KENNESAW STATE UNIVERSITY

CPE 4040: Data Collection and Analysis

Lab 6: Heart Rate Sensor Application with AWS IoT

Learning Objectives:

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.

AWS IoT Resources:

The AWS website provides a wealth of resources for learning to use the platform.

- Getting started with AWS IoT Core

Hardware and Software Requirement:

1. Heart rate sensor (MAX 30102). **Note that you need to solder the pin headers to the sensor before the lab**.
2. Python scripts (available in D2L Assignment)

Lab Procedure:

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:

RPi	HR Sensor
3.3V (pin1)	VIN
I2C_SDA1 (pin3; GPIO 2)	SDA
I2C_SCL1 (pin5; GPIO 3)	SCL
- (pin7; GPIO 4)	INT
GND (pin9)	GND

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:

- `max30102.py`: library for the heart rate sensor;
- `hrcalc.py`: procedures and algorithm to calculate heart rate and SpO2 values;
- `testHR.py`: test data collection.

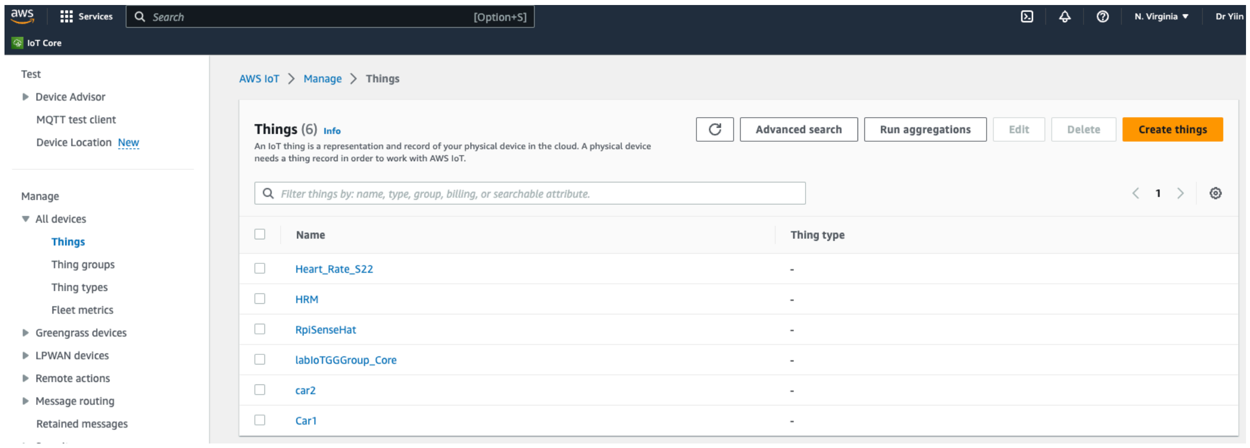
5. Modify the code in `testHR.py` with a ***for*** loop to print the data 10 times. Run the code and place your index finger on the sensor. You should see Heart Rate and SpO2 values displayed on the terminal for each loop iteration. Verify that the data appear valid for all samples, indicating that the heart rate sensor is functioning correctly.

   <u>Question:</u> What is SpO2? How is it measured by the heart rate sensor?

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".
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**.



Things (6) Info

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Name	Thing type
Heart_Rate_S22	-
HRM	-
RpiSenseHat	-
labIoTGGGroup_Core	-
car2	-
Car1	-

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

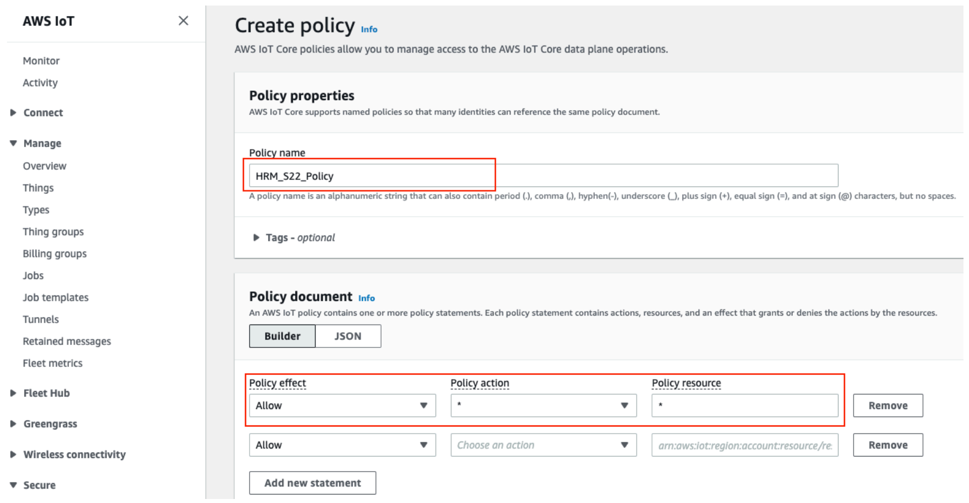
are in your Pi directory where you unzipped the ZIP folder from Step 7 where the device was connected. If a certificate does not appear in this tab, you can click on **Create Certificate**. Check to set the certificate as **Active.**

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.

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?



Create policy

AWS IoT Core policies allow you to manage access to the AWS IoT Core data plane operations.

Policy properties

AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name: HRM_S22_Policy

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen(-), underscore (_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

Policy document

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy effect	Policy action	Policy resource
Allow	*	*
Allow	Choose an action	arn:aws:iot:region:account:resource/re

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:
    - 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`

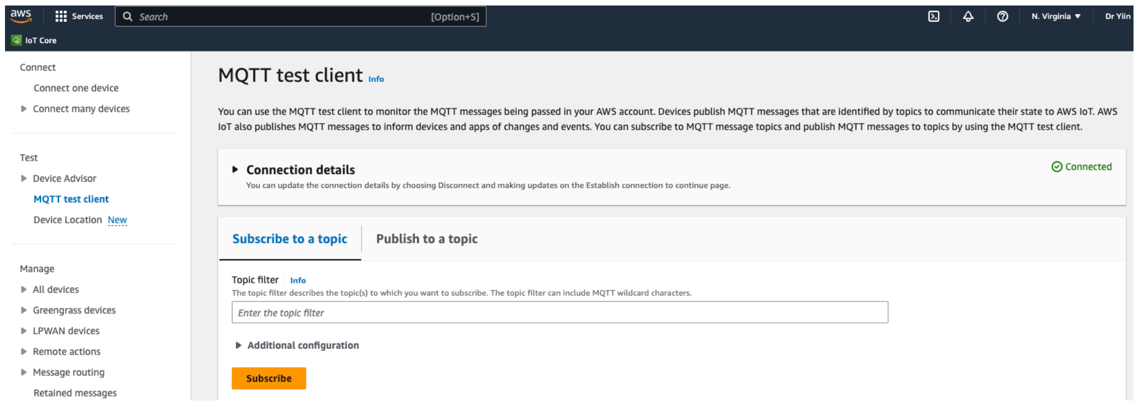
16. Once the installation is complete, open `testIoT.py` and modify the following parameters:

```
host = "aleebx1ktmwaqk-ats.iot.us-east-2.amazonaws.com"
certPath = "/home/pi/AppFolder/cert/"
clientId = "Device_Name"
topic = "sensor"
```

- 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.

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.



MQTT test client Info

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

Connection details — Connected

You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.

Subscribe to a topic | Publish to a topic

Topic filter Info

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

Enter the topic filter

Additional configuration

Subscribe

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:

```
import max30102
import hrcalc

m = max30102.MAX30102()
```

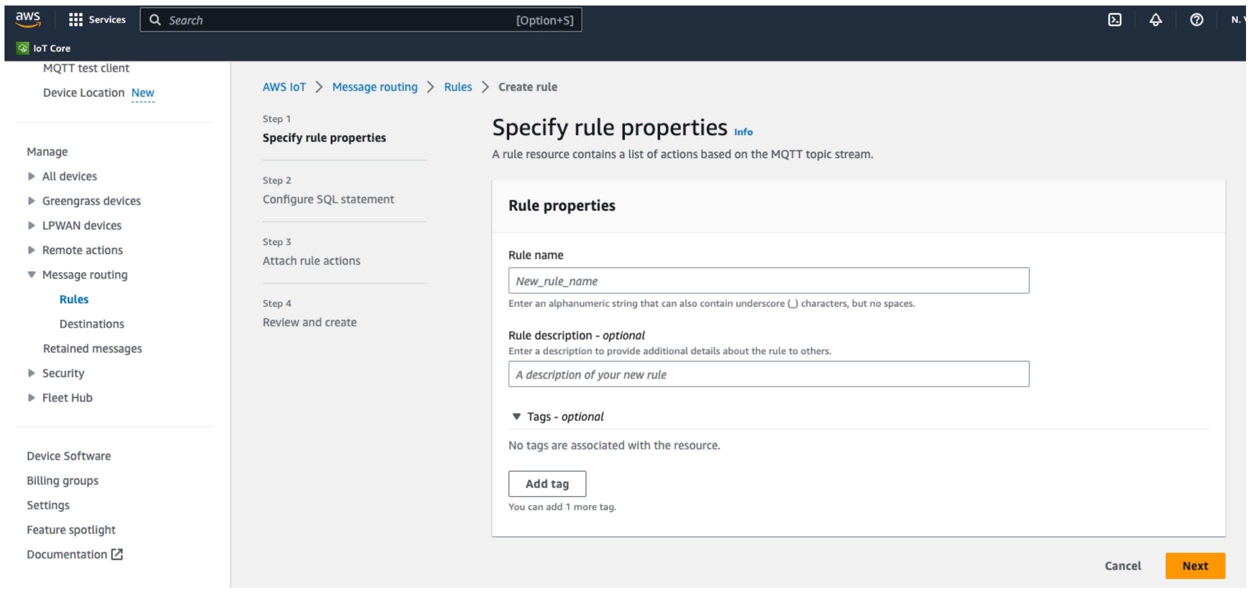
20. In the ***while*** loop, copy the codes from `testHR.py` that read raw sensor data and calculate the heart rate and SpO2 values. Modify the JSON string to have four key-value pairs:

{“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.



Specify rule properties Info

A rule resource contains a list of actions based on the MQTT topic stream.

Rule properties

Rule name

New_rule_name

Enter an alphanumeric string that can also contain underscore (_) characters, but no spaces.

Rule description - *optional*

Enter a description to provide additional details about the rule to others.

A description of your new rule

Tags - *optional*

No tags are associated with the resource.

You can add 1 more tag.

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

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

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

    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.

Lab Report

In the report, you will show screenshots of setup and results during critical steps in:

- Setting up heart rate sensor on Raspberry Pi
- Creating an AWS IoT "thing"
- Connecting the heart rate sensor to AWS IoT

For the heart rate and SpO2 data, use pandas DataFrame methods, such as describe( ), to show the basic statistics of the data set (max, min, count, percentiles, etc.). Use the data clearing techniques we cover in class, that is, identifying invalid data and detecting outliers, before making the plots.

Use the lab report template and submit the report via D2L Assignment, in PDF format.