connection to AWS

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

    c. The client connecting to AWS SQS

    d. The client showing the timestamps of incoming SQS messages

    e. Captures of failed communication for the server or client.

Publish.py (privacy keys removed)

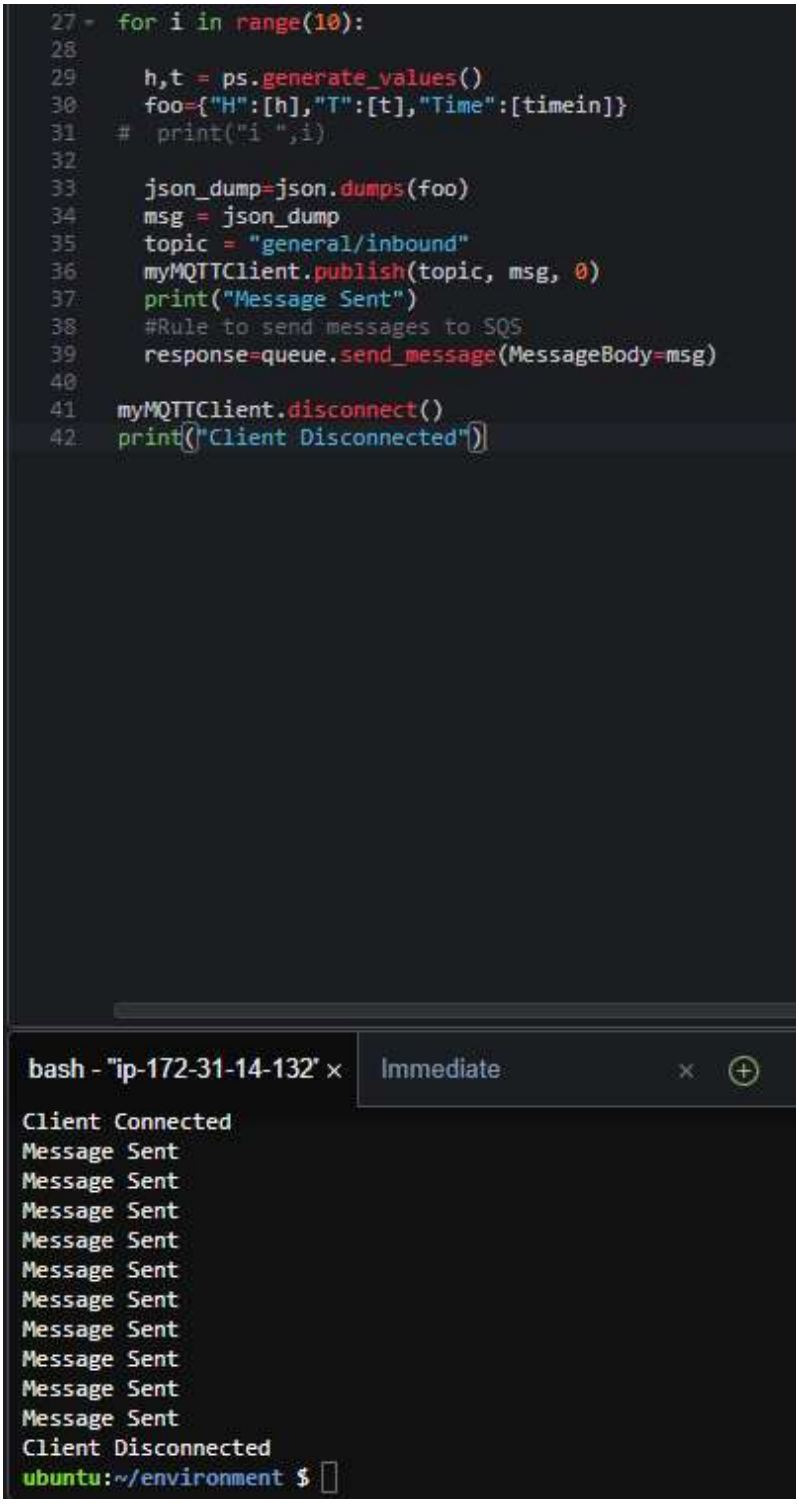
```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from pseudoSensor import PseudoSensor
from datetime import datetime
import json
import boto3
import sys

#Set service resource
sqs=boto3.resource('sqs', region_name='us-east-2', aws_access_key_id="
Wouldntyouliketoknow ", aws_secret_access_key="Wouldntyouliketoknow")

#Define queue for messages
queue=sqs.get_queue_by_name(QueueName='5318_Proj1')

#Define client parameters
myMQTTClient = AWSIoTMQTTClient("dojodevice1")
myMQTTClient.configureEndpoint("endpoint", 8883)
myMQTTClient.configureCredentials("./AmazonRootCA1 (1).pem","./
Wouldntyouliketoknow ", "./ Wouldntyouliketoknow.crt")


#Connect to client
myMQTTClient.connect()
print("Client Connected")


#Gettimestamp in string format
rn=datetime.now()
timein=str(rn.strftime("%H:%M:%S"))

ps=PseudoSensor()


#Generate 10 weather values
for i in range(10):

  h,t = ps.generate_values()
  foo={"H":[h],"T":[t],"Time":[timein]}

#Reformat json into something humanly readable
  json_dump=json.dumps(foo)
  msg = json_dump
  topic = "general/inbound"
```

```
  #Publish weather data
  myMQTTClient.publish(topic, msg, 0)
  print("Message Sent")
  #Rule to send messages to SQS
  response=queue.send_message(MessageBody=msg)

#Disconnect from MQTT client
myMQTTClient.disconnect()
print("Client Disconnected")
```

boto_retrieve.py

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from pseudoSensor import PseudoSensor
from datetime import datetime
import json
import boto3
import sys

#Get service resource
sqs=boto3.resource('sqs', region_name='us-east-2', aws_access_key_id="
Wouldntyouliketoknow ", aws_secret_access_key=" Wouldntyouliketoknow ")

#Get my queue
queue=sqs.get_queue_by_name(QueueName='5318_Proj1')

for message in queue.receive_messages(MessageAttributeNames=['Author']):
    # Get the custom author message attribute if it was set
    author_text = ''if message.message_attributes is not None:
        author_name = message.message_attributes.get('Author').get('StringValue')
        if author_name:
            author_text = ' ({0})'.format(author_name)

    # Print out the body and author (if set)
    print('Hello, {0}!{1}'.format(message.body, author_text))

    # Let the queue know that the message is processed, remove from queue
    message.delete()
```

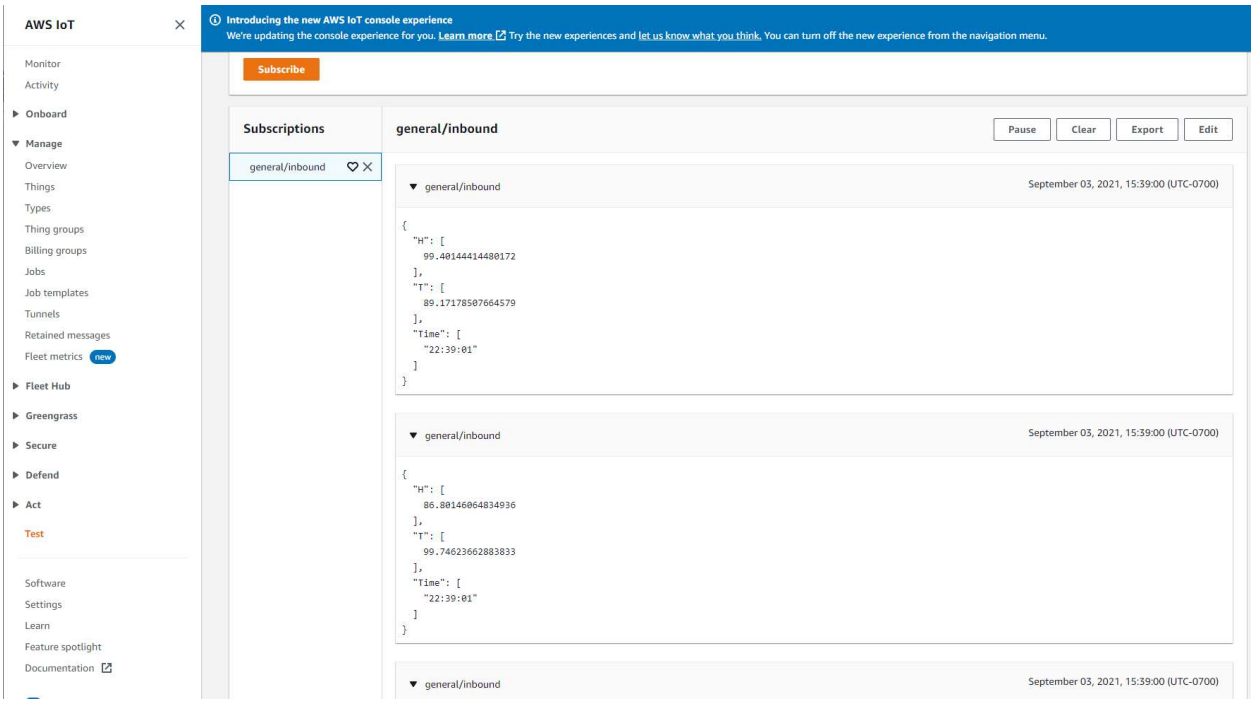
Capture 1- Data server tracing connection to AWS:



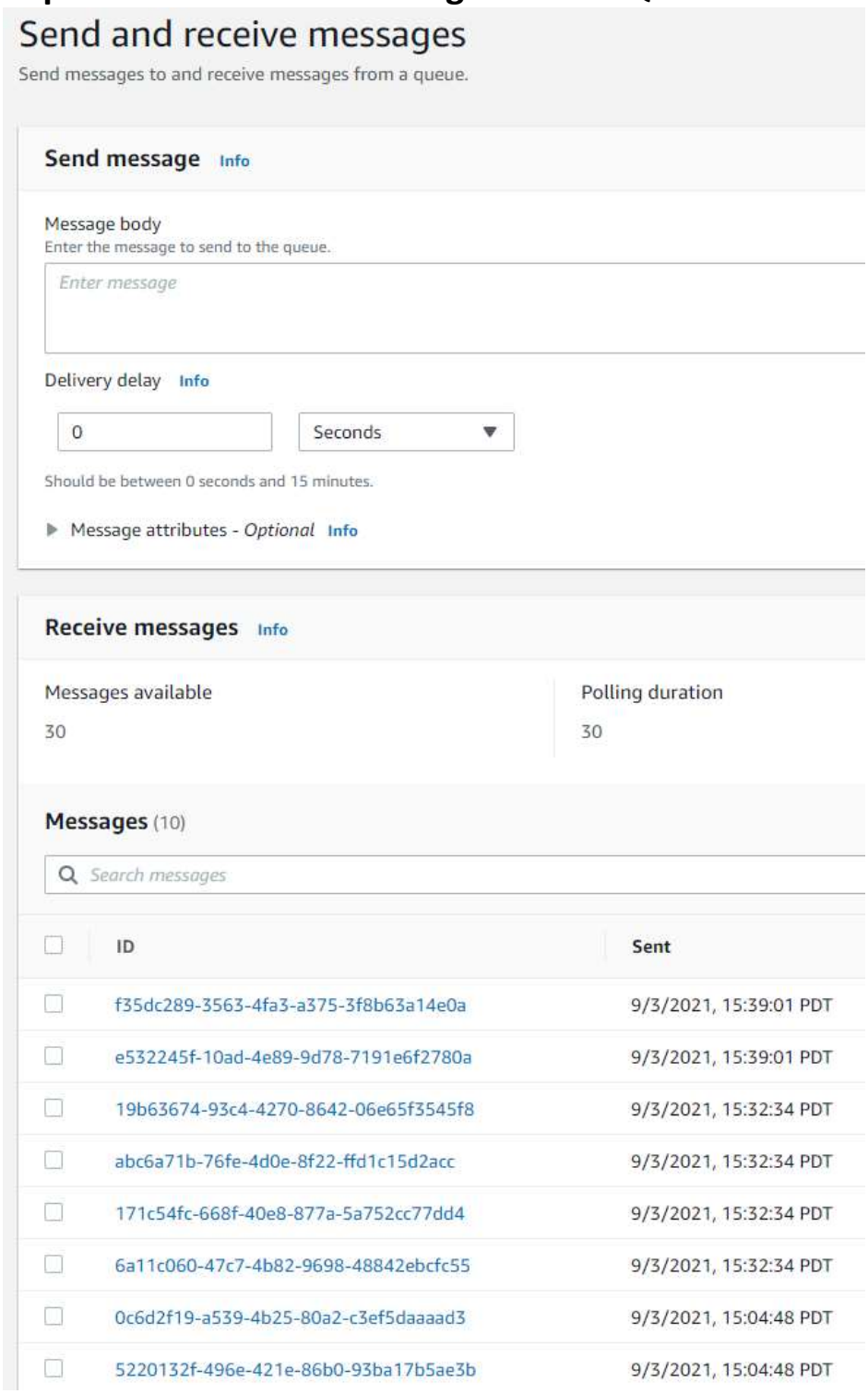
```
for i in range(10):

    h,t = ps.generate_values()
    foo={"H":[h],"T":[t],"Time":[timein]}
#   print("i ",i)

    json_dump=json.dumps(foo)
    msg = json_dump
    topic = "general/inbound"
    myMQTTClient.publish(topic, msg, 0)
    print("Message Sent")
    #Rule to send messages to SQS
    response=queue.send_message(MessageBody=msg)

myMQTTClient.disconnect()
print("Client Disconnected")
```

```
bash - "ip-172-31-14-132" ×   Immediate

Client Connected
Message Sent
Message Sent
Message Sent
Message Sent
Message Sent
Message Sent
Message Sent
Message Sent
Message Sent
Message Sent
Client Disconnected
ubuntu:~/environment $
```

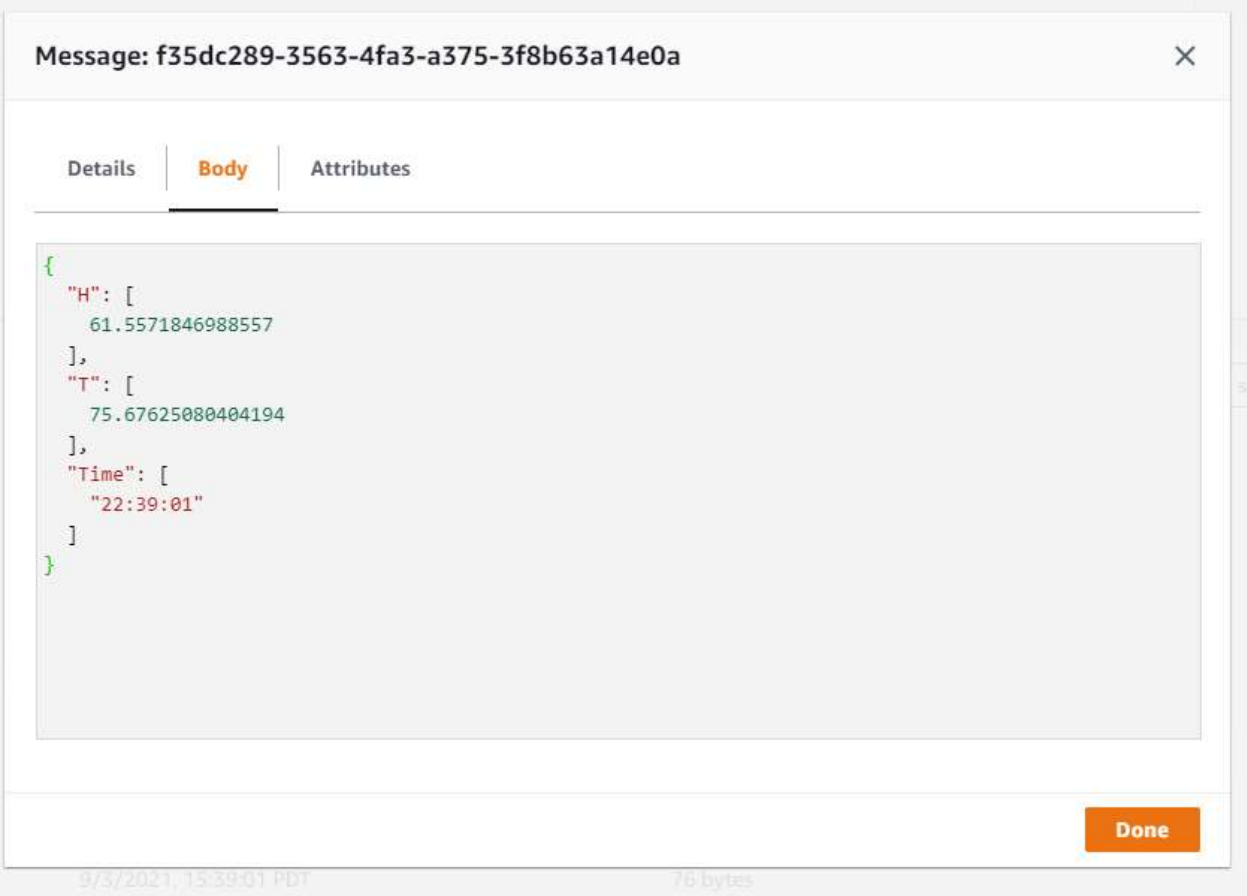
Capture 2- Data Server sending messages to AWS:



Capture 3- Client connecting to AWS SQS:



Capture 4- Client showing timestamps of incoming SQS messages:



Capture 5- Failed server communication: