

CPE 4040 LABORATORY EXPERIMENT #7

Tanim Choudhury

Johnny Lozano

April 2022

Dr. Yiin

Kennesaw State  
University

**Title:**

**Objective:**

1. Learn how to interface and read data from a heart rate sensor.
2. Understand how to create an AWS IoT application with Raspberry Pi.
3. Understand how to use DynamoDB app to store sensor data using AWS.

**Materials List:**

1. Raspberry Pi 3 or 4 w/ Raspian OS
2. Power supply adapter
3. Micro SD card (16+ GB)
4. USB keyboard, mouse, and HDMI monitor
5. Heart Rate Sensor (Max 30102)
6. Software:
  - Putty
  - Advanced IP Scanner
  - WinSCP

## Procedures & Results:

### I. Setting Up the Heart Rate Sensor on the Raspberry Pi

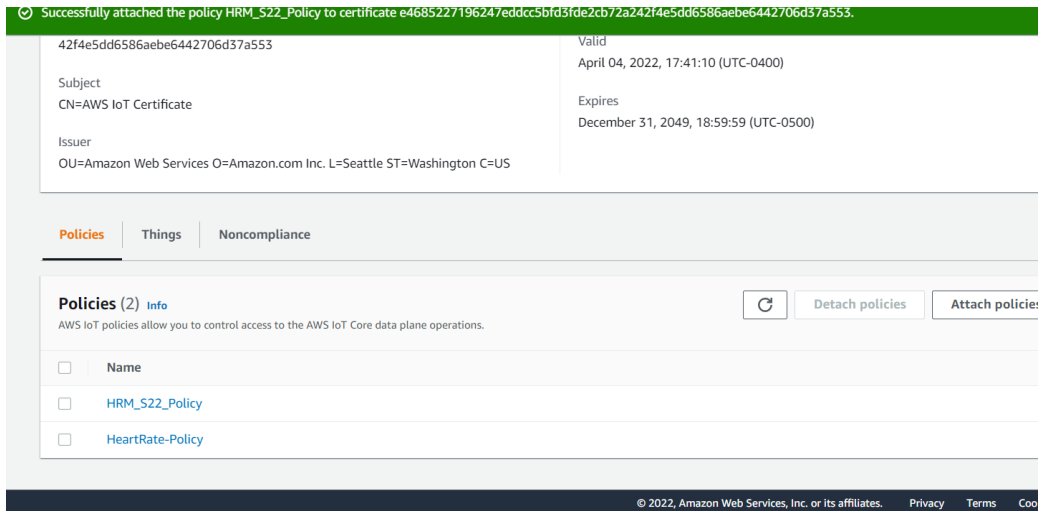
1. First the Raspberry Pi was powered on, and a remote SSH connection was established with a laptop.
2. Next, the heart rate sensor (Max 30102) was connected to the Pi through GPIO pins:
  - HR Sensor vin was connected to pin 1 for 3.3 voltage source
  - HR Sensor SDA was connected to GPIO 2 (pin 3)
  - HR Sensor SCL was connected to GPIO 3 (pin 5)
  - HR Sensor INT was connected to GPIO 4 (pin 7)
  - HR Sensor GND was connected to GND (pin 9)
3. The I2C interface on the RPi was enabled.
4. Within the RPi, a folder called heart\_rate was created and three Python scripts were added:
  - max30102.py (library for heart rate sensor)
  - hrcalc.py (calculations for heart rate and SpO2)
  - testHR.py (used for testing data collection)
5. A test was using testHR.py to ensure that the heart rate sensor was connected properly to the Pi.

#### Question: What is SpO2 and how does the MAX30102 sensor detect it?

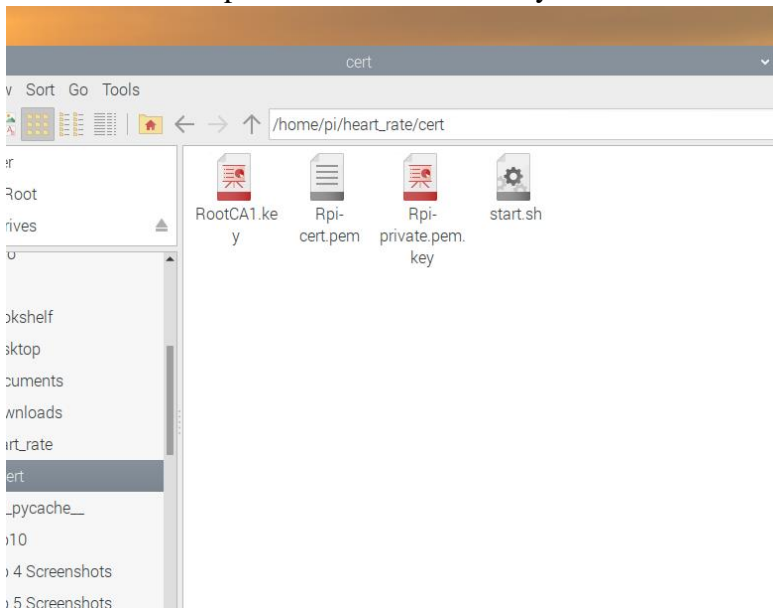
SpO2 is known as pulse oximetry or **oxygen saturation** and it is the ratio of blood cells that carry oxygen to blood cells that do not carry oxygen. The MAX30102 detects SpO2 by using two LEDs; one shines light into the skin, and the other detects the amount of light that shines back, and based on this, oxygen levels can be measured.

### II. Creating an AWS IoT “Thing”

6. An AWS account was created and IoT Core was used at <https://www.aws.amazon.com>
7. A thing (or IoT device) was created
  - The RPi was registered
  - A device certificate, public key, private key, and root CA1 were created
  - A policy was created and attached to the thing/certificate
8. Under the “Manage” tab in AWS IoT Core, “Create Things” was used in order to name the heart rate sensor.
9. AWS automatically generated certificates for the device.

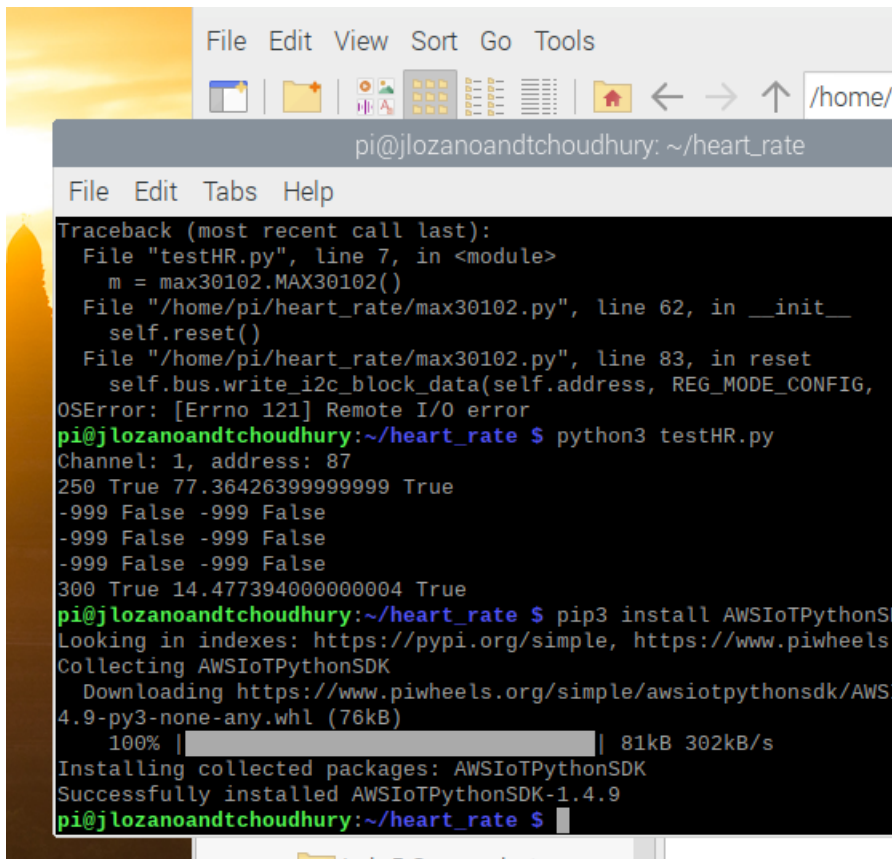


10. All the keys/certificates were downloaded to the laptop.
11. Under “Create policy”, a policy was created for the device. It was set to “Allow”, “\*”, “\*” for open access with the RPi.  
**Question: What is AWS IoT Core policy?**  
 AWS IoT Core policy allows control over which devices or operations can connect to the AWS IoT message broker.
12. The policy was connected to the certificates (under “Secure”).
13. A folder called “cert” was created within the heart\_rate directory in the RPi, and the certificates were placed in the cert directory.



### III. Connecting the Heart Rate Sensor to AWS IoT

14. The python scripts (from step 4) were downloaded to the heart\_rate directory.
15. AWS Python SDK was installed from the RPi terminal with `pip3 install AWSIoTPythonSDK` in order to establish a connection between the RPi and AWS IoT.

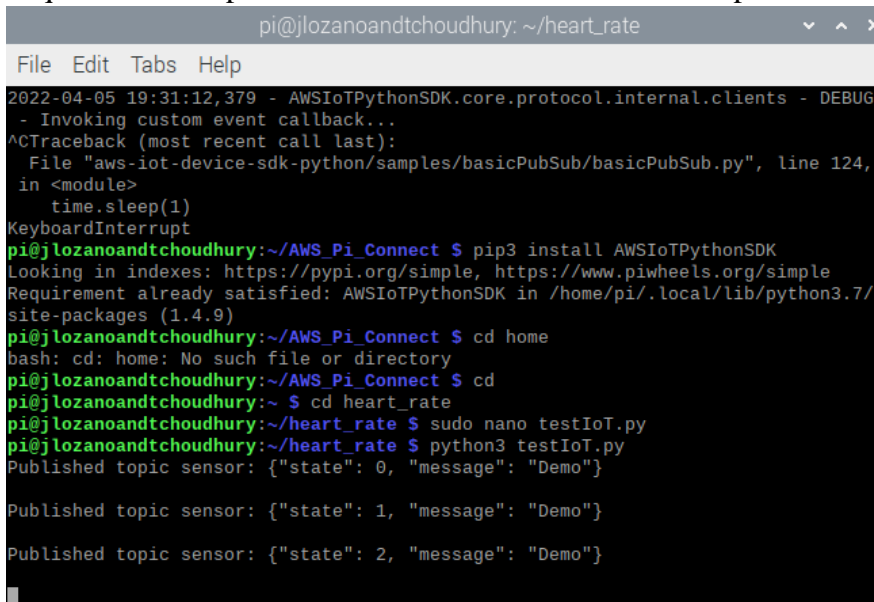


```

File Edit View Sort Go Tools
pi@jlozanoandtchoudhury: ~/heart_rate
File Edit Tabs Help
Traceback (most recent call last):
  File "testHR.py", line 7, in <module>
    m = max30102.MAX30102()
  File "/home/pi/heart_rate/max30102.py", line 62, in __init__
    self.reset()
  File "/home/pi/heart_rate/max30102.py", line 83, in reset
    self.bus.write_i2c_block_data(self.address, REG_MODE_CONFIG,
OSError: [Errno 121] Remote I/O error
pi@jlozanoandtchoudhury:~/heart_rate $ python3 testHR.py
Channel: 1, address: 87
250 True 77.36426399999999 True
-999 False -999 False
-999 False -999 False
-999 False -999 False
300 True 14.477394000000004 True
pi@jlozanoandtchoudhury:~/heart_rate $ pip3 install AWSIoTPythonSDK
Looking in indexes: https://pypi.org/simple, https://www.piwheels
Collecting AWSIoTPythonSDK
  Downloading https://www.piwheels.org/simple/awsiotpythonsdk/AWSIoT
4.9-py3-none-any.whl (76kB)
    100% | 81kB 302kB/s
Installing collected packages: AWSIoTPythonSDK
Successfully installed AWSIoTPythonSDK-1.4.9
pi@jlozanoandtchoudhury:~/heart_rate $

```

16. The testIoT.py were modified with the RPi credentials and certificates.
17. A quick test was performed in order to show that the script was functioning on the RPi.



```

pi@jlozanoandtchoudhury: ~/heart_rate
File Edit Tabs Help
2022-04-05 19:31:12,379 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG
- Invoking custom event callback...
^CTraceback (most recent call last):
  File "aws-iot-device-sdk-python/samples/basicPubSub/basicPubSub.py", line 124,
in <module>
    time.sleep(1)
KeyboardInterrupt
pi@jlozanoandtchoudhury:~/AWS_Pi_Connect $ pip3 install AWSIoTPythonSDK
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: AWSIoTPythonSDK in /home/pi/.local/lib/python3.7/
site-packages (1.4.9)
pi@jlozanoandtchoudhury:~/AWS_Pi_Connect $ cd home
bash: cd: home: No such file or directory
pi@jlozanoandtchoudhury:~/AWS_Pi_Connect $ cd
pi@jlozanoandtchoudhury:~ $ cd heart_rate
pi@jlozanoandtchoudhury:~/heart_rate $ sudo nano testIoT.py
pi@jlozanoandtchoudhury:~/heart_rate $ python3 testIoT.py
Published topic sensor: {"state": 0, "message": "Demo"}

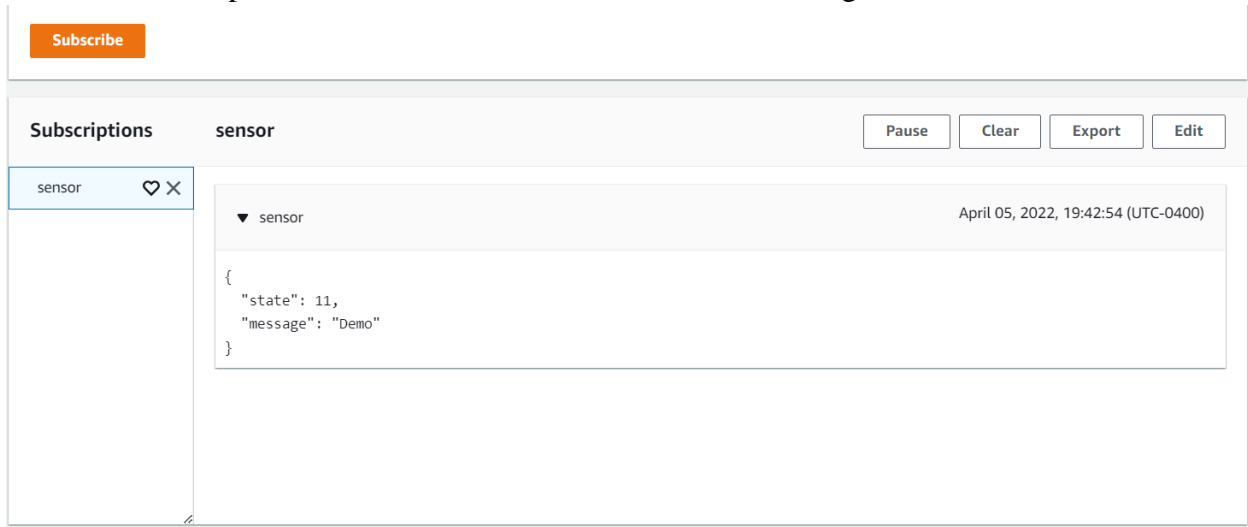
Published topic sensor: {"state": 1, "message": "Demo"}

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

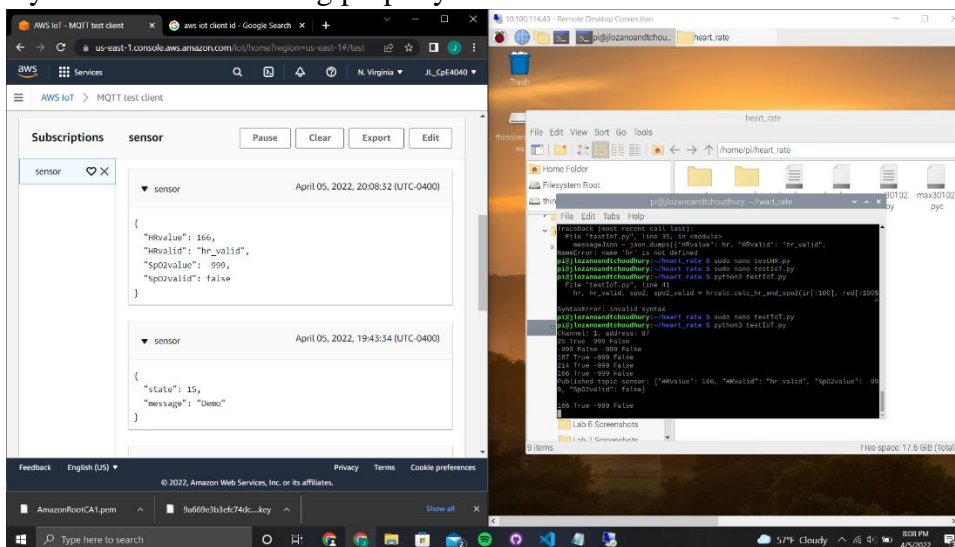
```

18. From the “Test” section in the AWS IoT Core menu page, a MQTT Client service was started, and topic name “sensor” was subscribed.

19. The first three lines from `testHR.py` were added to the `testIoT.py` script to read data and SpO2 data from the sensor.
20. In addition, the while loop was modified to have 4 key-value pairs for heart rate and SpO2.
21. Another test was performed to make sure that the RPi was sending data to the AWS IoT Test.



22. Next, a rule was created in order to pass data to DynamoDB.
23. The table was split into multiple columns, and created.
24. “Create a new resource” was clicked in order to enter DynamoDB. A table was configured for the data.
25. Back in AWS IoT, a new rule was created, and a policy was attached to the rule.
26. “Create Rule” was clicked to finalize the rule.
27. In the “Test” page, sensor topic was subscribed, and another test was performed to verify that DynamoDB was working properly.



28. Lastly, the application was ready for data collection, and the heart rate program was run for 10 minutes while AWS was collecting the data on the individual who kept his finger on the sensor for the duration of the 10 minute timeframe.

The screenshot shows a Raspberry Pi terminal window titled 'heart\_rate' with the following output:

```

Published topic sensor: {"HRvalue": 107, "HRvalid": "hr_valid", "SpO2value": -999, "SpO2valid": false}
100 True 59.487815999999999 True
Published topic sensor: {"HRvalue": 100, "HRvalid": "hr_valid", "SpO2value": 59.487815999999999, "SpO2valid": true}
44 True -999 False
Published topic sensor: {"HRvalue": 44, "HRvalid": "hr_valid", "SpO2value": -999, "SpO2valid": false}
44 True 15.713399999999999 True
Published topic sensor: {"HRvalue": 44, "HRvalid": "hr_valid", "SpO2value": 15.713399999999999, "SpO2valid": true}
88 True 99.831474 True
Published topic sensor: {"HRvalue": 88, "HRvalid": "hr_valid", "SpO2value": 99.831474, "SpO2valid": true}
^CTraceback (most recent call last):
  File "testIoT.py", line 49, in <module>
    time.sleep(10)
KeyboardInterrupt
pi@lozanoandchoudhury:~/heart_rate $

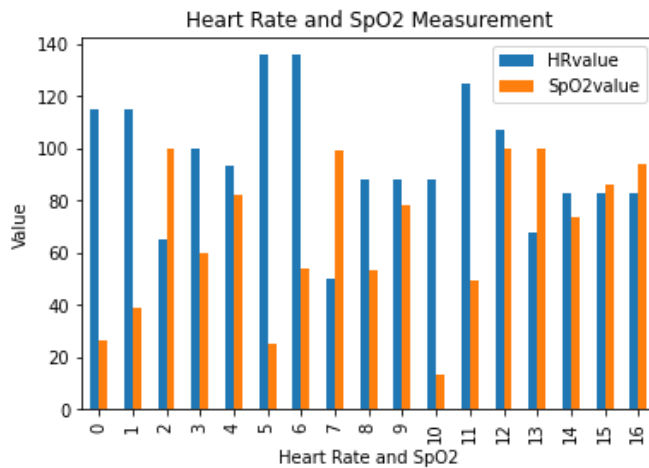
```

The web interface on the right shows a table with the following data:

ts	HRvalid	SpO2valid	SpO2value
164920588...	hr_valid	false	-999
164920591...	hr_valid	false	-999
164920602...	hr_valid	true	93.468594
164920576...	hr_valid	true	26.431913999999999
164920577...	hr_valid	true	38.540999999999999
164920607...	hr_valid	false	-999

#### IV. Data

Data was exported to a .csv file and the analytics are shown below:



	HRvalue	ts	SpO2value
count	17.000000	1.700000e+01	17.000000
mean	95.470588	1.649210e+12	66.606864
std	24.321075	0.000000e+00	29.108266
min	50.000000	1.649210e+12	13.232376
25%	83.000000	1.649210e+12	49.559634
50%	88.000000	1.649210e+12	73.381800
75%	115.000000	1.649210e+12	93.803400
max	136.000000	1.649210e+12	99.855786

**Conclusion:**

The results of our experiment showed very sporadic jumps in heart rate and SpO2 values which makes sense because the subject of the test had consumed large amounts of caffeine prior to the experiment. As far as the experiment goes, the Raspberry Pi setup was very streamlined with no issues with setting up the scripts, sensor, and GPIO connection. We did run into trouble with setting up the AWS IoT during the lab session. Specifically, burning the circuit traces of the PCB with the soldering iron. We needed to wait for the soldering iron to cool down from another group use. Instead of using it immediately. Afterwards, we tried again and got it working properly. From there, the only real obstacle was modifying the testIoT.py script with the data calculation scripts from the testHR.py file. It was a bit confusing how to configure it, so we removed the for loop and had it do the calculation with the while loop time delay instead. In conclusion, all three objectives were demonstrated through configuration, connection, and analysis of the data collected from the heart rate sensor experiment.