

CPE 4040: Data Collection and Analysis, Spring 2024

Laboratory Report #6

Heart Rate Sensor Application with AWS IoT

Team Members: Anindita Deb and Damisi Kayode


Electrical and Computer Engineering

Kennesaw State University

Faculty: Dr. Jeffrey L Yiin

Date of Lab Session: March 25, 2024 and March 27, 2024


I. Objective

- 
1. To gain knowledge on interfacing and reading data from a heart rate sensor.
 2. To develop an AWS IoT application using Raspberry Pi.
 3. To understand how to store sensor data using DynamoDB app on AWS.
 4. To learn basic data cleaning techniques for raw heart rate sensor data.

II. Material List

AWS IoT Resource for setting up the account

Hardware:

- 
- Heart rate sensor must be soldered to header pins.
 - 5 male to female jumper wires
 - breadboard

Software:

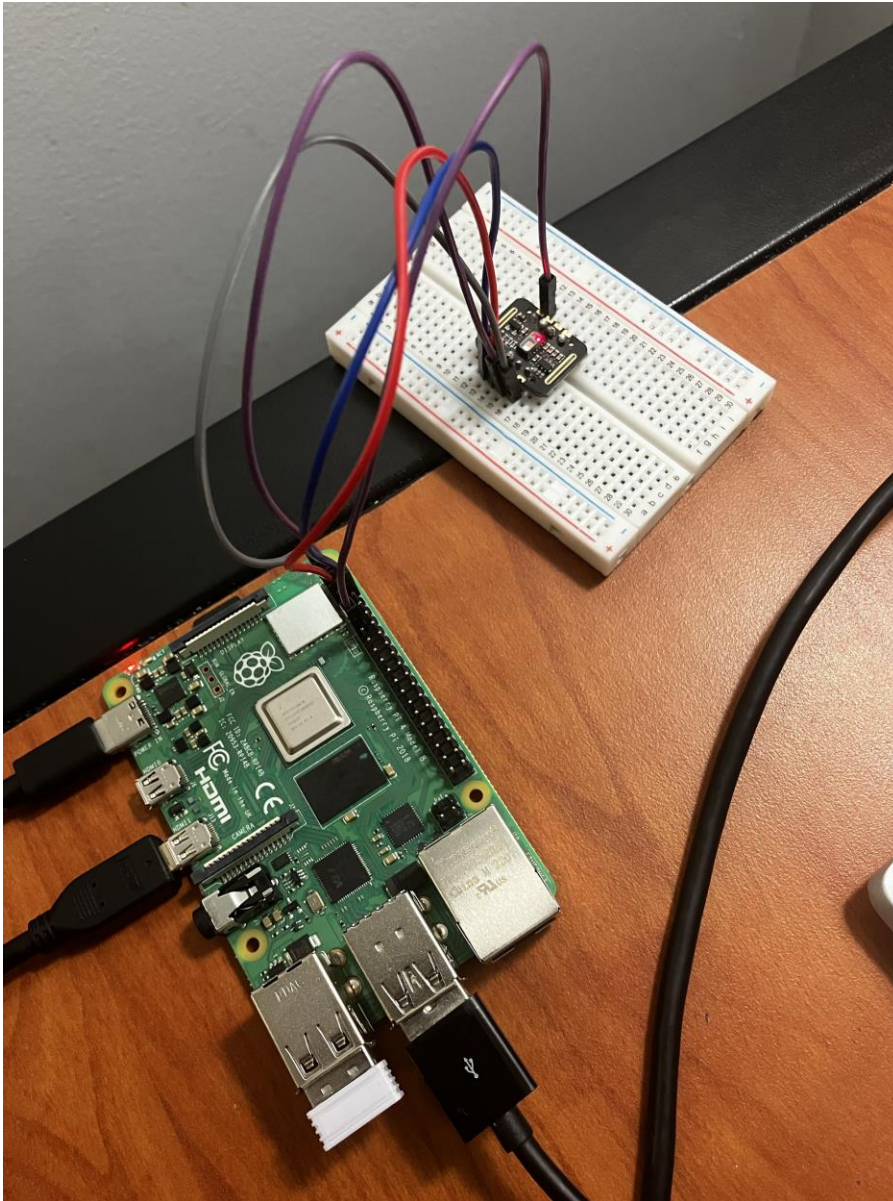
- 
- Provided Python scripts
 - AWS IoT Core
 - Remote Desktop Connection

III. Lab Procedures and Results

1. Plug the power adapter into the Raspberry Pi and power it up. Next, open Remote Desktop Connection or SSH connection on your laptop and connect to the Raspberry Pi

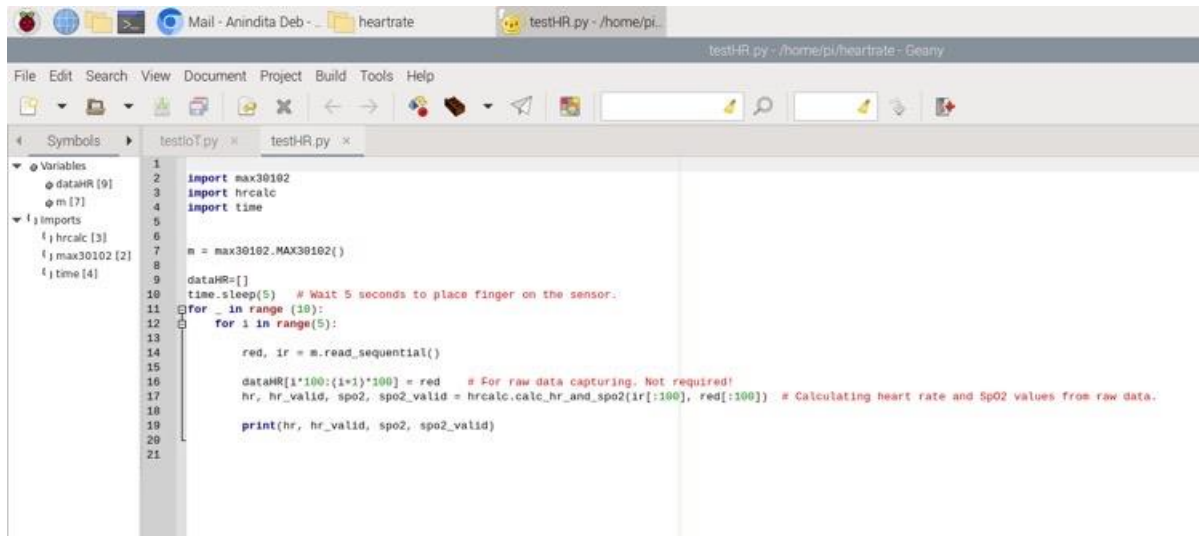
Stage 1: Setting Up Heart Rate Sensor on the Raspberry Pi:

2. Connect the heart rate sensor to Raspberry Pi using the connection table below:



3. Make sure the I2C interface on the Raspberry Pi is enabled.
4. Create a folder for the heart rate sensor application on your Raspberry Pi and download the following Python scripts:
 - a. max30102.py: library for the heart rate sensor;
 - b. hrcalc.py: procedures and algorithm to calculate heart rate and SpO2 values;
 - c. testHR.py: test data collection.
5. Modify the code in testHR.py with a for loop to print the data 10 times. Run the code and place your index finger on the sensor. You should see Heart Rate and SpO2 values displayed on the terminal for each loop iteration. Verify that the data appear valid for all samples, indicating that the heart rate sensor is functioning correctly.

CPE4040 Lab Report

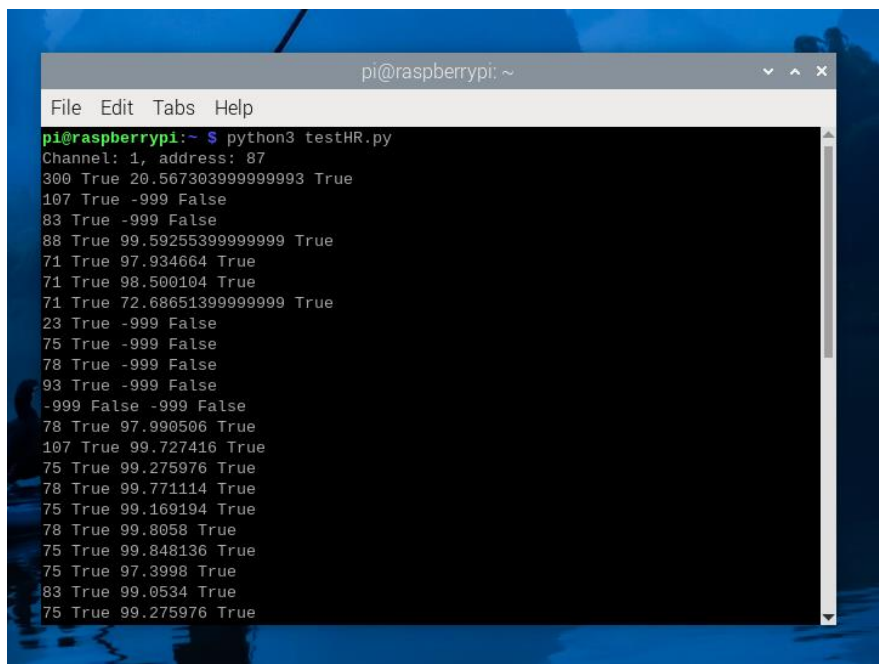


```
1 import max30102
2 import hrcalc
3 import time
4
5 m = max30102.MAX30102()
6
7 dataHR=[]
8 time.sleep(5) # Wait 5 seconds to place finger on the sensor.
9
10 for i in range(10):
11     for j in range(5):
12         red, ir = m.read_sequential()
13
14         dataHR[i*100:(i+1)*100] = red # For raw data capturing. Not required!
15         hr, hr_valid, spo2, spo2_valid = hrcalc.calc_hr_and_spo2(ir[:100], red[:100]) # Calculating heart rate and SpO2 values from raw data.
16
17         print(hr, hr_valid, spo2, spo2_valid)
```

Question: What is SpO2? How is it measured by the heart rate sensor?

SpO2 stands for peripheral capillary oxygen saturation. It's a measure of the amount of oxygen-carrying hemoglobin in your blood.

The heart rate sensor measures SpO2 by emitting light into the skin and detecting the amount of light that is absorbed or reflected back. Based on the differences in light absorption, the sensor calculates the percentage of oxygen saturation in the blood.



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ python3 testHR.py
Channel: 1, address: 87
300 True 20.567303999999993 True
107 True -999 False
83 True -999 False
88 True 99.59255399999999 True
71 True 97.934664 True
71 True 98.500104 True
71 True 72.68651399999999 True
23 True -999 False
75 True -999 False
78 True -999 False
93 True -999 False
-999 False -999 False
78 True 97.990506 True
107 True 99.727416 True
75 True 99.275976 True
78 True 99.771114 True
75 True 99.169194 True
78 True 99.8058 True
75 True 99.848136 True
75 True 97.3998 True
83 True 99.0534 True
75 True 99.275976 True
```

Stage 2: Creating an AWS IoT "Thing":

6. Go to <https://aws.amazon.com/> and create a free AWS account. Once you have signed up, log in to the account and navigate to the AWS Management Console. From there, search and open "IoT Core" under "Services".

```

pi@raspberrypi: ~
File Edit Tabs Help

KeyboardInterrupt

pi@raspberrypi:~$ S ping a18vqeue1wbg61-ats.iot.us-east-2.amazonaws.com
PING a18vqeue1wbg61-ats.iot.us-east-2.amazonaws.com (3.20.1.100) 56(84) bytes of
data.
64 bytes from ec2-3-20-1-100.us-east-2.compute.amazonaws.com (3.20.1.100): icmp_
seq=1 ttl=243 time=35.4 ms
64 bytes from ec2-3-20-1-100.us-east-2.compute.amazonaws.com (3.20.1.100): icmp_
seq=2 ttl=243 time=38.5 ms
64 bytes from ec2-3-20-1-100.us-east-2.compute.amazonaws.com (3.20.1.100): icmp_
seq=3 ttl=243 time=39.1 ms
64 bytes from ec2-3-20-1-100.us-east-2.compute.amazonaws.com (3.20.1.100): icmp_
seq=4 ttl=243 time=39.2 ms
64 bytes from ec2-3-20-1-100.us-east-2.compute.amazonaws.com (3.20.1.100): icmp_
seq=5 ttl=243 time=38.9 ms
64 bytes from ec2-3-20-1-100.us-east-2.compute.amazonaws.com (3.20.1.100): icmp_
seq=6 ttl=243 time=38.2 ms
64 bytes from ec2-3-20-1-100.us-east-2.compute.amazonaws.com (3.20.1.100): icmp_
seq=7 ttl=243 time=46.3 ms
^ [64 bytes from ec2-3-20-1-100.us-east-2.compute.amazonaws.com (3.20.1.100): icm
p_seq=8 ttl=243 time=39.9 ms
64 bytes from ec2-3-20-1-100.us-east-2.compute.amazonaws.com (3.20.1.100): icmp_
seq=9 ttl=243 time=38.3 ms
^C

```

7. To create a thing (i.e., an IoT device) in AWS IoT, **three steps are needed:**
 - Step 1: Register the device
 - Step 2: Create and save the device certificate, public key, and private key. Download the root CA as well.
 - Step 3: Create and attach a policy to the thing key/certificate.

In the AWS IoT Core page, click on "Connect" on the sidebar, then "Connect one device" and follow the device setup instructions. Note: Ensure that Python is selected as the SDK language, and when you download the ZIP folder, transfer it onto your Pi.
8. . Verify that your Thing is now properly created. In the AWS IoT Core page, choose "Manage" from the sidebar, then click on "All devices" followed by "Things". Choose Create Things then click on Create a Single Thing. Provide a name for your heart rate sensor device, then click Next.
9. Now, verify that your certificates are active. Click on your Thing from Step 8, and ensure that a certificate is in the Certificates tab, and that it is active. Additionally, verify the certificates 2 are in your Pi directory where you unzipped the ZIP folder from Step 7 where the device was connected. If a certificate does not appear in this tab, you can click on Create Certificate. Check to set the certificate as Active. Note: The ZIP folder includes the device certificate, the public key, the private key and the Amazon Root CA 1.
10. In the AWS IoT Core page, select "Security" from the sidebar and choose "Policies". If your policy already appears here, click on your policy then Edit Active Version, then continue to the next step. If not, click Create policy to create a new policy for your device. Provide a name for the policy and find the Policy Document section.

Missing screenshots for creation of AWS IoT Thing,
Certificates activated, and policy configuration

-4

Using the lab manual as a reference for your report is good, but avoid copy/pasting the manual. Try and write it in your own words

11. In Policy Document, enter wildcard (*) for both Policy Action and Policy Resource. Click the “Allow” box for Policy Effect then click either Save as new version or Create.
Question: What is a AWS IoT Core policy? What is the purpose?
12. To attach the policy to your certificate, go to “Certificates” (under “Security”) in the AWS IoT Core page. Click the certificate you created and choose Attach policy from the Actions menu. Select the policy from the list and click Attach policies. Now AWS IoT is ready to receive data from the device.
13. Go back to the Raspberry Pi and create a folder named “cert” in your heart rate application folder. Transfer the device certificate, private key, and root CA files into the “cert” folder. Before transferring, you can rename the files as follows:

(we did not and changed the testIoT code instead to the updated names)

- Device certificate: RaspberryPi-cert.pem.crt
- Private key: RaspberryPi-private.pem.key
- Root CA: RootCA1.pem

Stage 3: Connecting the Heart Rate Sensor to AWS IoT:

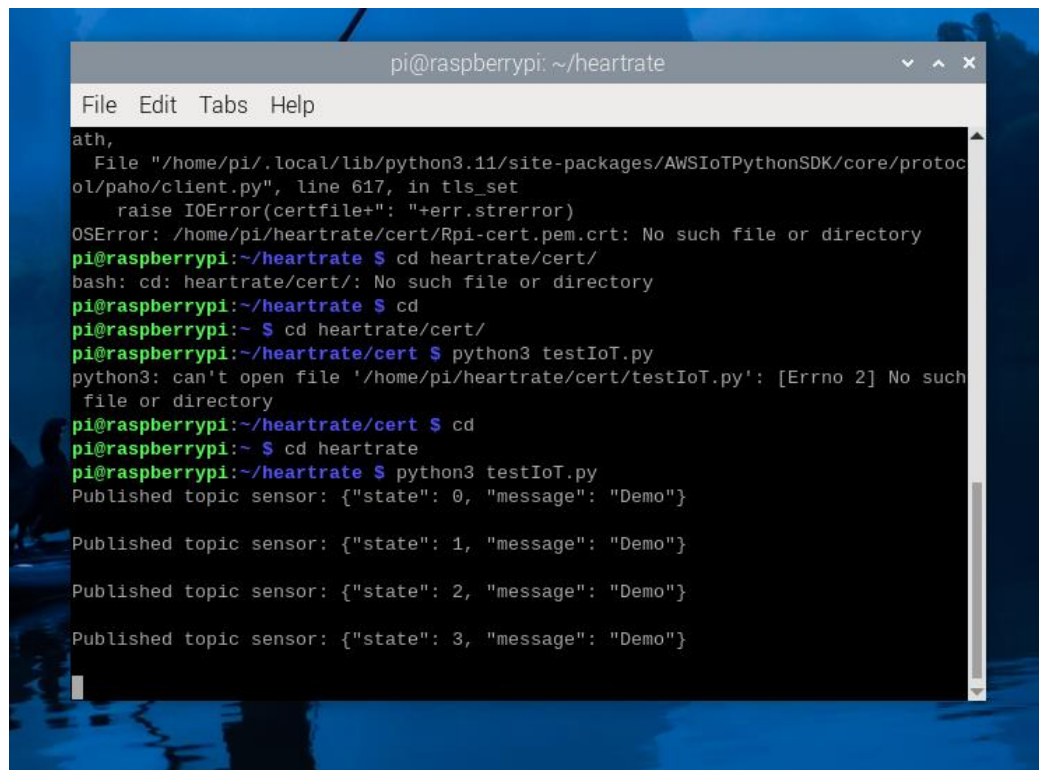
14. Download testIoT.py python script to the heart rate sensor application folder in your Raspberry Pi.
15. . To make a connection from Raspberry Pi to AWS IoT, we need to install AWS Python SDK.
Open a terminal and enter: pip3 install AWSIoTPythonSDK
16. . Once the installation is complete, open testIoT.py and modify the following parameters:
 - For host URL, go to AWS IoT Core. Scroll down in the menu pane and click “Settings”. Your AWS account has a device data endpoint (for both REST API and MQTT) to connect to AWS. Copy the endpoint link and replace the one in the code.
 - For certPath, replace it with your own app folder name.
 - Put your IoT device name as clientId.
 - Keep the topic as “sensor”.
 - You do not need to change the filenames for Root-CA, private key and device certificate if you already did so in earlier step.
17. Run the modified code and check published demo messages displayed in the terminal.

You can see that we renamed values in the code here in this particular line, #3.



```
8 # Init AWSIoTClient
9 myAWSIoTClient = None
10 myAWSIoTClient = AWSIoTClient(clientId)
11 myAWSIoTClient.configureEndpoint(host, 8883)
12 myAWSIoTClient.configureCredentials("{}AmazonRootCA1.pem".format(certPath), "{}e519be21f72005e9a0ab1e985afcf66f6b23dc42008295a966e652154c2478-private.pem.key".format(certPath), "{}e519be21f72005e9a0ab1e985a
```

A little hard to read



A terminal window on a Raspberry Pi showing the execution of a Python script. The window title is 'pi@raspberrypi: ~/heartrate'. The menu bar includes 'File', 'Edit', 'Tabs', and 'Help'. The terminal output shows an error from the AWS IoT Python SDK, followed by the user navigating to the 'cert' directory and running 'python3 testIoT.py'. The script successfully publishes four MQTT messages to the 'sensor' topic, each with a state value from 0 to 3 and the message 'Demo'.

```
pi@raspberrypi: ~/heartrate
File Edit Tabs Help
ath,
File "/home/pi/.local/lib/python3.11/site-packages/AWSIoTPythonSDK/core/protocol/paho/client.py", line 617, in tls_set
    raise IOError(certfile+": "+err.strerror)
OSError: /home/pi/heartbeat/cert/Rpi-cert.pem.crt: No such file or directory
pi@raspberrypi:~/heartrate $ cd heartbeat/cert/
bash: cd: heartbeat/cert/: No such file or directory
pi@raspberrypi:~/heartrate $ cd
pi@raspberrypi:~ $ cd heartbeat/cert/
pi@raspberrypi:~/heartrate/cert $ python3 testIoT.py
python3: can't open file '/home/pi/heartbeat/cert/testIoT.py': [Errno 2] No such file or directory
pi@raspberrypi:~/heartrate/cert $ cd
pi@raspberrypi:~ $ cd heartbeat
pi@raspberrypi:~/heartrate $ python3 testIoT.py
Published topic sensor: {"state": 0, "message": "Demo"}

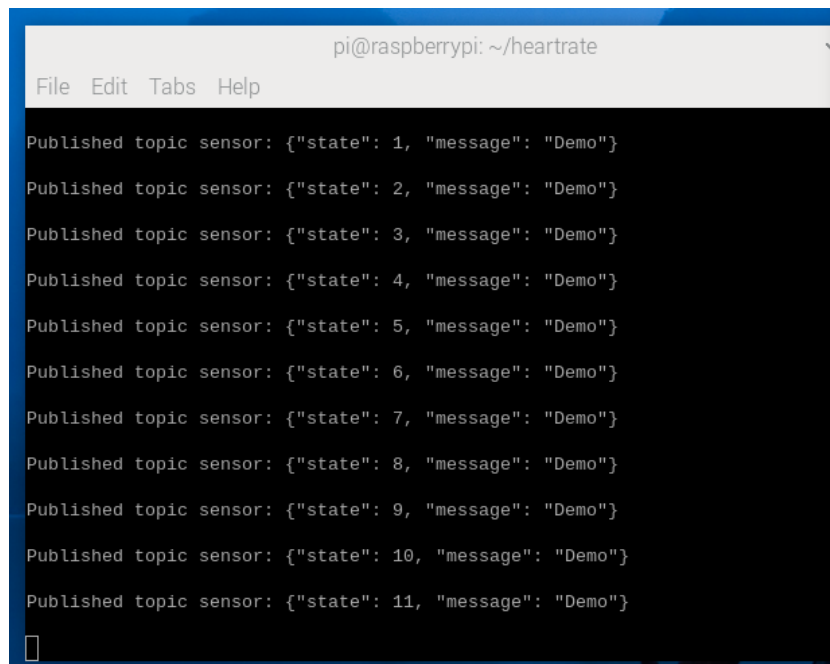
Published topic sensor: {"state": 1, "message": "Demo"}

Published topic sensor: {"state": 2, "message": "Demo"}

Published topic sensor: {"state": 3, "message": "Demo"}


```

18. Go to AWS IoT Core page, choose “Test” followed by “MQTT test client” from the menu pane. This will start a MQTT Client service. In Subscribe to a topic, enter the topic name (sensor) then click Subscribe. You should now be able to see the demo messages being received in real-time at the bottom of the page.



A terminal window on a Raspberry Pi showing a continuous stream of MQTT messages received on the 'sensor' topic. The window title is 'pi@raspberrypi: ~/heartrate'. The menu bar includes 'File', 'Edit', 'Tabs', and 'Help'. The terminal output shows 11 messages, each with a state value from 1 to 11 and the message 'Demo'.

```
pi@raspberrypi: ~/heartrate
File Edit Tabs Help

Published topic sensor: {"state": 1, "message": "Demo"}

Published topic sensor: {"state": 2, "message": "Demo"}

Published topic sensor: {"state": 3, "message": "Demo"}

Published topic sensor: {"state": 4, "message": "Demo"}

Published topic sensor: {"state": 5, "message": "Demo"}

Published topic sensor: {"state": 6, "message": "Demo"}

Published topic sensor: {"state": 7, "message": "Demo"}

Published topic sensor: {"state": 8, "message": "Demo"}

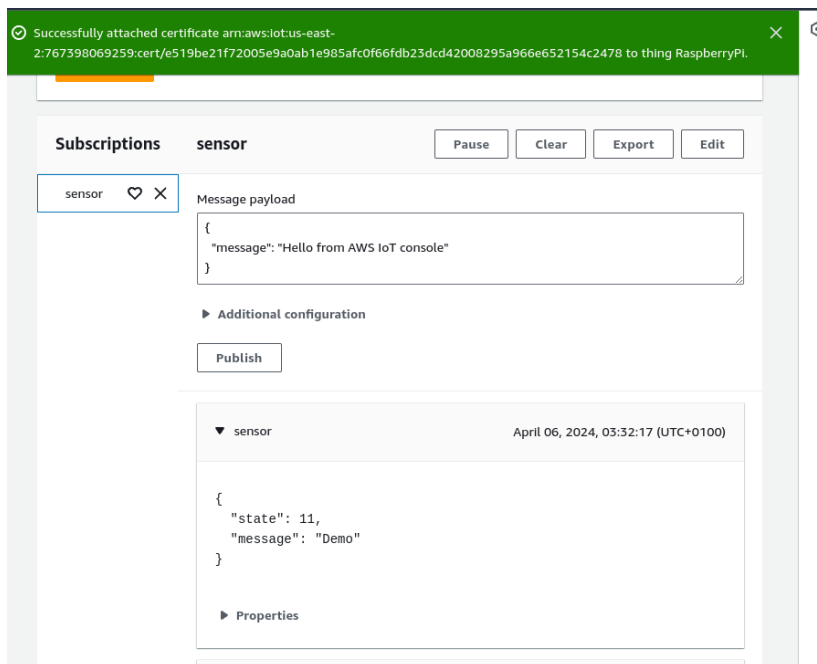
Published topic sensor: {"state": 9, "message": "Demo"}

Published topic sensor: {"state": 10, "message": "Demo"}

Published topic sensor: {"state": 11, "message": "Demo"}


```

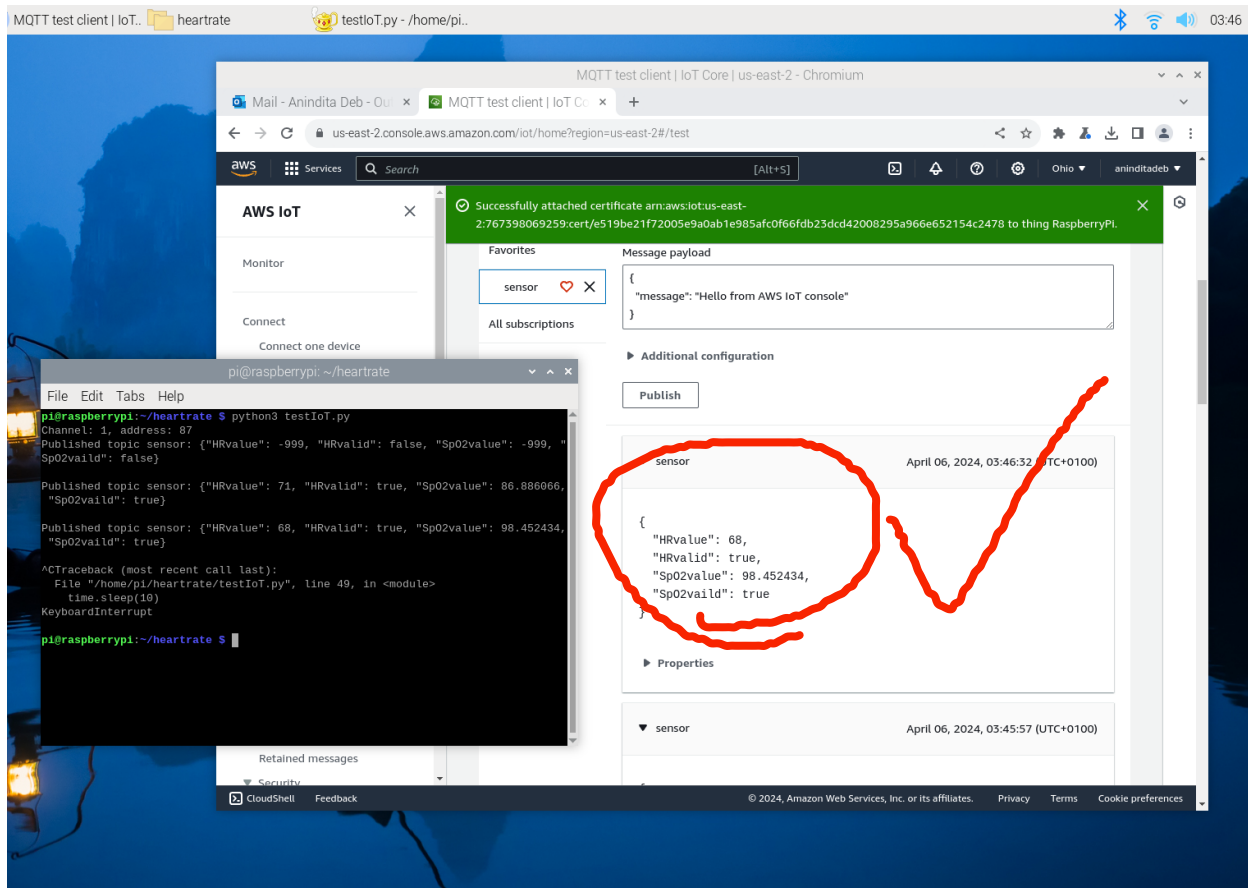
CPE4040 Lab Report



19. The next step is to combine testHR.py and testIoT.py to read heart rate data and SpO2 data from the sensor and send them to AWS IoT. Add the following three lines at the top of testIoT.py to import related packages:
20. In the while loop, copy the codes from testHR.py that read raw sensor data and calculate the heart rate and SpO2 values. Modify the JSON string to have four key-value pairs: {"HRvalue": hr, "HRvalid": hr_valid, "SpO2value": spo2, "SpO2valid": spo2_valid}.



21. Run the updated code and verify if the messages are displayed correctly in both the Raspberry Pi terminal and the AWS IoT Test interface.



22. The next step is to create a database table for storing the sensor data. We will first need to create a rule for passing sensor data to the DynamoDB database service. To do so, you will click on "Manage" followed by "Message routing" and then "Rules" in the AWS IoT Core page. Choose Create rule to start creating your new rule. We will outline the procedure in the next few steps.

23. In Rule properties, enter a new rule name then choose Next to continue.

24. In SQL Statement edit box, enter the following query:

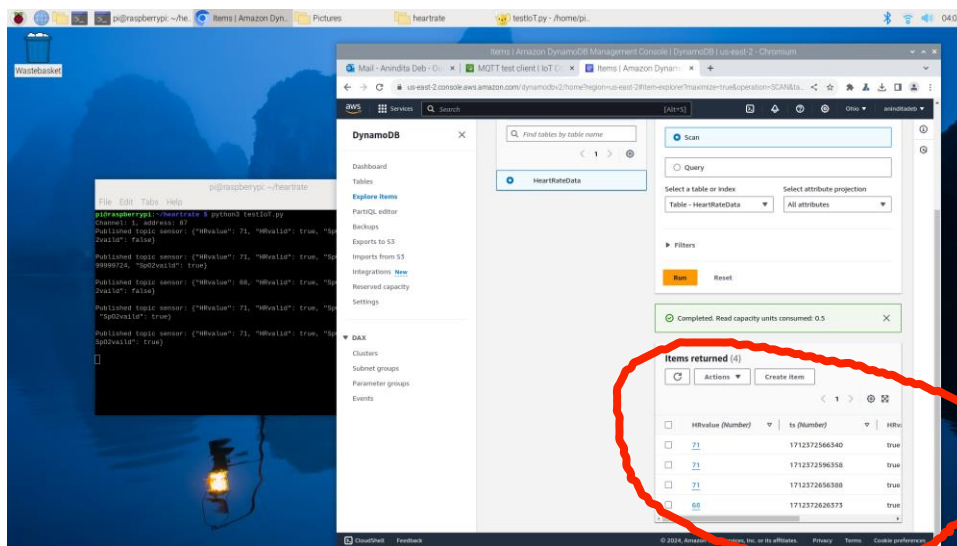
This statement will:

- Listen for MQTT messages with a topic that matches the sensor topic filter.
- Pass all sensor data to DynamoDB
- Format the timestamp attribute

25. In Rule actions, here are the steps to follow:

- Open a list of rules in Action 1 then choose DynamoDBv2 (Split message into multiple columns of a DynamoDB table).
- Click on Create DynamoDB table and this will open a new browser window in the DynamoDB console. Do not close the previous page, as you will need to return to it after completing the configuration in DynamoDB.
- In Table details, enter a new Table name. In Partition key, enter "HRvalue" and in Sort key, enter "ts". Both of them should be selected as "Number" data type. Finally, click on Create table and you will see the new table name appear on the DynamoDB main page.

- Go back to the Rule actions page that you left in the previous step and click the refresh icon next to Table name. Select the table that was created in the previous step.
 - In IAM role, choose Create new role. Provide a name and click on the Create button. You will choose this role in IAM role.
 - Finally, in Review and create page, choose Create and you will see the new rule has been created.
26. Go back to the “MQTT test client” page that was last visited in Step 18 and subscribe to the topic sensor. Next, run the testIoT.py script on the Raspberry Pi and verify that the messages are received on the MQTT subscribe page.
 27. To view the data stored in the table, go to the DynamoDB Console and click on “Tables” in the menu pane. Then choose the table that you just created and click on Explore table items. If you don't see the latest data, you may need to click on the refresh icon to update the values in the table. This will allow you to view all the sensor data in tabular format.
 28. Once you see the correct data on the table, it means the overall system is working properly.



29. For the final task, you will need to delete the existing data in the table and then begin a new data collection run to record your own heart rate and SpO2 readings for 15 minutes. Once you have collected the data, select all entries in the table, click on the Actions button, and export the table as a CSV file.
30. Open the CSV file in Jupyter Notebook and use it to create a line chart for heart rate and another line chart for SpO2. Finally, include the CSV file, the Jupyter Notebook file, and these charts in your report submission.

Conclusion Missing screenshots from cleaning dataset & final line charts

-2

In conclusion, this lab taught us a lot about using heart rate sensors and connecting them to AWS IoT with a Raspberry Pi. During the lab, we faced some challenges. First,

our Raspberry Pi stopped working, so we had to borrow another one from the ECE department. Then, we had trouble transferring the AWS files we needed onto the new Raspberry Pi and we fixed that by changing the testIoT code instead to the updated names. The names of the individual files were not allowing any rename changes before download so we opted to change the code instead for a more clean and smooth experience. The heart rate sensor provided readings at a slower pace than expected, impacting the efficiency and quantity of data collection. We also learned a lot about applying simple data cleaning methods to the raw heart rate sensor data that we retrieved.

Remember for the procedures section that you should list the steps in your own words, and with adequate screenshots that depict the steps you take