

CPE 4040: Data Collection and Analysis, Spring 2023

# **Laboratory Report #6**

## **Lab 6: Heart Rate Sensor Application with AWS IoT**

Team Members: Neal Jarzen & Clarence Barron

Electrical and Computer Engineering

Kennesaw State University

Faculty: Dr. Jeffrey L Yiin

Date of Lab Experiment: January 23, 2023

## **I. Objective**

In this lab we must learn how to interface and read data from heart rate sensors. We also will learn data cleaning techniques for raw heart rate sensor data. We will then develop an AWS IoT application. Then store our sensor data using DynamoDB on the AWS application.

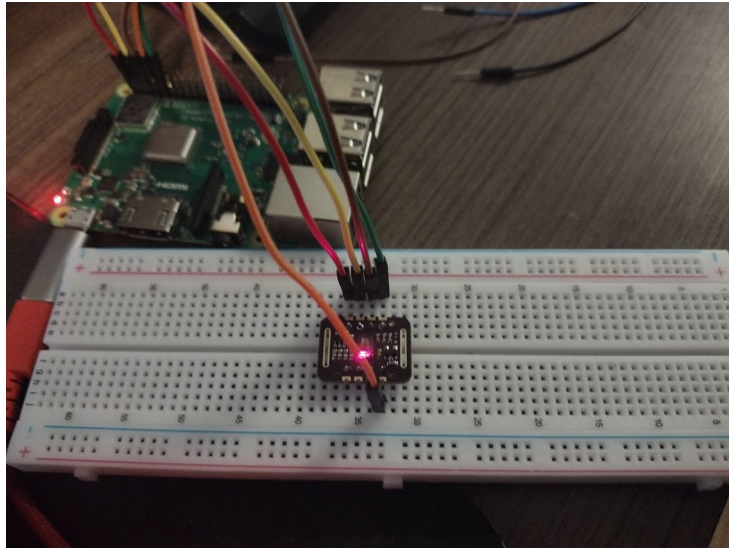
## **II. Material List**

1. Raspberry Pi 3 or 4
2. Power supply adapter
3. Micro SD card (16+GB)
4. Ethernet cable
5. (optional) USB Keyboard, mouse and HDMI monitor or TV
6. Install Putty, Advanced IP Scanner and WinSC
7. Heart rate sensor (MAX 30102) - **Pin headers need to be soldered to the sensor**
8. Python scripts (available online)

## **III. Lab Procedures and Results**

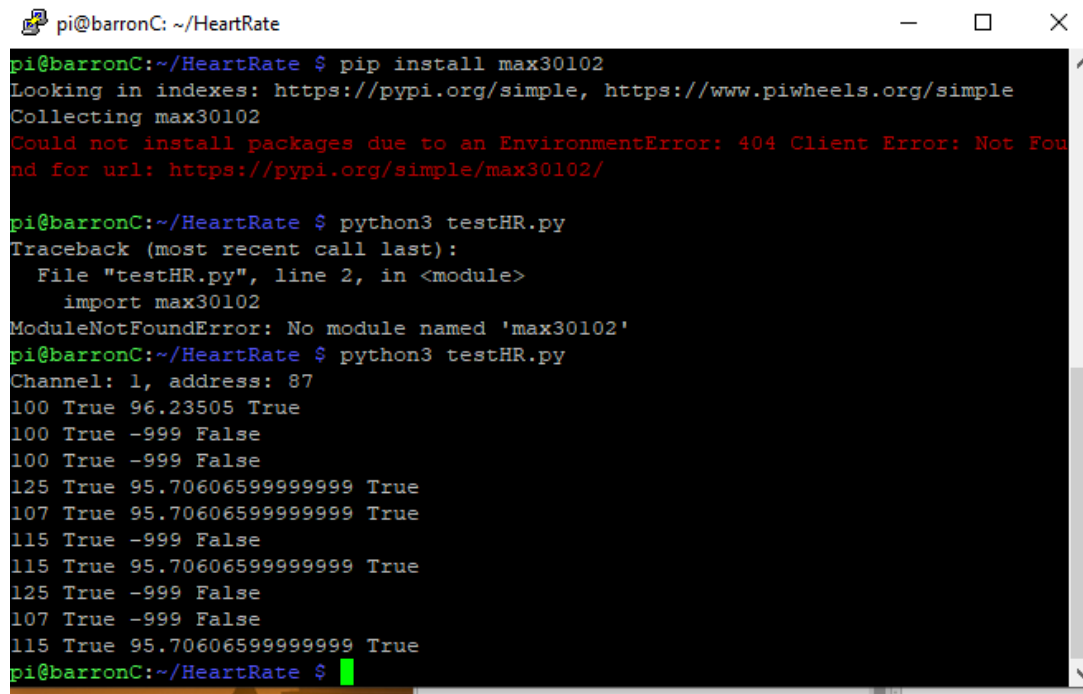
1. Plug in and connect to your Raspberry Pi. When the Pi is on and connected, assemble the heart rate sensor onto the Pi. When connecting the heart rate sensor, the 3.3V Pin (Pin 1) goes into VIN, GPIO2 (Pin3) goes into SDA, GPIO3 (Pin5) into SCL, GPIO4 (Pin 7)

goes into INT, and GND (Pin 9) goes into GND on the heart rate sensor. Make sure you have enabled I2C on the Raspberry Pi.



2. Create a folder in the Raspberry Pi for the Heart Rate sensor and download the following Python code:
  - a. **max30102.py**
  - b. **hrcalc.py**
  - c. **testHR.py**

After downloading the following scripts open the **testHR.py** file. Within it, modify the for loop inside to print the data 10 times and test it with the heart rate sensor. The values should display the heart rate and SpO2 values. SpO2 is the measurement of how much oxygen your blood is carrying as a percentage of the maximum it could carry. This value should not be below 80%. If it is below this, you need to go seek a doctor.



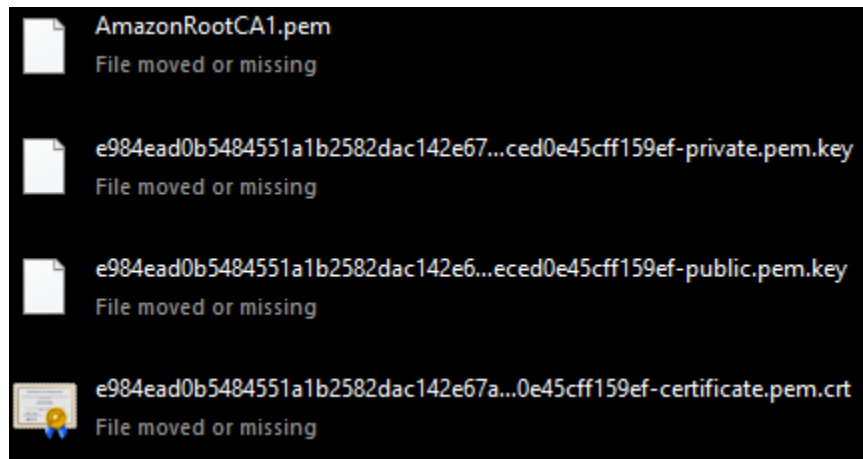
```

pi@barronC: ~/HeartRate
pi@barronC:~/HeartRate $ pip install max30102
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting max30102
Could not install packages due to an EnvironmentError: 404 Client Error: Not Found
for url: https://pypi.org/simple/max30102/

pi@barronC:~/HeartRate $ python3 testHR.py
Traceback (most recent call last):
  File "testHR.py", line 2, in <module>
    import max30102
ModuleNotFoundError: No module named 'max30102'
pi@barronC:~/HeartRate $ python3 testHR.py
Channel: 1, address: 87
100 True 96.23505 True
100 True -999 False
100 True -999 False
125 True 95.70606599999999 True
107 True 95.70606599999999 True
115 True -999 False
115 True 95.70606599999999 True
125 True -999 False
107 True -999 False
115 True 95.70606599999999 True
pi@barronC:~/HeartRate $

```

3. To create the database for this lab, we will use the AWS cloud network provided by Amazon. Create a free account and open “IoT Core” under “Services”. We need to create a thing (which is an IoT Device), we need to first register the device. Afterward, we need to create and save the device certificate, public key, and private key. Also, we need to download the Root CA. Once, we have the correct certificates, we can attach a policy to the certificate.
4. In the AWS IoT core page, go to “Manage” > “All devices” > “Things” and choose “Create Things”. Once on the page, click “Create a Single Thing”. Choose a name for your heart rate sensor then hit “Next”.
5. You will be brought to a page that will allow you to create/configure device certificates. Click “Next”, so AWS will automatically create the certificates for you. You will ask to create/apply a policy. At this step, this can be skipped. However, **this is important. You will be prompted to download the certificate and keys. Download the Device Certificate, the public key, the private key, and the Amazon Root CA 1. Then hit done. They will look like the following when downloaded onto the device.**



- Next, we will create a policy for our device. From the IoT Core page on the side menu panel go to “Security” > “Policies” and click “Create policy” to create a new policy for the device. Give the policy a name and in the “Policy Document” field enter \* (which is the symbol for Wildcard) for both “Policy Action” and “Policy Resource”. Once entered, click “Allow” for “Policy Effect” then click “Create”

The AWS IoT Core Policy allows us to control access to the AWS IoT Core data plane. The purpose for the policy is to connect to the AWS IoT Core message broker, send and receive MQTT messages, and get or update a thing's Device Shadow, which will be needed for us to receive and store the heart sensor data.

HRM\_32Policy [info](#) Edit active version Delete

---

**Details**

Policy ARN arn:aws:iot-us-east-2:525751818487:policy/HRM_32Policy	Active version 1	Created March 20, 2023, 09:51:46 (UTC-0400)	Last updated March 20, 2023, 09:51:46 (UTC-0400)
--	---------------------	--	---

---

[Versions](#) [Targets](#) [Noncompliance](#) [Tags](#)

---

**Active version: 1** [info](#) Builder JSON

Policy effect	Policy action	Policy resource
Allow	*	*

---

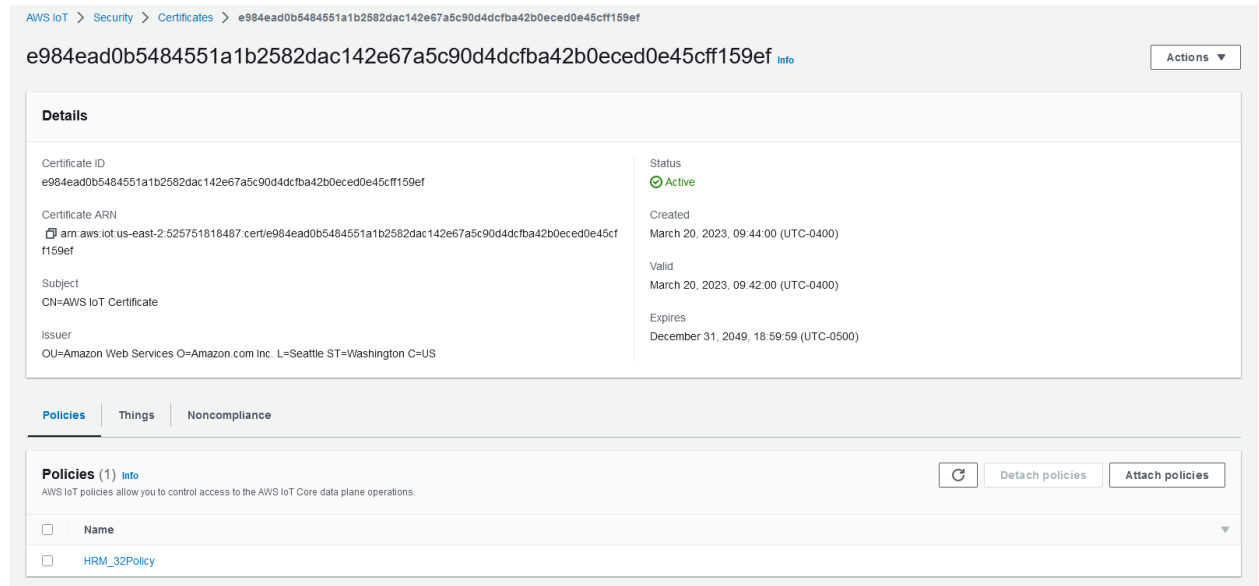
**All versions (1)** [info](#)  
The active and previous versions of this policy. Only one version can be active. A policy can have no more than 5 versions. To update a policy with 5 versions, you must first delete one.

Refresh Delete Set as active Edit version View JSON

<input type="checkbox"/>	Version number	Status	Created
<input type="checkbox"/>	1	Active	March 20, 2023, 09:51:46 (UTC-0400)

- After creating the policy, attach the policy by going to “Security” > “Certificates” on the menu panel and navigate to the certificate you made in the previous step. Afterward,

click “Attach Policy” in the “Actions” menu and select the policy that you just made, then click “Attach Polices”.



8. In the Raspberry Pi, go the heart rate file that you made in Step 2 and create a subfolder called “cert”. Before you transfer the certificates, rename the certificates to the following:

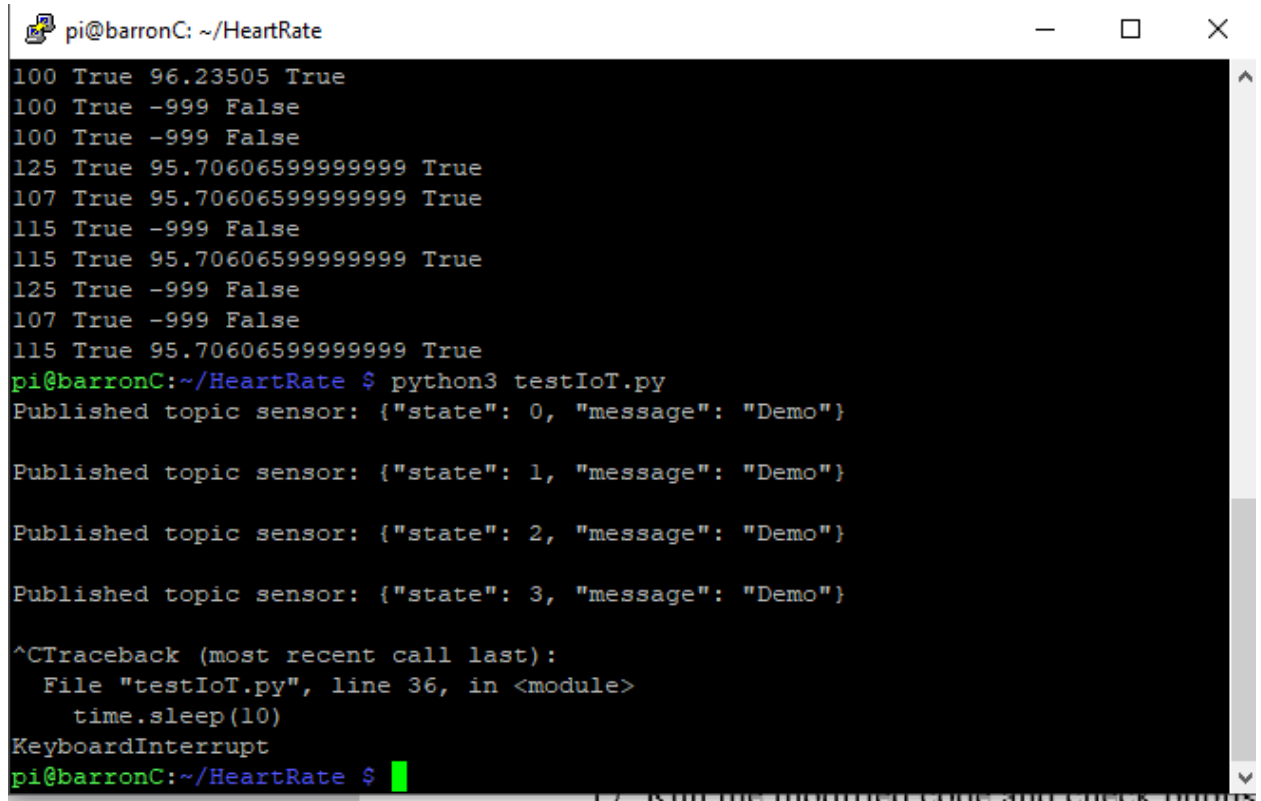
- Device certificate: RaspberryPi-cert.pem
- Private key: RaspberryPi-private.pem.key
- Root CA: RootCA1.pem

After renaming the certificates, transfer them into the newly created “cert” folder in the Pi.

9. Next, download testIoT.py and place it the heart rate folder. Then, install the AWS Python SDK by going to the terminal and entering **pip3 install AWSIoTPythonSDK**. Once the installation is finished, open testIoT.py and modify the parameters.

- a. For the host URL, go back to the AWS IoT Core page and in the menu pane click “Settings” and copy the end point.
- b. For the certPath, put the app folder path of the cert file.
- c. For IoT device, this is the name of your device
- d. The topic will stay as sensor.
- e. The file names for the certificates that were downloaded in the previous step does not have to be changed.

10. Run the modified code on the Pi, then on AWS IoT Core, go to “MQTT test client” on the side panel, and on “Subscribe to a topic”, type in “sensor” and click subscribe. On the Pi, you should see demo messages to display that the connection is working.



```

pi@barronC: ~/HeartRate
100 True 96.23505 True
100 True -999 False
100 True -999 False
125 True 95.70606599999999 True
107 True 95.70606599999999 True
115 True -999 False
115 True 95.70606599999999 True
125 True -999 False
107 True -999 False
115 True 95.70606599999999 True
pi@barronC:~/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"}

^CTraceback (most recent call last):
  File "testIoT.py", line 36, in <module>
    time.sleep(10)
KeyboardInterrupt
pi@barronC:~/HeartRate $

```

11. Once the connection has been verified, we will now combine the **testHR.py** and **testIoT.py** files. At the top of the **testIoT.py** file, type the following:

```

import max30102
import hrcalc

m = max30102.MAX30102()

```

After putting the lines of code at the top, go to the **while** loop in the same folder and put in the code that reads the raw sensor data and calculates both the heart rate and SpO2 values, which are the following:

```

red, ir = m.read_sequential()
hr, hr_valid, spo2, spo2_valid = hrcalc.calc_hr_and_spo2(ir[:100], red[:100])

```

Also in the messageJson line, inside the parenthesis of the json.dumps put in:  
`{"HRvalue": hr, "HRvalid": hr_valid, "SpO2value": spo2, "SpO2valid": spo2_valid}}`

This will deliver the sensor data to the database within the AWS IoT Core. Afterward, run the code to verify that the changes are working by seeing if the Raspberry Pi and the IoT Test interface is displaying the heart rate correctly.

12. Next, we will create the database table. On the side panel, go to "Message routing" > "Rules" and click "Create Rule". Once on the page, give the rule a name and hit next. On the SQL statement, in the box type:

```
SELECT *, timestamp() AS ts FROM 'sensor'
```

This will help us get the data from the sensor and pass the data to DynamoDB along with the timestamps.

13. In the rules actions on the next page, under "Action1" and on the drop down choose "DynamoDBv2". Click "Create DynamoDB Table", and it will bring you to another page. **Do not close the previous page, as we will be going back to it soon.** Under "Table Details" enter a table name of your choosing. In the "Partition Key" type "HRvalue" and in "Sort key" put in "ts". Select both keys as number values.
14. Go back to the "Rules action" on the previous page, and click the refresh icon next to "Table name" and select the newly created table. Next, in the "IAM role", choose "Create new role" and give it a name of your choosing then click "Create", then choose the created "IAM role". Under the "Review and Create" page, click create and the new role will be created.
15. Go back to "MQTT test client" and subscribe to the "sensor" topic. On the Pi, run the modified testIoT.py code to verify if the sensor data is being passed to the database. To verify this, go the DynamoDB Console and click "Tables". Choose the table you made and click "Explore table items". This will display all the recent data that has been collected. If you do not see the data, try clicking the refresh button. Once you see data, this means that the system is working as intended. Afterward, delete the data.



## CPE4040 Lab Report

```

{
  "HRvalue": 136,
  "HRvalid": true,
  "SpO2value": 99.8058,
  "SpO2valid": true
}

```

► Properties

▼ sensor March 31, 2023, 16:51:43 (UTC-0400)

```

{
  "HRvalue": 136,
  "HRvalid": true,
  "SpO2value": 98.500104,
  "SpO2valid": true
}

```

► Properties

▼ sensor March 31, 2023, 16:51:29 (UTC-0400)

```

{
  "HRvalue": 125,
  "HRvalid": true,
  "SpO2value": 99.43662599999999,
  "SpO2valid": true
}

```

► Properties

16. Next, for 15 minutes, begin to collect data from the heart rate sensor to record the heart rate and SpO2 readings. Afterward, select all the entries, in the table, and export them as a CSV file. Then proceed to Jupyter Notebook and clean the data and analyze the data.

Items returned (14)

Actions

Create item

<

1

>

<input type="checkbox"/>	HRvalue	ts	HRvalid	SpO2valid	SpO2value	
<input type="checkbox"/>	65	1680295861515	true	true	99.0534	
<input type="checkbox"/>	166	1680295804789	true	true	48.602999999999994	
<input type="checkbox"/>	100	1680295974974	true	true	99.657	
<input type="checkbox"/>	150	1680295875646	true	true	99.848136	
<input type="checkbox"/>	150	1680295960741	true	true	99.831474	
<input type="checkbox"/>	136	1680295904011	true	true	98.500104	
<input type="checkbox"/>	136	1680295918143	true	true	99.8058	
<input type="checkbox"/>	88	1680295847385	true	true	97.802616	
<input type="checkbox"/>	88	1680295946610	true	true	99.727416	
<input type="checkbox"/>	125	1680295889777	true	true	99.43662599999999	
<input type="checkbox"/>	125	1680295989209	true	true	99.854424	
<input type="checkbox"/>	107	1680295932378	true	true	99.462504	
<input type="checkbox"/>	68	1680295833150	true	true	94.129194	
<input type="checkbox"/>	83	1680295819018	true	true	95.70606599999999	

#### **IV. Conclusion**

Overall, this lab was not bad. However, it was interesting to see how much the heart can jump around knowing you are measuring it. The most notable thing we have learned was how to solder the pins into the board. It was a very nice experience to know, and, considering how important it is to know how to solder something, we feel like this should have been taught earlier in the major, but that is more feedback in regards to the department rather than the class itself. However, it was still a nice experience. Also, using the AWS system was interesting as well, as this can help with other projects and ideas in terms of data tracking.