

SCHOOL OF DIGITAL MEDIA AND INFOCOMM TECHNOLOGY (DMIT)



IOT CA2

Step-by-step Tutorial

DIPLOMA IN BUSINESS INFORMATION TECHNOLOGY
DIPLOMA IN INFORMATION TECHNOLOGY
DIPLOMA IN INFOCOMM SECURITY MANAGEMENT

ST0324 Internet of Things (IOT)

Date of Submission: 20 February 2018

Prepared for: Ms Lin Zhao

Class: DISM/FT/3A/33

Submitted by:

Student ID **Name**

1529377 Law Si Ying, Joy

1529393 Sean Phang

Table of Contents

Section 1 Overview of Project	3
A. Where We Have Uploaded Our Tutorial.....	3
B. Why Have We Chosen to Upload to This Site	3
C. What Have We Uploaded	3
D. What is the Application About?.....	4
E. Summary of the Steps That will be Described	4
F. How does the Final RPI Set-up Looks Like?	5
G. Completed Fritzing Diagram	6
H. How does the Web Application Look Like?	7
Section 2 Hardware Requirements	9
Hardware Checklist	9
Section 3 Install the Necessary Libraries	10
A. Install Flask.....	10
B. Install gevent.....	10
C. Install Python-dev	10
D. Install SPI-Py.....	10
E. Install Boto	10
F. Install AWS CLI	11
G. Install AWS Python Library.....	11
H. Install Paho MQTT.....	11
Section 4 Enable the Necessary Configurations.....	12
A. Enable SPI via raspi-config	12
B. Enable Device Tree in boot.txt.....	13
Section 5 Sign Up for an AWS Educate Account	14
A. Create a New AWS Educate Account.....	14
Section 6 Sign in to AWS.....	19
A. Sign in to AWS.....	19
Section 7 Configure AWS Services.....	21
A. Configure AWS IoT	21
B. Configure AWS Relational Database Service	28
C. Create a Database using AWS RDS	32
D. Configure AWS S3	36
E. Configure AWS Lambda & AWS IoT Rule	39

Section 8 Configure AWS IoT Rules	46
A. Create an AWS Role	46
B. Create an AWS Policy.....	47
Section 9 The Subscriber/Server	48
A. Write the Code for server.py	48
B. Write the Code for index.html.....	53
C. Write the Code for image.html.....	61
D. Write the Code for newUser.html	63
E. Expected Outcome.....	66
Section 10 The Publisher	67
F. Write the Code for pub.py	67
G. Expected Outcome.....	69
H. Final Outcome (Dashboard).....	70
Section 11 Task List	71
A. Task List.....	71

Section 1

Overview of Project

A. Where We Have Uploaded Our Tutorial

https://github.com/joylaw/IOT_CA2

B. Why Have We Chosen to Upload to This Site

The main reason why our group has decided to upload our project onto GitHub is that it makes the process of sharing and editing the code extremely easy for all of us. Additionally, we are very familiar with GitHub as compared to the rest of the sites since we have made use of it in the past.

C. What Have We Uploaded

```
iot_ca2
└── static (bower_components, build, dist, plugins, card.jpg, greenled.jpg, home.jpg,
    redled.jpg, stylesheet.css, yellowled.jpg)
└── templates (image.html, index.html, newUser.html)
└── iot-policy.json
└── iot-role-trust.json
└── MFRC522.py
└── pub.py
└── server.py
resources
└── iot_file.zip
DISMFT3A33_CA2-TutorialDocument_JoyLaw_SeanPhang.pdf
```

Note: bower_components, build, dist & plugins contain other files needed for the web interface

D. What is the Application About?

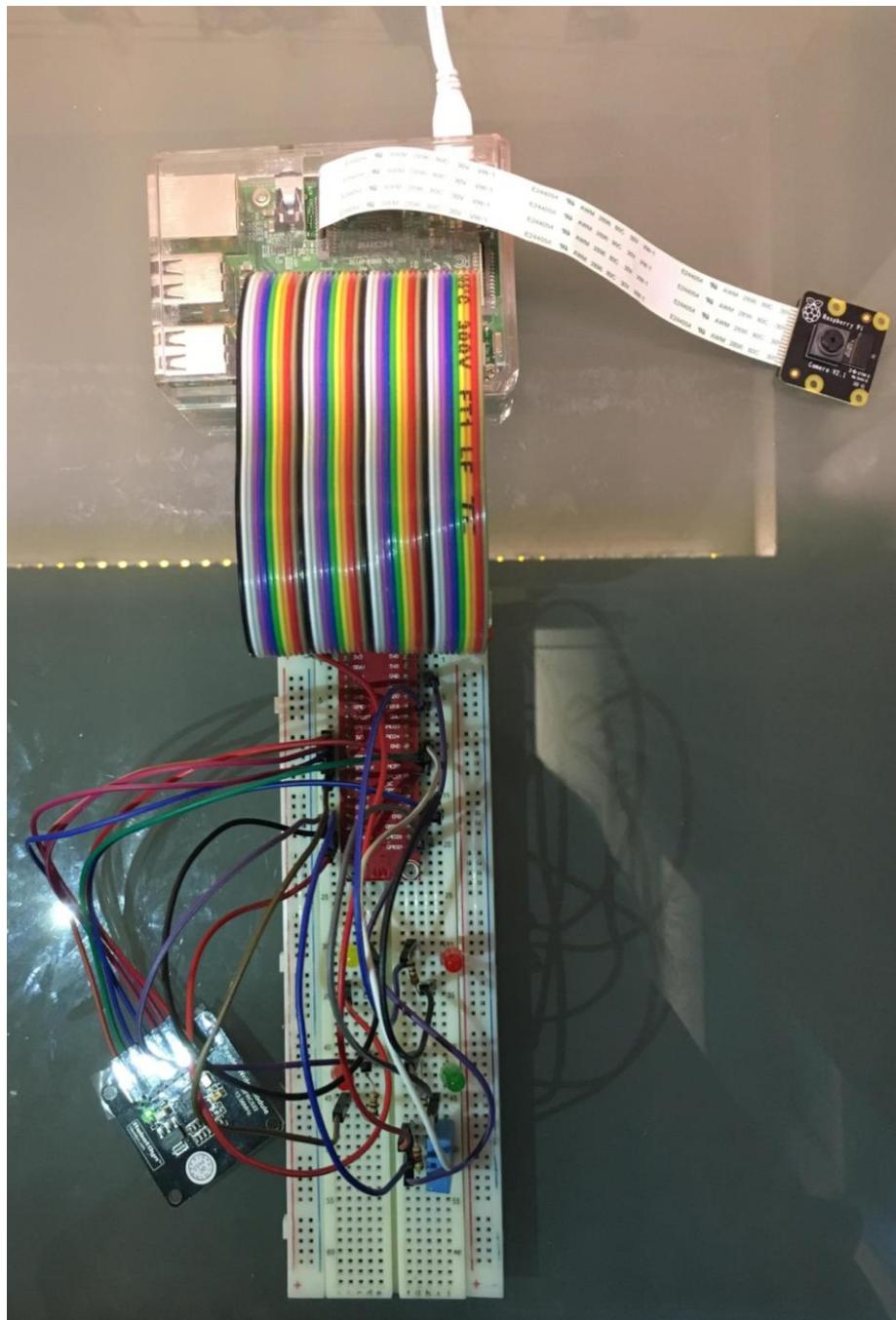
The concept of the application is a smart home security and environment monitoring system, where historical view of people who has entered the home via an image, date & time stamp, as well as live data of the current temperature and humidity of the home can be seen. Additionally, users are also able to control their lights in the house remotely.

E. Summary of the Steps That will be Described

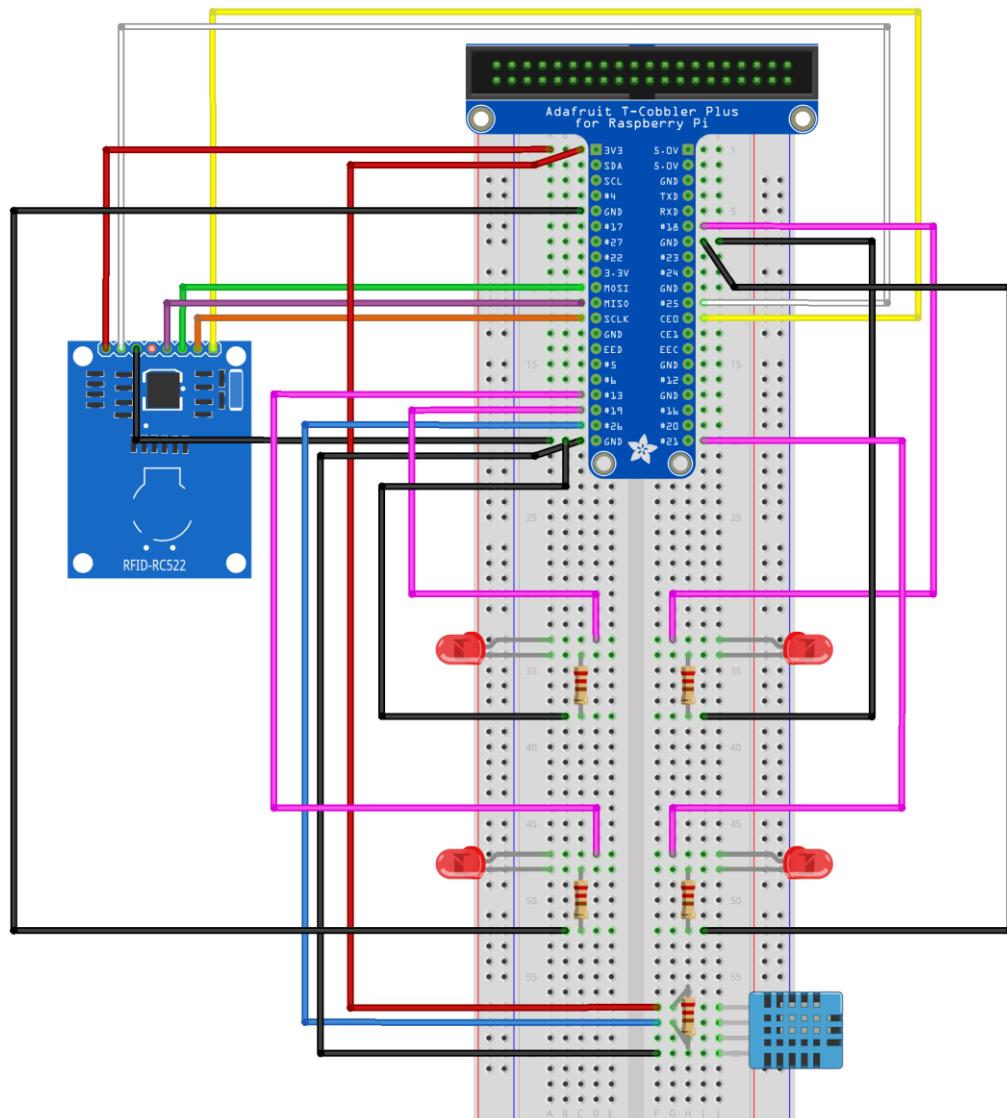
Provide a bullet list of the steps that will be covered in the other parts of this tutorial

Section	Description	
1)	Overview	Overview of this assignment tutorial
Sections 2 to 11 provides the step-by-step instructions to set up the application		
2)	Hardware requirements	Provides a hardware checklist
3)	Install Libraries	Shows the steps needed to installed the required libraries
4)	Enable Configurations	Provides the steps needed to enable certain configurations
5)	Create AWS Account	Steps needed to create an AWS Educate Account
6)	Login to AWS	Logging in to AWS Educate
7)	Configure AWS Services	Provides a step-by-step guide to enable the following AWS Services: AWS IoT, RDS, S3, Lambda
8)	Configure AWS IoT Rules	Creating and assigning role and policies
9)	The Subscriber	The code needed to run the server/subscriber
10)	The Publisher	The code needed to run the publisher
11)	The Web Interface Design	The code needed to display onto a dashboard

F. How does the Final RPI Set-up Looks Like?

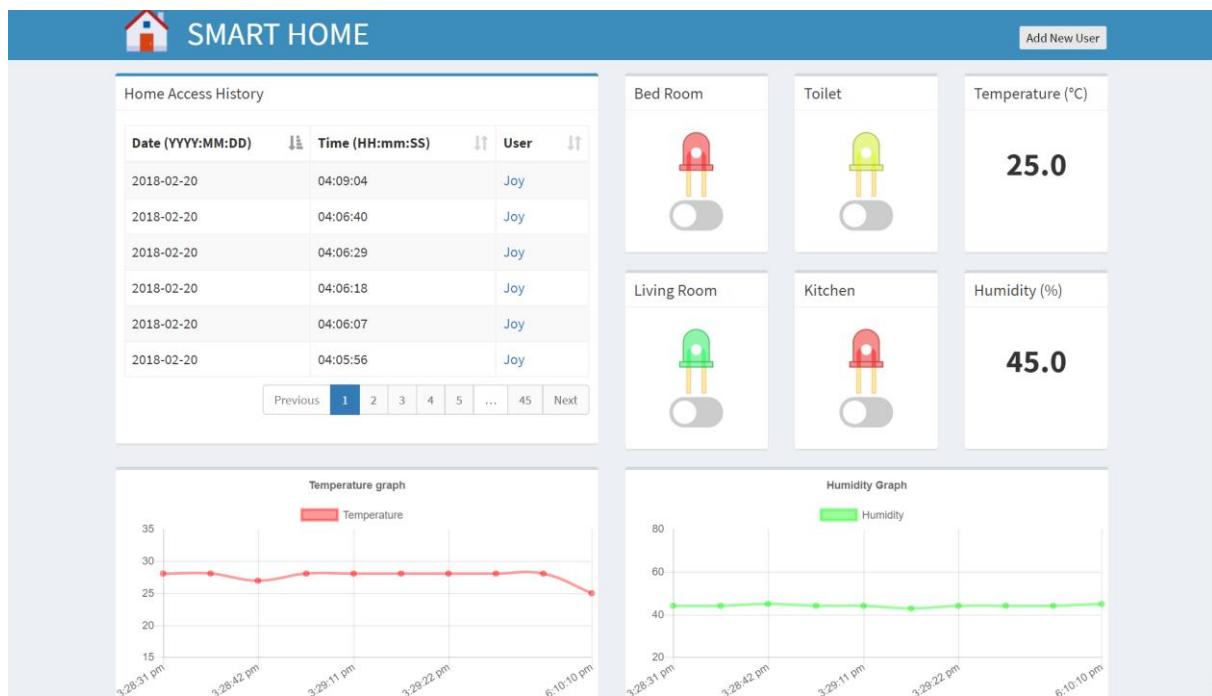
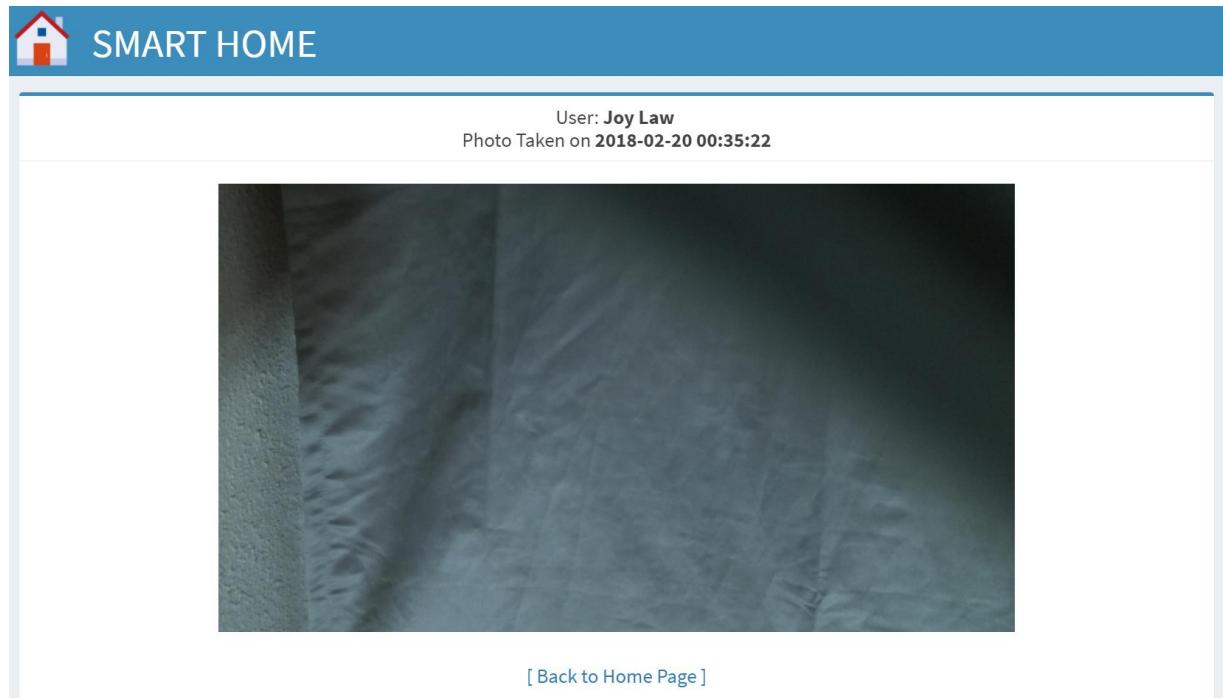


G. Completed Fritzing Diagram

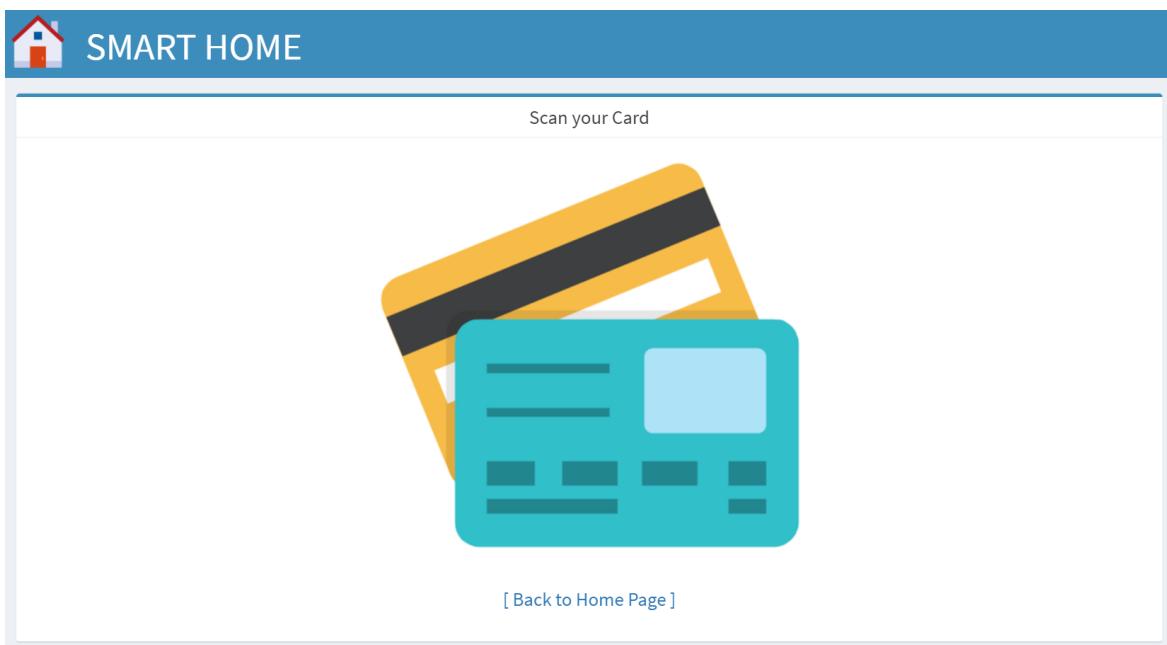


Note: This fritzing diagram excludes the Raspberry Pi, Rainbow Ribbon, piCamera as well as the power source.

H. How does the Web Application Look Like?

index.htmlimage.html

newUser.html



Section 2

Hardware Requirements

Hardware Checklist

A review of the hardware components needed for this assignment.

Hardware		Quantity
1)	Raspberry Pi 3 Model B	1
2)	Breadboard	1
3)	T-Cobbler Kit	1
4)	Ribbon Cable	1
5)	DHT11 Temperature & Humidity Sensor	1
6)	220Ω Resistor for DHT11 Sensor	1
7)	LED	4
8)	1.6kΩ Resistor for LED	4
9)	RFID/NFC MFRC522 Card Reader/Writer Module	1
10)	Raspberry Pi Camera (piCam)	1
11)	Jumper Wire	18

Section 3

Install the Necessary Libraries

A. Install Flask

- a) To install flask:

```
sudo pip install flask
```

B. Install gevent

- a) To install gevent:

```
sudo pip install gevent
```

C. Install Python-dev

- a) To install the Python development libraries:

```
sudo apt-get install python-dev
```

D. Install SPI-Py

- a) Set up the SPI Python libraries as the card reader uses the SPI interface:

```
cd /home/pi  
git clone https://github.com/lthiery/SPI-Py.git  
cd /home/pi/SPI-Py  
sudo python setup.py install
```

E. Install Boto

- a) To install Boto, the Python library for AWS:

```
sudo pip install boto3
```

F. Install AWS CLI

- a) To install the AWS Command-Line Interface Client:

```
sudo pip install awscli
```

G. Install AWS Python Library

- a) To install the AWS Python Library:

```
sudo pip install AWSIoTPythonSDK
```

H. Install Paho MQTT

- a) As the AWS Python SDK has dependency on paho-mqtt, install Paho MQTT:

```
sudo pip install paho-mqtt
```

Section 4

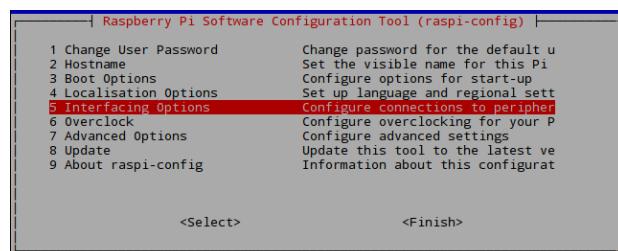
Enable the Necessary Configurations

A. Enable SPI via raspi-config

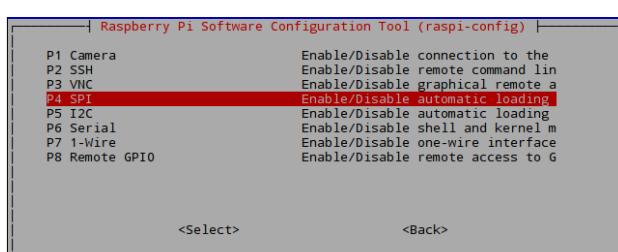
- a) In your command prompt, enter the following:

```
sudo raspi-config
```

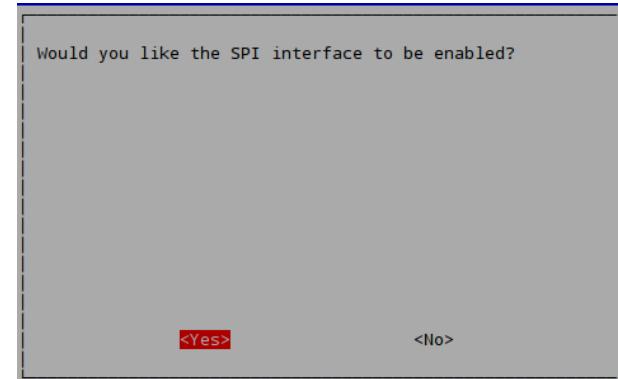
- b) Using your up-down arrow on your keyboard, move the red highlight over to the following, 5 Interfacing Options, and press “Enter” on your keyboard.



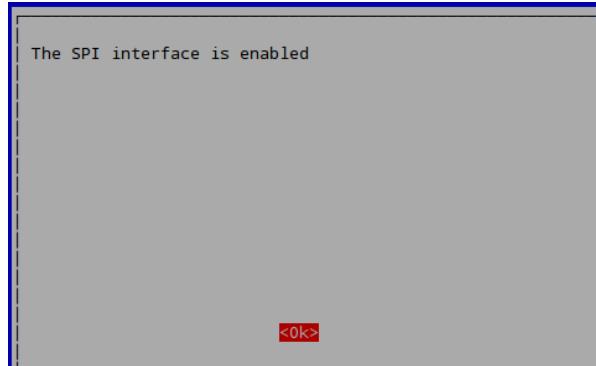
- c) Move the red highlight over to P4 SPI and press “Enter” on your keyboard.



- d) You will then be prompted as to whether you want to enable SPI interface, press “Enter” on your keyboard again for Yes.



- e) You will then see the following. Press “Enter” again to select Ok.



- f) You will then be directed back to the original raspi-config location, press ‘Tab’ on your keyboard twice, then ‘Enter’ to select Finish.

B. Enable Device Tree in boot.txt

- a) View /boot/config.txt such that SPI is enabled

```
sudo nano /boot/config.txt
```

- b) Ensure that the following are in /boot/config.txt, if not, add them in.

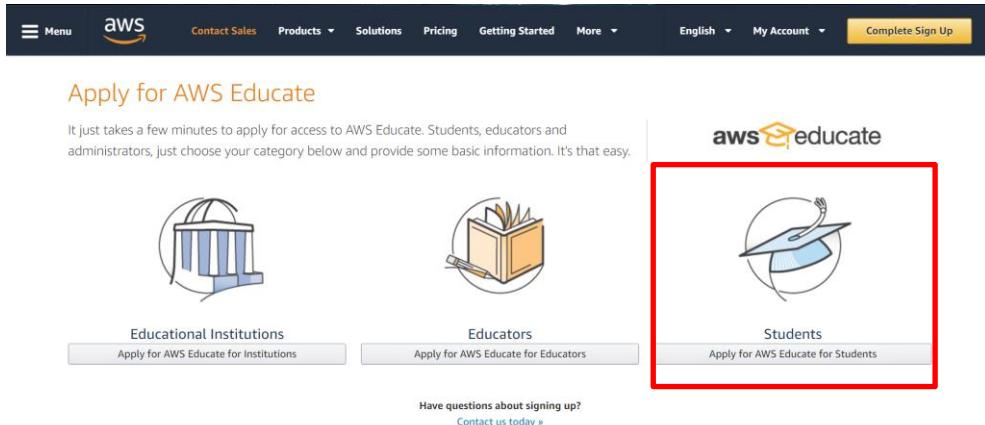
```
device_tree_param=spi=on  
dtoverlay=spi-bcm2835
```

Section 5

Sign Up for an AWS Educate Account

A. Create a New AWS Educate Account

- a) Go to AWS Educate website: <https://aws.amazon.com/education/awseducate/apply/>
- b) Select 'Students' by click on 'Apply for AWS Educate for Students'



- c) You will then be redirect to the following page to fill in your personal information.

The screenshot shows the AWS Educate application form. At the top, it says "aws educate" and "Apply to join AWS Educate". Below that, the title "Step 2/3: Tell us about yourself" is displayed. The form fields include:

- School dropdown: Singapore Polytechnic School of DMIT
- Location dropdown: Singapore
- Name dropdown: Joy
- Major dropdown: Information Technology
- Email dropdown: JOYLAW.15@ichat.sp.edu.sg
- Course dropdown: Undergraduate-Intro Courses
- Year dropdown: 05
- Year dropdown: 2018
- Year dropdown: 9
- Year dropdown: 1998
- Promo Code input field

At the bottom left, there's a note: "Please click the box below to help assure that a person and not an automated program is submitting this application. If a set of letters is displayed enter them on the line. If you have any difficulty with the letters, you can click the reload icon to get a new set of letters, or click the headphones to hear audio of what to enter." Below this is a reCAPTCHA box with the text "I'm not a robot" and a checkbox. To the right of the checkbox is the reCAPTCHA logo and link. At the bottom right is a "NEXT" button with a circular arrow icon.

- d) Click 'Next' to proceed.
e) Select 'Click here to select an AWS Educate Starter Account' and click 'Next'

This section contains two options:

- Click here to enter an AWS Account ID
Approved students are sent a welcome email and benefits including an AWS promotional code.
- Click here to select an AWS Educate Starter Account
An AWS Educate Starter Account is a free, capped-account that doesn't require a credit card. There are some usage limitations including an approximately 25% reduction in access to AWS services. Because Starter Accounts are capped, a separate AWS promotional code is not provided.

Below these options is a note: "With a Starter Account you will receive access to AWS Educate and a lab account with your usage of AWS services capped at the lab amount. A separate AWS promotional credit will not be provided to you." At the bottom is a "NEXT" button with a circular arrow icon.

- f) If the previous steps have been done properly, you should see the following web page.



- g) The following step may take awhile, but once your AWS Educate account has been set up, you will receive an email that you used to signed up an account for. It will look similar to the following:

Email Verification - AWS Educate Application

 AWS Educate Support <support@awseducate.com>
Today, 4:13 AM
You ▾

 Reply | ▾

Hello Joy,

Thank you for submitting your AWS Educate application!

In order for your AWS Educate application to be processed, we need to verify your email address. Please use the verification URL below to confirm your email address, review the AWS Educate program terms and conditions, and complete the application process.

<https://www.awseducate.com/ConfirmEmail?ref=d9f00b6fcba68c84e5a7e561bcf535be>

Thank you,

The AWS Educate Team

- h) Click on the link that is attached to the email to verify your email address and you will be redirected to the following web page for terms and condition. Scroll down to the bottom of the terms & condition, then select 'I Agree' and click 'Submit'

Terms & Conditions

language. If we provide a translation of the English language version of this Agreement, the English language version of this Agreement will control if there is any conflict.

10.0. CONTRACTING ENTITY
Notwithstanding anything to the contrary in these Terms:

10.1 **India Customers.** If you are located in India, your contracting party will be Amazon Internet Services Private Limited ("AISPL"), and this Agreement is an agreement between you and AISPL, located at Ground Floor, EROS Plaza, Eros Corporate Centre, Nehru place, New Delhi, India – 110019. If you are located in India, all references to "AWS," "we," or "us" in this Agreement shall be deemed as referring to AISPL. Additionally, if you are located in India, this Agreement shall be deemed to differ from the above provisions as follows:

(a) The Amazon.com Privacy Notice defined in Section 4.1 shall be deemed to refer to the Amazon.in Privacy Notice located at <http://www.amazon.in/gp/help/customer/display.html?nodeId=200534380>; and
(b) Under Section 9.4, any notice by you to AISPL under this Agreement must be made by registered or certified mail to Amazon Internet Services Private Limited, Ground Floor, Eros Corporate Towers, Nehru Place, New Delhi - 110 019, India (not to Amazon Web Services, Inc.).

You must scroll through the entire Terms and Conditions before accepting or declining.

I Agree

I Decline

SUBMIT 

- i) If the verification is successful, you should see the following web page.



- j) Additionally, you will receive an email similar to the one below.

Thank You for Applying for AWS Educate



AWS Educate Support <support@awseducate.com>

Tue 16/01, 09:36

LAW SI YING JOY ▾



Reply all | ▾

Hi Joy,

Thank you for applying for AWS Educate. We have received your application, and it is currently under review. You will receive an email once the review is complete.

If you have any questions, please click [here](#) to contact AWS Educate support.

Thank You!

The AWS Educate Team

- k) The review process may take a while, but once it is successful, you will receive an email notifying you that your AWS Educate account is ready for use.

AWS Educate Application Approved



AWS Educate Support <support@awseducate.com>

Wed 17/01, 04:14

LAW SI YING JOY ▾



Reply all | ▾

Dear Joy,

Congratulations!

Your AWS Educate application has been approved. As a member of the AWS Educate program, you will gain access to the benefits listed below:

[AWS Educate Student Portal](#)

The AWS Educate Student Portal is the hub for AWS Educate students around the world to find AWS content to help with classwork, connect to self-paced labs and training resources.

[Click here](#) to set your password / login to the AWS Educate Student Portal. After logging in, click AWS Account at the top of the page to access AWS services, whether you entered an AWS ID or selected Starter Account on your application. Note that Starter Accounts are not eligible for AWS Free Tier.

Bookmark the AWS Educate Student Portal for easy access, or [click here](#) to sign in directly.

You can access a video walk-through of the AWS Educate Student portal [here](#).

[Free AWS Essentials Training](#)

To access our foundational AWS Essentials online learning class for free and find other self-paced labs, you must have either an AWS account or an Amazon ID.

- If you have an AWS account, sign in and [click here](#) to receive these benefits.
- If you do not have an AWS account, [click here](#) and follow the instructions to create an Amazon ID to access these benefits.

Once you access the Training and Certification portal, click "Find Training" and search for "Technical" to easily locate and enroll in AWS Technical Essentials on-line training. You can access AWS training any time after setting up your account by clicking [here](#).

Thank you again for participating in AWS Educate and we hope you enjoy the program!

The AWS Educate Team

Getting too much email from AWS Educate Support <support@awseducate.com>? [You can unsubscribe](#)

- l) Following the instructions given in the email, set your password. Afterwards, click on the second 'click here' to access the login page. You can bookmark the page for easy access later on.

Section 6

Sign in to AWS

A. Sign in to AWS

- a) Login with the email and password that you have previously set. And click 'SIGN IN'.

welcome :)

The screenshot shows a login form with a light gray background. At the top left is a placeholder text 'welcome :)'. Below it are two input fields: the first contains the email 'joylaw.15@ichat.sp.edu.sg' and the second contains a password represented by a series of dots. To the right of these fields is a blue 'SIGN IN' button with a white circular arrow icon. At the bottom left of the form is a blue link 'Forgot password?'. The entire form is enclosed in a thin gray border.

- b) Upon logging in, you should see the following web page. Click on 'AWS Account' on the top right-hand corner.

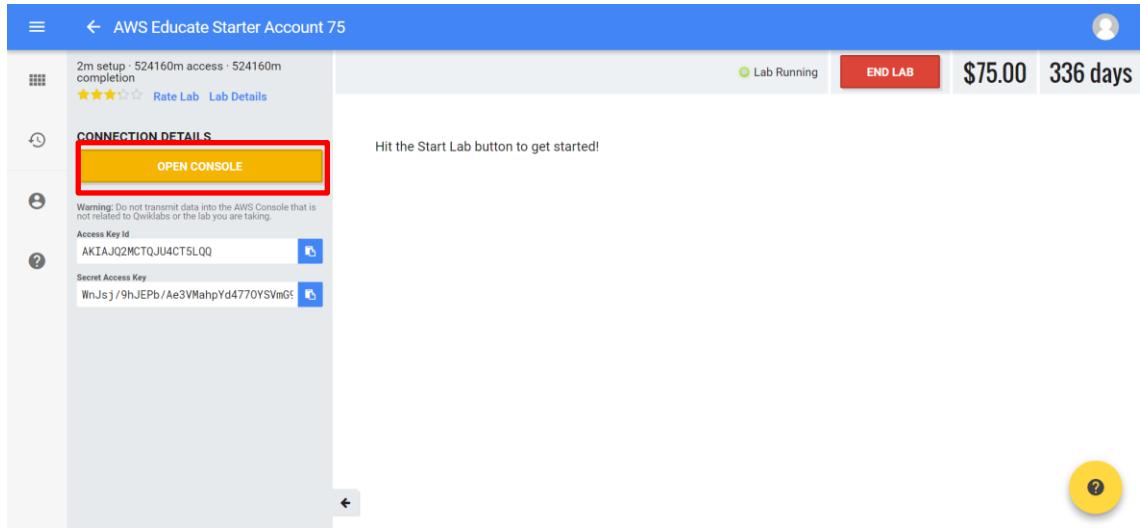
The screenshot shows the AWS Educate dashboard. At the top, there's a blue header bar with the 'aws educate' logo, user profile 'Joy Law', and various navigation links: Portfolio, Career Pathways, Learn, Badges, Jobs, AWS Account (which has a red box around it), and Logout. Below the header, there's a message 'New: Click to check out video updates from Educate @ re:invent 2017.' On the left, there's a sidebar with text about cloud technology and a 'Begin your journey today!' button. The main content area has a video player showing a video titled 'What is Cloud Computing? - Amazon Web Se...'. To the right, there's a 'Suggested Jobs' sidebar listing 'Solution Analyst - Technology Consulting' and 'Summer 2018 Intern, Software Engineer Salesforce'.

- c) Click on 'Go to your AWS Educate Starter Account'



Note: Clicking this button will take you to a third party site managed by qwikLABS, Inc. ("Third Party Servicer"). In addition to the AWS Educate terms of service, your use of the AWS Educate Starter Account is governed by the Third Party Servicer's terms, including its Privacy Policy. AWS assumes no responsibility or liability and makes no representations or warranties regarding services provided by a Third Party Servicer.

- d) You will then be redirected to the following web page. Click on 'OPEN CONSOLE' on the left-hand menu.

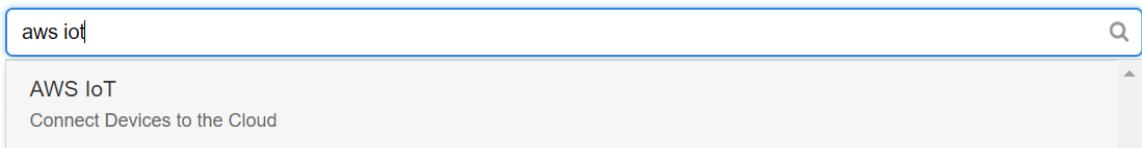


Section 7

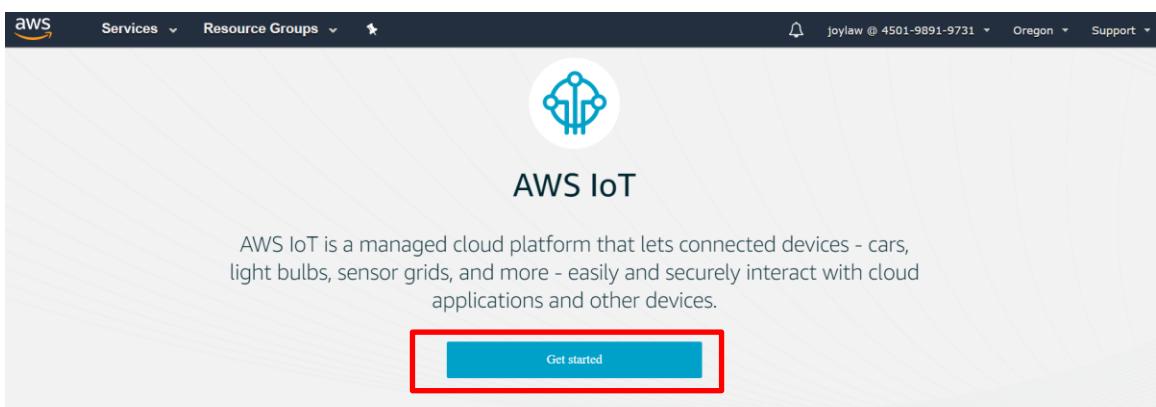
Configure AWS Services

A. Configure AWS IoT

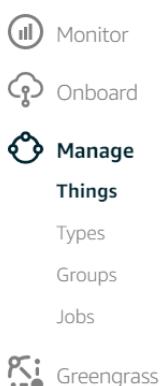
- a) Go back to the home page of the console and search for the service: AWS IoT.



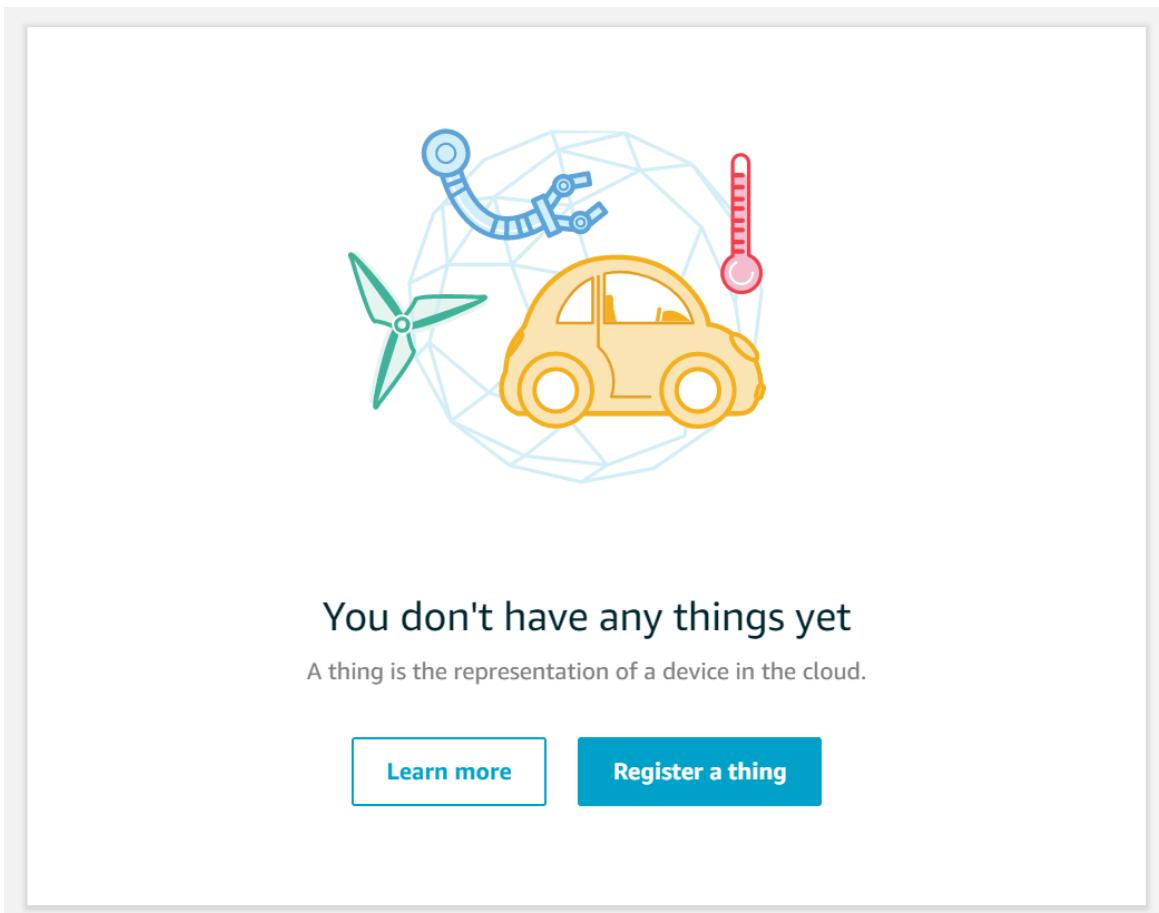
- b) You will then be redirect to the following page and click on 'Get started'



- c) On the left-hand panel, click on 'Manage'



- d) Then click on 'Register a thing'.



- e) You will then be asked if you want to 'Create a single thing' or 'Create many things'. Select the first one – 'Create a single thing'.

Creating AWS IoT things

An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. [Learn more](#).

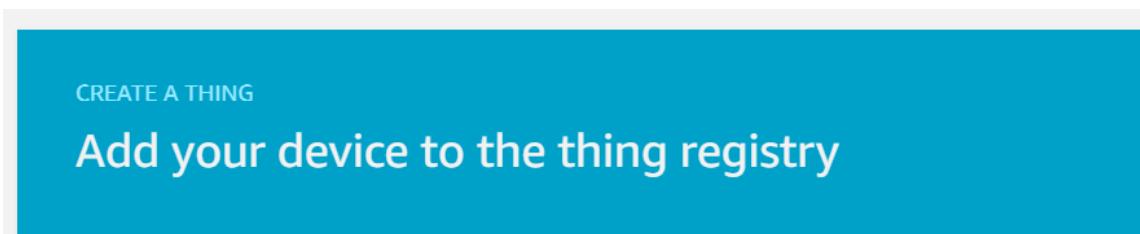
Register a single AWS IoT thing
Create a thing in your registry

Create a single thing

Bulk register many AWS IoT things
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Create many things

- f) Add in a name for your Raspberry Pi, leave the rest as default and click 'Next'



- g) When asked if you want to create a certificate for your thing, select 'Create thing without certificate'.

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)
This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

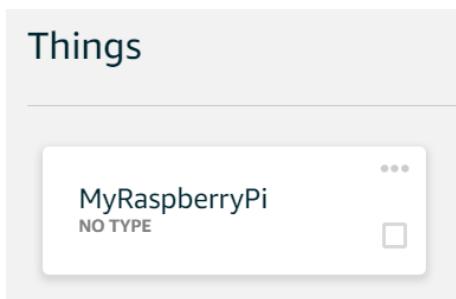
Create with CSR
Upload your own certificate signing request (CSR) based on a private key you own.

Use my certificate
Register your CA certificate and use your own certificates for one or many devices.

Skip certificate and create thing
You will need to add a certificate to your thing later before your device can connect to AWS IoT.

Create thing without certificate

- h) Once successful, you will be redirected to the following web page and you should see the following. Click on 'MyRaspberryPi'.



- i) You should see the following. Click on 'Interact'.

The screenshot shows the AWS IoT Thing details page for 'MyRaspberryPi'. At the top, it displays the Thing name 'MyRaspberryPi' and 'NO TYPE'. On the left, a navigation menu lists 'Details', 'Security', 'Groups', 'Shadow', 'Interact' (which is highlighted with a red box), 'Activity', and 'Jobs'. The main content area shows 'Thing ARN' as 'arn:aws:iot:us-west-2:450198919731:thing/MyRaspberryPi' and 'Type' as 'No type'. An 'Edit' button is located at the top right of the main content area.

- j) Copy the following onto a notepad as it will be used later on. Next, click 'Security' on the left menu

HTTPS

Update your Thing Shadow using this Rest API Endpoint. [Learn more](#)

a7gx67jy14o4h.iot.us-west-2.amazonaws.com

- k) Click on 'Create certificate' to generate a X.509 certificate as well as key pair.

The screenshot shows the AWS IoT Thing details page for 'MyRaspberryPi'. The left navigation menu includes 'Details', 'Security' (which is highlighted with a red box), 'Groups', 'Shadow', 'Interact', 'Activity', and 'Jobs'. The main content area is titled 'Certificates' and features a circular icon with a shield. It contains text: 'To securely connect to AWS IoT, your thing will need a certificate and policy.' Below this, it says 'Certificates help things establish a secure connection. AWS IoT policies give things permission to access AWS IoT resources (like other things, MQTT topics, or thing shadows)'. At the bottom of this section are two buttons: 'Create certificate' (which is highlighted with a red box) and 'View other options'.

- I) Once successful, you should see the following web page. Click on 'Activate'

Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

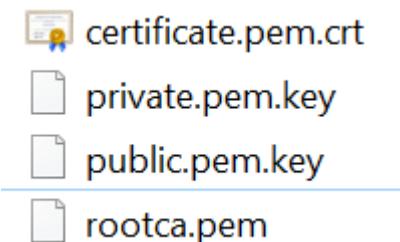
In order to connect a device, you need to download the following:

A certificate for this thing	4bc963d6c5.cert.pem	Download
A public key	4bc963d6c5.public.key	Download
A private key	4bc963d6c5.private.key	Download

You also need to download a root CA for AWS IoT from Symantec:
A root CA for AWS IoT [Download](#)

Activate

- m) Click on the 4 download links to download the respective files and rename them accordingly.



- n) Afterwards, click on 'Attach a policy' at the bottom and you will be redirected to the following page.

Add authorization to certificate

You are attaching a policy to the following certificate:
dacdd60feb5c55909ae23627a6e81fad174c904fc996eb7b7689d0e5450234ec

Select a policy to attach to this certificate:

No match found
There are no policies in your account.

[Create new policy](#)

- o) Afterwards, click on 'Create new policy' and enter the following details accordingly then click 'Create'.

Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the [AWS IoT Policies documentation page](#).

Name
MyRaspberryPiSecurityPolicy

Add statements

Policy statements define the types of actions that can be performed by a resource.

Action
iot:*

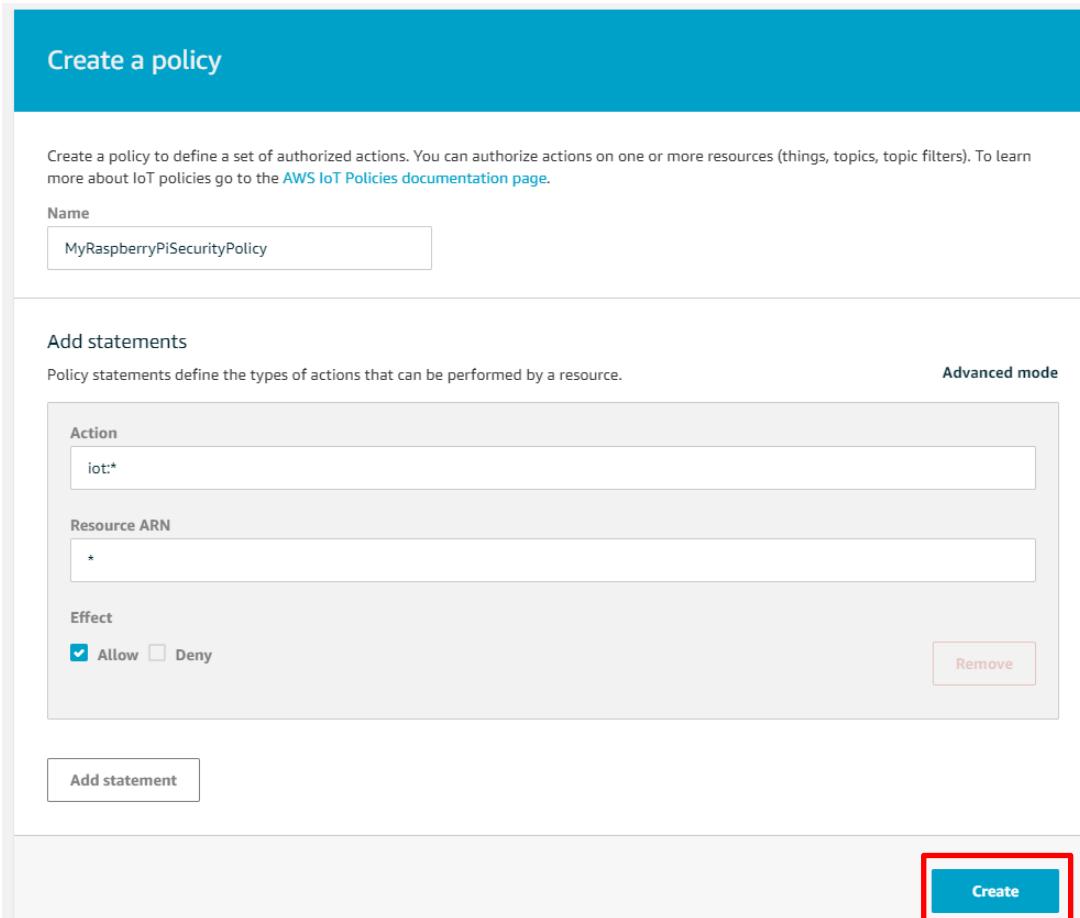
Resource ARN
*

Effect
 Allow Deny

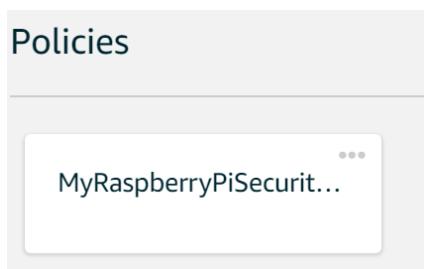
[Remove](#)

[Add statement](#)

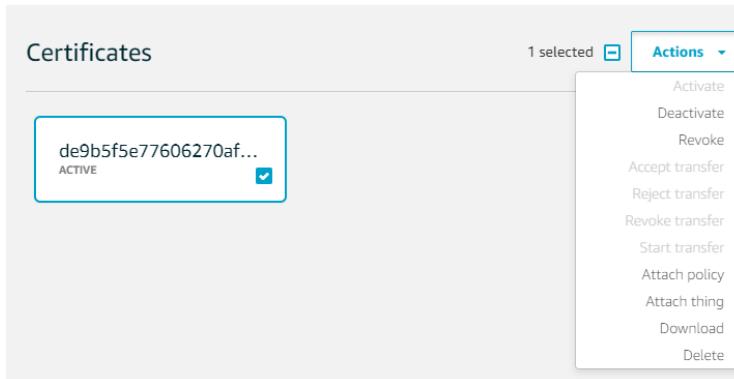
Create



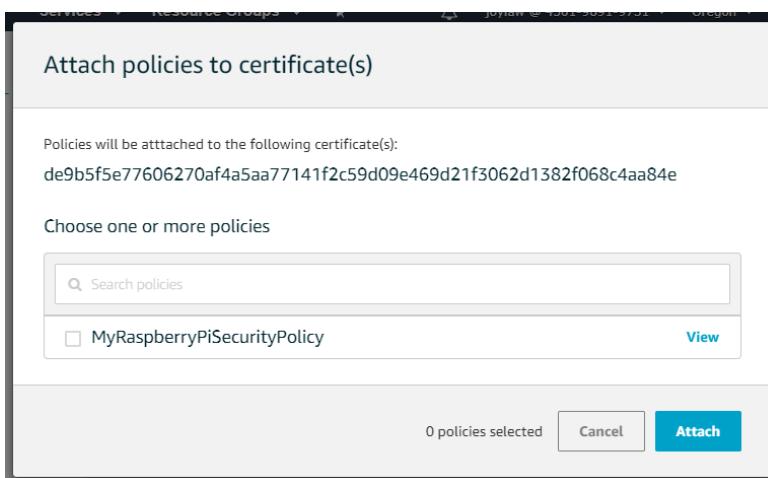
- p) Once successful, you should see the following policy.



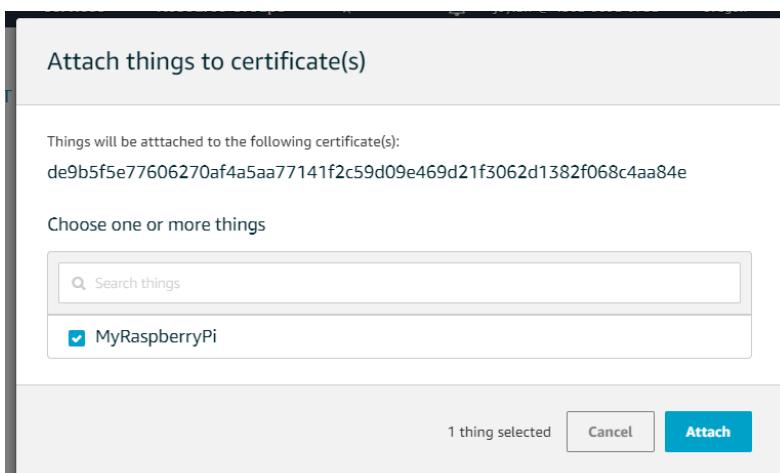
- q) On the left-hand menu, select ‘Certificate’. You should see the following X.509 certificate that you have created previously. Click on the checkbox beside the certificate and from the ‘Action’ drop down and select ‘Attach policy’.



- r) You will then be redirected to the following screen and select ‘MyRaspberryPiSecurityPolicy’ then ‘Attach’. Once successful, you will be notified.



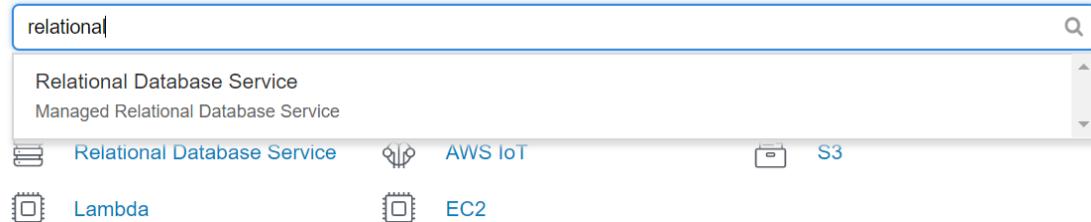
- s) Click on the checkbox beside the certificate again and from the ‘Action’ drop down and select ‘Attach thing’, you will then see the following screen. Select ‘MyRaspberryPi’ and click ‘Attach’



B. Configure AWS Relational Database Service

- a) Continuing from the previous step, search for ‘Relational Database Service’ and click on it to begin configuring a MySQL database.

AWS services



- b) You will see the following page. Click on ‘Instances’ on the left-hand menu.

- c) Click on ‘Launch DB instance’. Note that you should not have an existing ‘DB Instance’ prior to this step.

- d) When selecting an engine, pick MySQL and click 'Next'.

Select engine

Engine options

- Amazon Aurora

Amazon Aurora
- MySQL

MySQL
- MariaDB

MariaDB
- PostgreSQL

PostgreSQL
- Oracle

Oracle
- Microsoft SQL Server

Microsoft SQL Server

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 16 TB.
- Instances offer up to 32 vCPUs and 244 GiB Memory.
- Supports automated backup and point-in-time recovery.
- Supports cross-region read replicas.

Only enable options eligible for RDS Free Usage Tier [info](#)

[Cancel](#) [Next](#)

- e) When prompted for use case, pick 'Dev/Test – MySQL' and click 'Next'.

Choose use case

Use case

Do you plan to use this database for production purposes?

Use case

- Production - Amazon Aurora** Recommended
MySQL-compatible, enterprise-class database at 1/10th the cost of commercial databases.
- Production - MySQL**
Use Multi-AZ Deployment and Provisioned IOPS Storage as defaults for high availability and fast, consistent performance.
- Dev/Test - MySQL**
This instance is intended for use outside of production or under the [RDS Free Usage Tier](#).

Billing is based on [RDS pricing](#).

[Cancel](#) [Previous](#) [Next](#)

- f) Leave everything as default except for the following:

DB instance class [info](#)

db.r4.large — 2 vCPU, 15.25 GiB RAM

- g) For the Settings portion, enter a name for 'DB instance identifier' and enter a master username and password. Take note that these information will be made use of later on. Click 'Next'.

Settings

DB instance identifier [info](#)
Specify a name that is unique for all DB instances owned by your AWS account in the current region.

DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance".

Master username [info](#)
Specify an alphanumeric string that defines the login ID for the master user.

Master Username must start with a letter. Must contain 1 to 16 alphanumeric characters.

Master password [info](#) Confirm password [info](#)

Master Password must be at least eight characters long, as in "mypassword".

[Cancel](#) [Previous](#) [Next](#)

- h) Leave everything as default except for the following:

Availability zone [info](#)

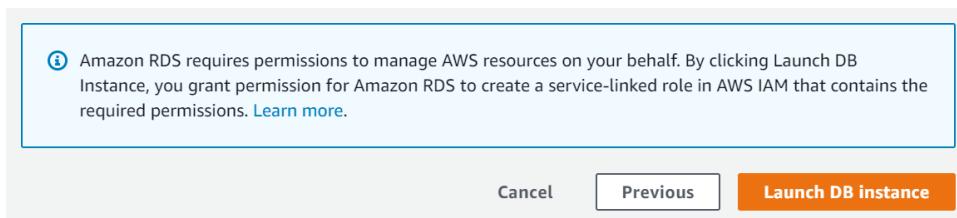
us-west-2a

Database name

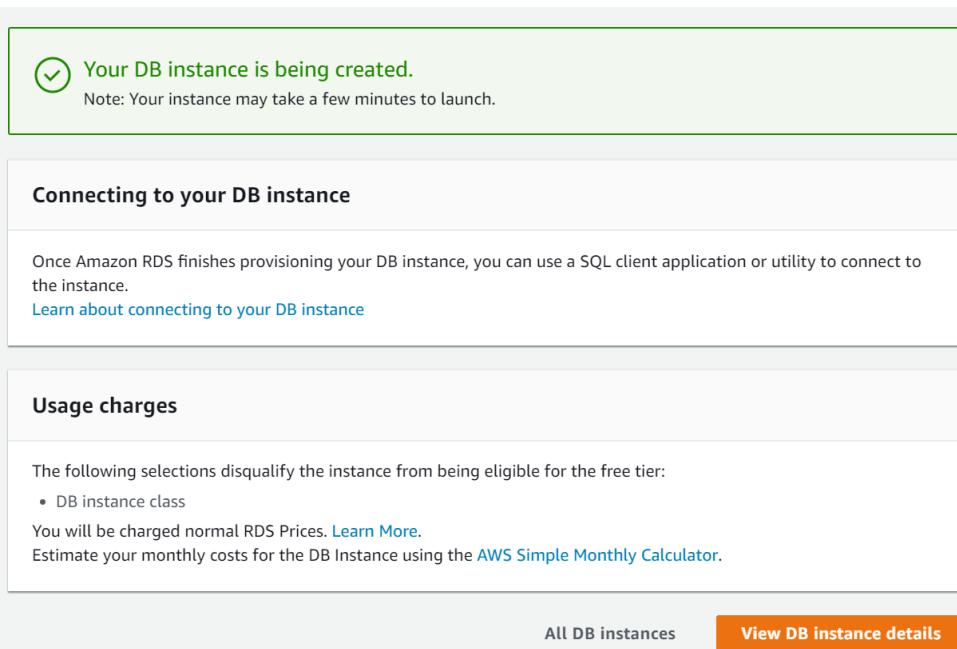
iotca2

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

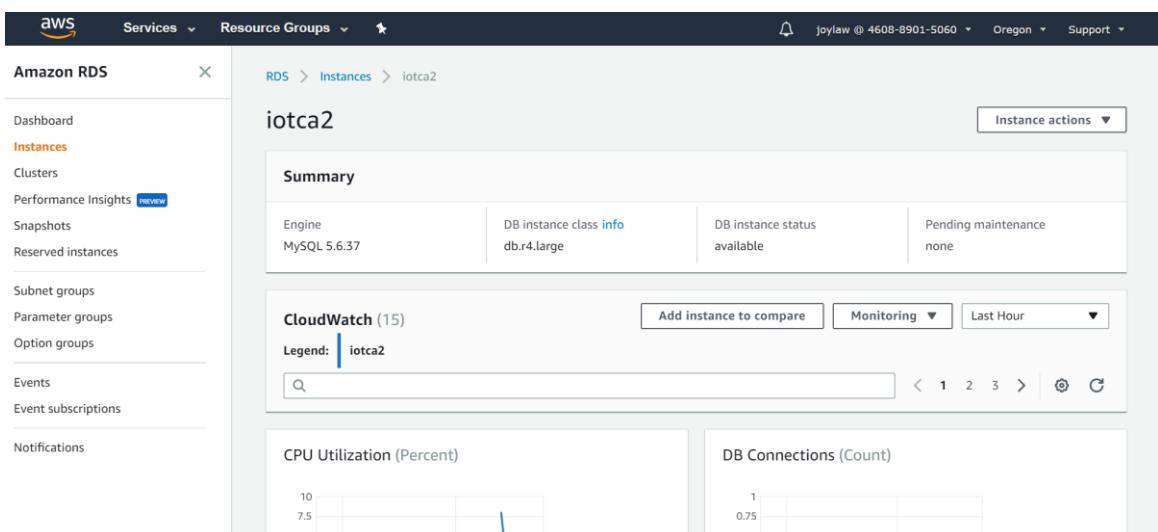
- i) Click on 'Launch DB instance' after the configuration has been done.



- j) If successful, you should see the following page. Click on 'View DB instance details'. It will take awhile for the database to complete its creation.



- k) When it is completed, you should see the following page.

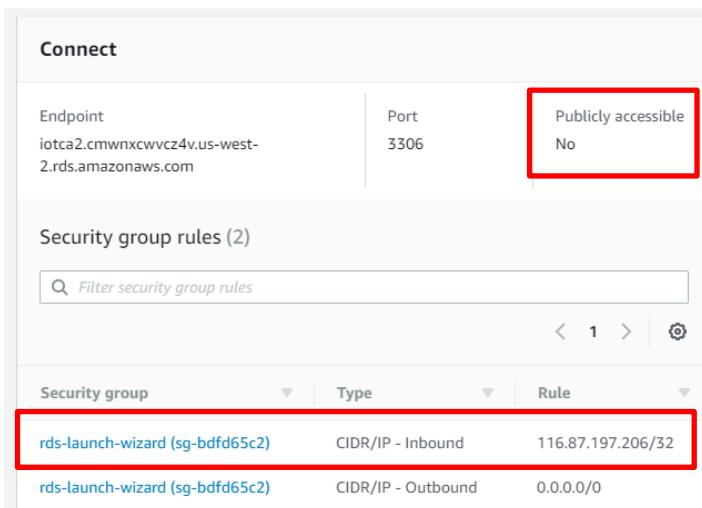


C. Create a Database using AWS RDS

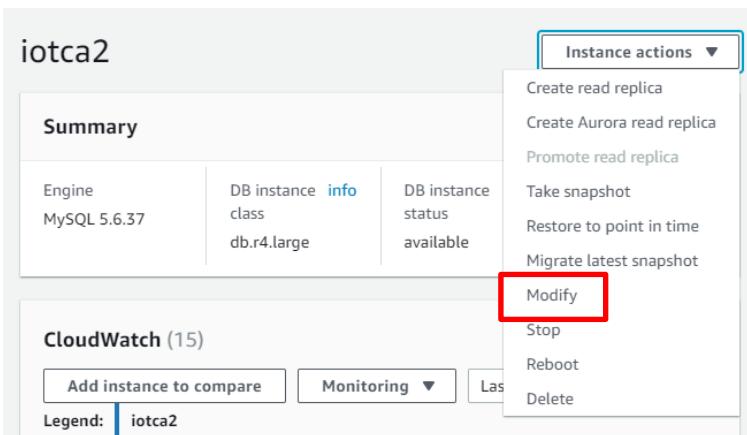
- a) In your Raspberry Pi, enter the following

```
pi@joylaw:~ $ mysql -h iot.cygylyw5wux.us-west-2.rds.amazonaws.com -u admin -p
```

- b) Replace 'iot.cygylyw5wux.us-west-2.rds.amazonaws.com' with your own Endpoint which can be found here and 'admin' with the Master username that you have previously created. Scroll down to 'Connect' to ensure that the 'Publicly accessible' has been set to 'Yes' and the 'CIDR/IP – Inbound' rule has been set to '0.0.0.0/0'. If not, continue to the following steps. Else, continue to part i.



- c) Scroll all the way up and click one the drop down button 'Instance actions'. Select 'Modify'.



- d) Scroll down to ‘Network & Security’, under ‘Public accessibility’, pick ‘Yes’ and then scroll down to the bottom and click on ‘Continue’.

Network & Security

Subnet group
Use this field to move the DB instance to a new subnet group in another vpc. [Learn more](#).

default ▾

Security group
List of DB security groups to associate with this DB instance.

Select security groups ▾

rds-launch-wizard (sg-bdfd65c2) (vpc-de71f4b9) X

Certificate authority
Certificate authority for this DB instance

rds-ca-2015 ▾

Public accessibility [info](#)

Yes
EC2 instances and devices outside of the VPC hosting the DB instance will connect to the DB instances.
You must also select one or more VPC security groups that specify which EC2 instances and devices can connect to the DB instance.

No
DB instance will not have a public IP address assigned. No EC2 instance or devices outside of the VPC will be able to connect.

- e) In this web page, select ‘Apply immediately’ then ‘Modify DB Instance’.

Modify DB Instance: iotca2

Summary of Modifications
You are about to submit the following modifications. Only values that will change are displayed. Carefully verify your changes and click Modify DB Instance.

Attribute	Current Value	New Value
Publicly accessible	No	Yes

Scheduling of Modifications

When to Apply Modifications

Apply during the next scheduled maintenance window
Current maintenance window: thu:13:02-thu:13:32

Apply immediately
The modifications in this request and any pending modifications will be asynchronously applied as soon as possible, regardless of the maintenance window setting for this database instance.

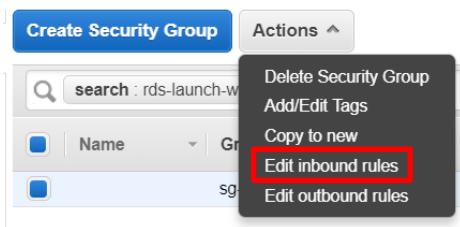
Potential unexpected downtime
If you choose to apply changes immediately, please note that any changes in the pending modifications queue are also applied. If any of the pending modifications require downtime, choosing this option can cause unexpected downtime.

Cancel Back **Modify DB Instance**

- f) Under the ‘Security group rules’, click on the security group that corresponds to the type ‘CIDR/IP – Inbound’

Security group	Type
rds-launch-wizard (sg-bdfdf65c2)	CIDR/IP - Inbound
rds-launch-wizard (sg-bdfdf65c2)	CIDR/IP - Outbound

- g) You will be redirected to another web page. From there, click on the ‘Action’ drop down button and select ‘Edit inbound rules’.



- h) Change the default to the following values and click ‘Save’

Type	Protocol	Port Range	Source	Description
MYSQL/Aurora	TCP	3306	Custom	0.0.0.0/0

- i) You will then be prompted for a password, enter the Master password you have previously created. If successful, you should see the following:

```
pi@joylaw:~ $ mysql -h iotca2.cyygylwyw.us-west-2.rds.amazonaws.com -u admin
-p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 24
Server version: 5.6.37-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

- j) Run the command 'show databases;' to see all the databases that are available.

```
mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| innodb        |
| iotca2        |
| mysql          |
| performance_schema |
| sys            |
+-----+
6 rows in set (0.22 sec)
```

- k) Pick the database that you have previously created. In our case, iotca2. Run the command 'use iotca2;'

```
|mysql> use iotca2;
|Database changed
```

- l) To create the tables needed for this project. Enter the following:

```
CREATE TABLE rfid_access (rfid_uid VARCHAR(50) NOT NULL, name
VARCHAR(255) NOT NULL, PRIMARY KEY(rfid_uid));

CREATE TABLE rfid_accesslog (id int(10) NOT NULL
AUTO_INCREMENT, date_time VARCHAR(50) NOT NULL, rfid_uid
VARCHAR(50) NOT NULL, PRIMARY KEY(id));

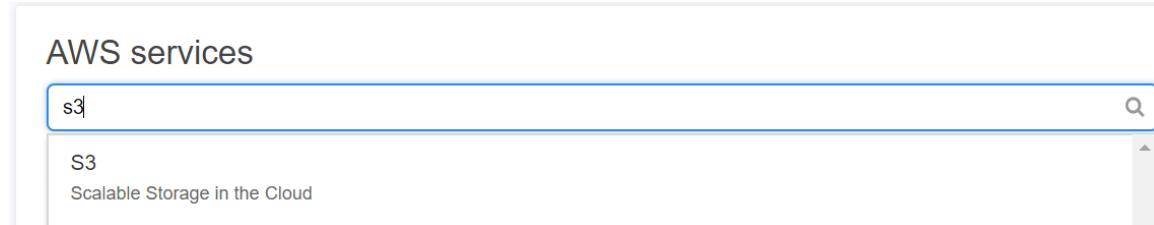
CREATE TABLE temp_humidity (id int(10) NOT NULL
AUTO_INCREMENT, date_time datetime NOT NULL, temperature
VARCHAR(10) NOT NULL, humidity VARCHAR(10) NOT NULL, PRIMARY
KEY(id));
```

- m) Run 'show tables;' to see all the tables you have just created.

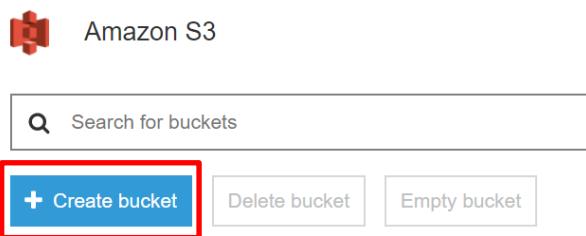
- n) To quit the session, type 'quit'.

D. Configure AWS S3

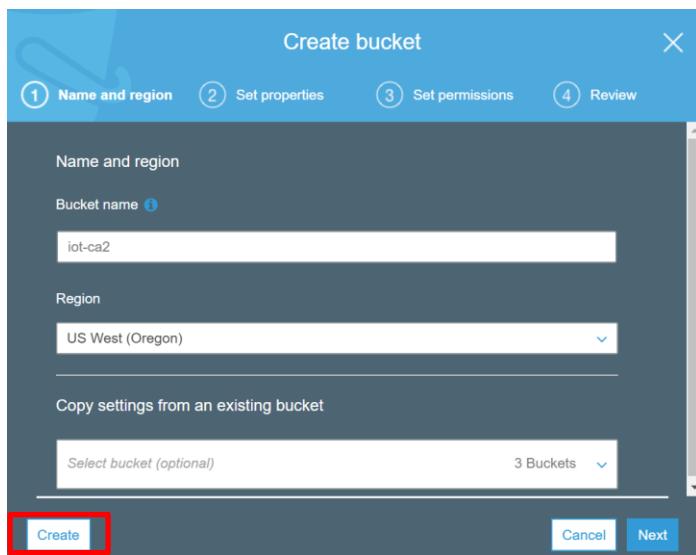
- a) Go back to the home page of the console and search for the service: S3. We will be using S3 to store our images. Click on 'S3'.



- b) Click on the 'Create bucket' button.



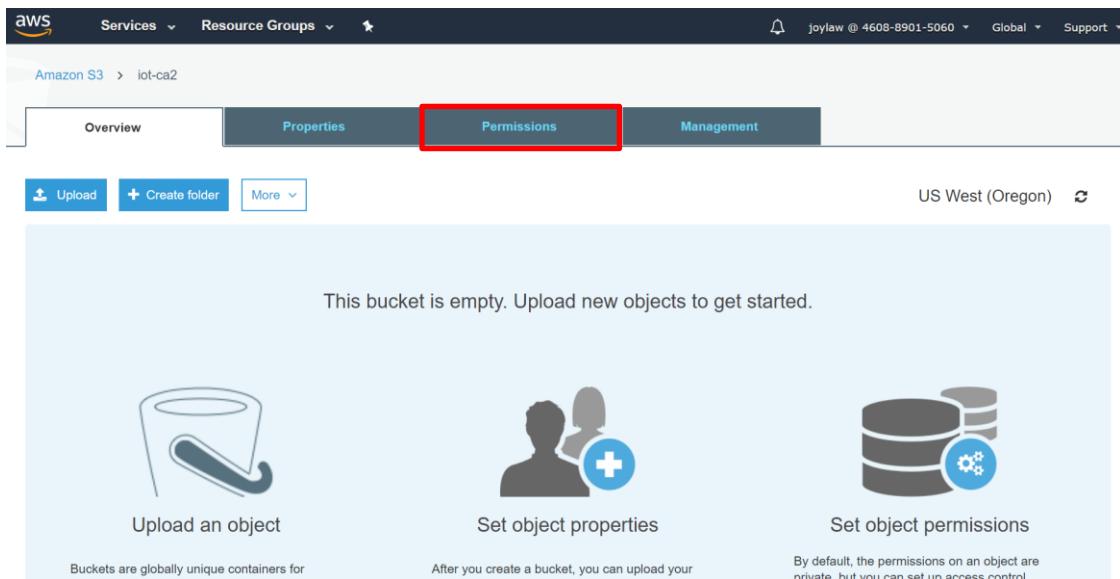
- c) Enter a bucket name and leave the rest as default. Click 'Create'.



- d) Once successful, you should see the following bucket created. Click on the bucket name.

iot-ca2	Not public *	US West (Oregon)	Feb 19, 2018 2:02:58 PM GMT+0800
---------	--------------	------------------	-------------------------------------

- e) You will then be redirected to the following web page and click on ‘Permissions’.



- f) Click on ‘Bucket policy’ and enter the following and change “arn:aws:s3:::iot-ca2” to your own bucket ARN.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MakeItPublic",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::iot-ca2/*"
    }
  ]
}
```

- g) If successful, your bucket should allow public access.

The screenshot shows the 'Bucket Policy' section of the AWS S3 console. At the top are tabs for 'Access Control List', 'Bucket Policy' (which is selected and highlighted in yellow), and 'CORS configuration'. A warning message in a box states: '⚠ This bucket has public access. You have provided public access to this bucket. We highly recommend that you never grant any kind of public access to your S3 bucket.' Below this is a text area titled 'Bucket policy editor ARN: arn:aws:s3:::iot-ca2'. It contains the JSON policy code from the previous step. At the bottom right are buttons for 'Delete', 'Cancel', and 'Save'.

```

1  [
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Sid": "MakeItPublic",
6        "Effect": "Allow",
7        "Principal": "*",
8        "Action": "s3:GetObject",
9        "Resource": "arn:aws:s3:::iot-ca2/*"
10       }
11     ]
12   ]

```

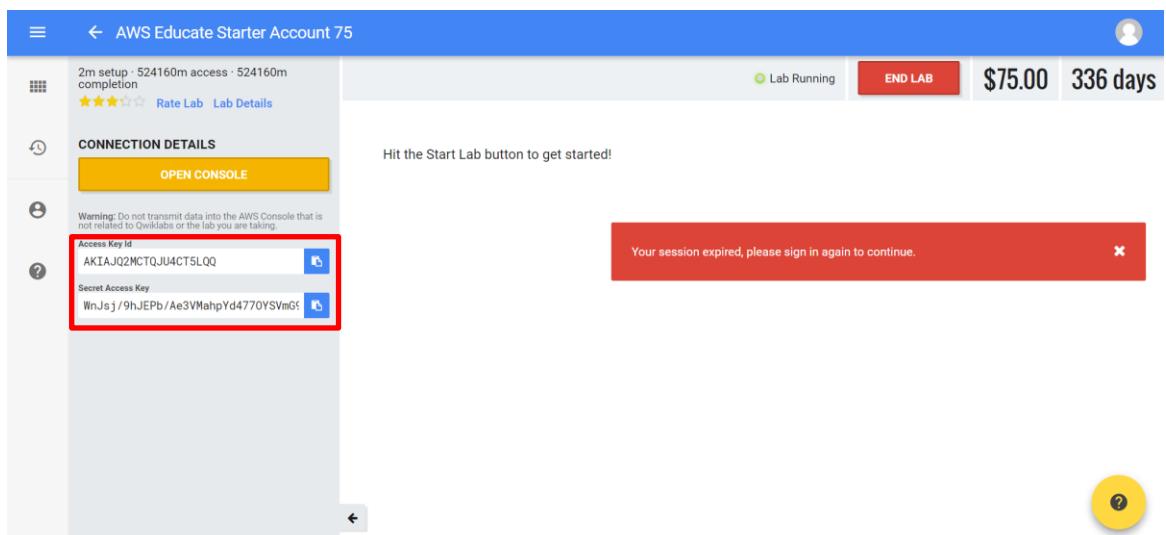
- h) Go to your Raspberry Pi and run the following command to create a new folder

```
mkdir ~/assignment
```

- i) Change directory to ~/assignment and type the command 'aws configure'.

```
pi@joylaw: ~ $ aws configure
AWS Access Key ID [*****5LQQ]: AKIAJQ2MCTQJU4CT5LQQ
AWS Secret Access Key [*****0M95]: WnJsj/9hJEPb/Ae3VMahpYd4770YSVmG9/
ayOM95
Default region name [None]: us-west-2
Default output format [None]:
```

- j) Enter the following accordingly. Your AWS Access Key ID & AWS Secret Access Key can be retrieved from the following web page.



E. Configure AWS Lambda & AWS IoT Rule

- a) Go back to the home page of the console and search for the service: Lambda.

AWS services

lambda



Lambda

Run Code without Thinking about Servers

- b) You will see the following web page. Click on 'Create a function'.

The screenshot shows the AWS Lambda service landing page. The search bar at the top contains the text "lambda". Below the search bar, the "Lambda" service is listed with the description "Run Code without Thinking about Servers". To the right, there is a "Get started" box containing the text "Author a Lambda function from scratch, or choose from one of many preconfigured examples." and a prominent orange "Create a function" button.

- c) Scroll down to 'Role' and select 'Create a custom role'. You will then be redirected to another web page.

Role*

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Create a custom role



The custom role creation experience will open in a new tab. Ensure that popups are enabled to create a custom role.

- d) Accept the defaults and click 'Allow'

Role Description Lambda execution role permissions

IAM Role

Create a new IAM Role



Role Name

lambda_basic_execution

▶ View Policy Document

- e) Go back to the previous tab and refresh. Select the following and click 'Create function'.

Name*
iotca2

Runtime*
Python 2.7

Role*
Define the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.
Choose an existing role
lambda_basic_execution

Create function

- f) Once successfully created, you should see the following:

ARN - arn:aws:lambda:us-west-2:450198919731:function:iotca2

Qualifiers ▾ Actions ▾ Select a test event... ▾ Test Save

⚠ You are not authorized to perform: iam>ListRolePolicies.

✔ Congratulations! Your Lambda function "iotca2" has been successfully created. You can now change its code and configuration. Click on the "Test" button to input a test event when you are ready to test your function.

Configuration Monitoring

▼ Designer

Add triggers
Click on a trigger from the list below to add it to your function.

API Gateway

iotca2

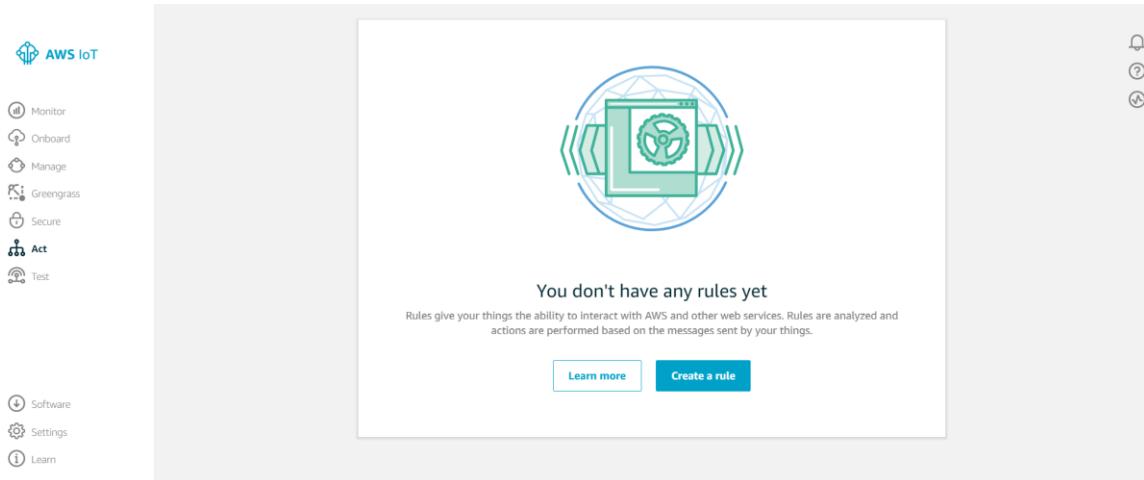
- g) Go back to the home page of the console and search for the service: AWS IoT. Click on it.

AWS services

aws iot

AWS IoT
Connect Devices to the Cloud

- h) On the left-hand menu, click on ‘Act’ and you should be redirected to the following web page. Click on ‘Create a rule’.



- i) Enter the following accordingly and scroll down.

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name	<input type="text" value="iotca2"/>
Description	<input type="text" value="iot ca2"/>

- j) Under the ‘Message source’, enter the following:

Message source	Indicate the source of the messages you want to process with this rule.
Using SQL version	?
<input type="text" value="2016-03-23"/>	▼
Rule query statement	<input type="text" value="SELECT * FROM 'sensors/temp_humid'"/>
Attribute	?
<input type="text" value="*"/>	
Topic filter	?
<input type="text" value="sensors/temp_humid"/>	

- k) Scroll down and under 'Set one or more actions', click on 'Add action'.

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

[Add action](#)

- l) You will then be redirected to the following page. Select 'Invoke a Lambda function passing the message data' and click 'Configure action'.

Select an action.

<input type="radio"/>	 Insert a message into a DynamoDB table DYNAMODB
<input type="radio"/>	 Split message into multiple columns of a database table (DynamoDBV2) DYNAMODBV2
<input checked="" type="radio"/>	 Invoke a Lambda function passing the message data LAMBDA

- m) Click on the drop down bar for 'Function name'. Select 'iotca2' and click on 'Add action'



Invoke a Lambda function passing the message data
LAMBDA

We'll set [the permissions](#) on the Lambda function for you.

*Function name

Choose a resource
iotca2

[Create a new resource](#)

[Add action](#)

- n) Scroll down and press 'Create rule'.

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)



Invoke a Lambda function passing the message data
iotca2

[Remove](#) [Edit](#)

[Add action](#)

Error action

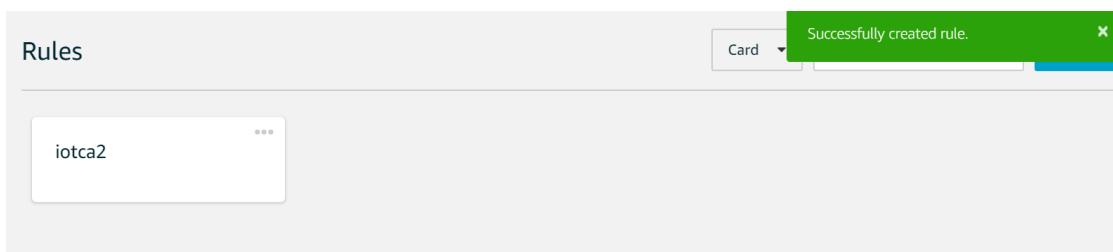
Optionally set an action that will be executed when something goes wrong with processing your rule.

[Add action](#)

[Cancel](#)

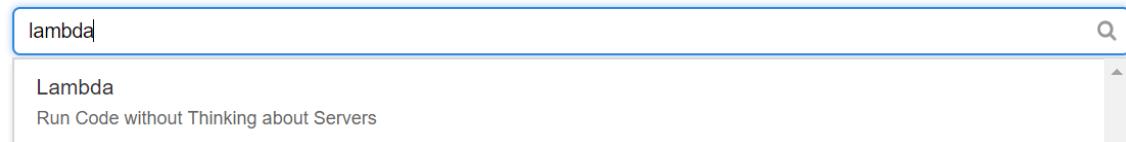
[Create rule](#)

- o) Once successfully created, you will see the following.

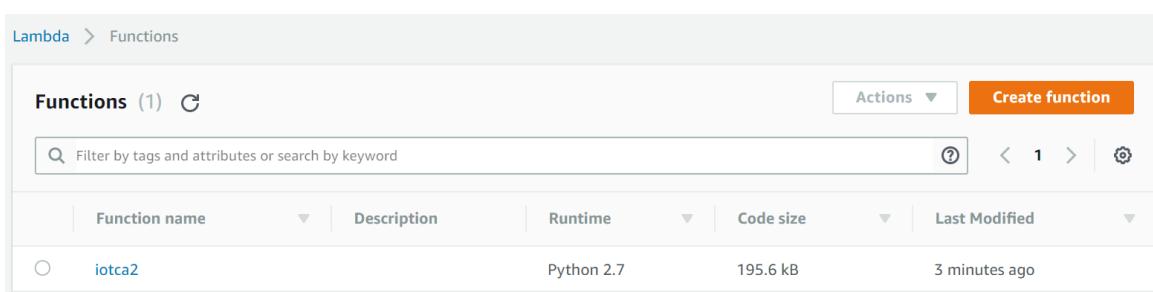


- p) Go back to the home page of the console again and search for the service: Lambda.

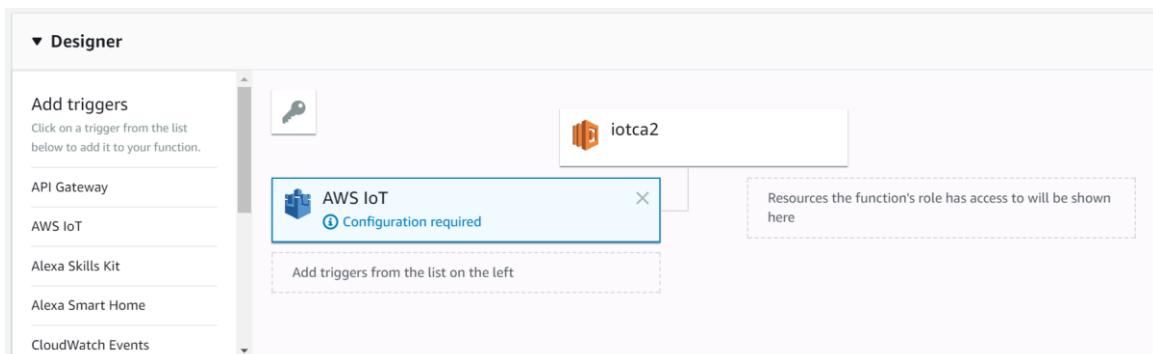
AWS services



- q) Under the function name, select 'iotca2'.



- r) On the left-hand panel, under 'Add triggers', press 'AWS IoT' and you should achieve the following



- s) Scroll down to configure the trigger. Under ‘Configure trigger’, select ‘Custom IoT rule’ then select the rule ‘iotca2’ that you have previously created. Then click ‘Add’.

Configure triggers

IoT type
Configure a custom IoT rule, or set up an IoT button.
 Custom IoT rule
 IoT Button

Rule
Pick an existing rule, or create a new one.
iotca2

Rule description
iot ca2

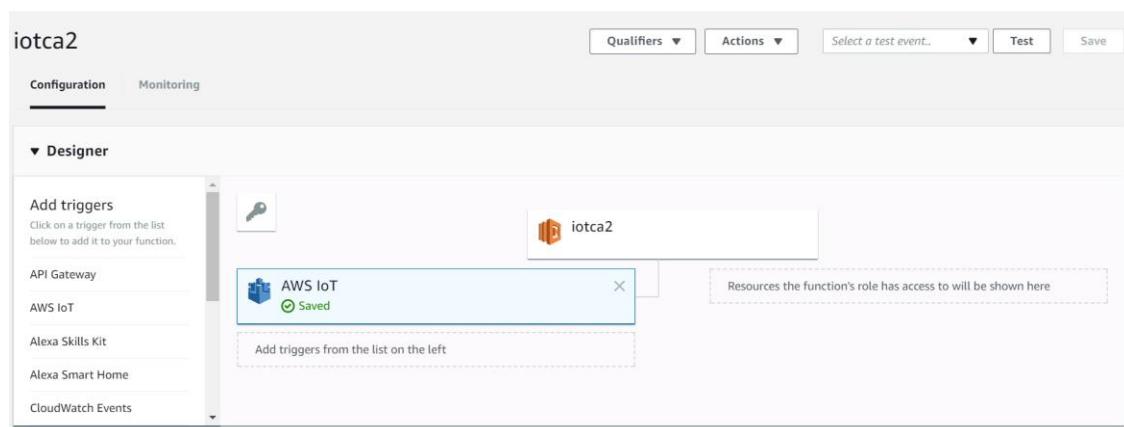
Rule query statement
SELECT * FROM `sensors/temp_humid`

Lambda will add the necessary permissions for AWS IoT to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).

[Cancel](#) [Add](#)

- t) Afterwards, on the top right-hand corner, click on ‘Save’ to save the configuration. Once successful, you should see the following. Then, click on ‘iotca2’ and scroll down.



- u) Under ‘Code entry type’, select ‘Upload a .ZIP file’ and click on ‘Upload’.

Function code [Info](#)

Code entry type

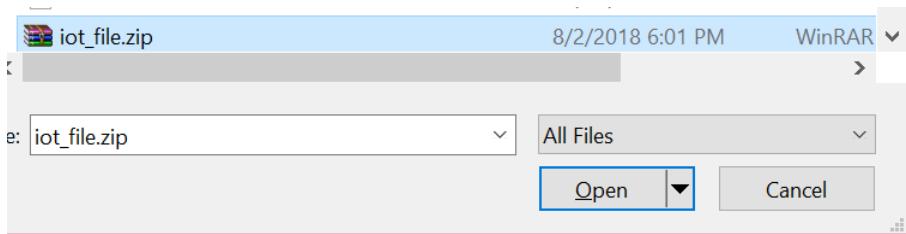
Runtime

Handler [Info](#)
lambda_function.lambda_handler

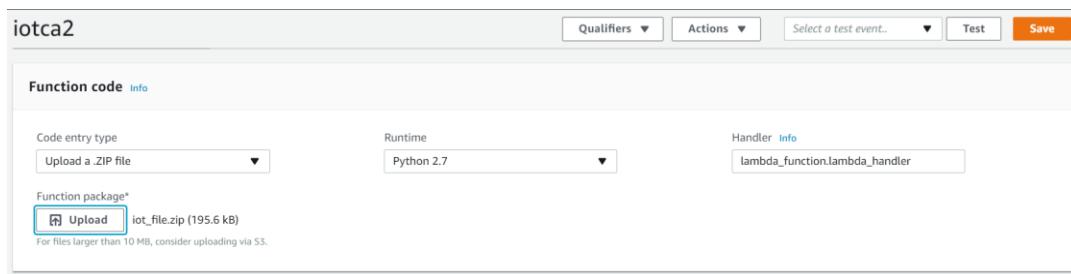
Function package*

For files larger than 10 MB, consider uploading via S3.

- v) ‘iot_file.zip’ can be downloaded under the ‘resources’ folder in GitHub. In your computer, search for the file ‘iot_file.zip’ and click ‘Open’.



- w) Once uploaded, on the top-right hand corner, click on ‘Save’.



Section 8

Configure AWS IoT Rules

A. Create an AWS Role

- a) Create a file `~/assignment/iot-role-trust.json` and add the following

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "iot.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

- b) Change directory to `~/assignment` and run the following command:

```
aws iam create-role --role-name my-iot-role --assume-  
role-policy-document file://iot-role-trust.json
```

B. Create an AWS Policy

- a) Create a file ~/assignment/iot-policy.json and add the following

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
  
            "Effect": "Allow",  
            "Action": "lambda:*",  
            "Resource": "arn:aws:lambda:us-west-2:460889015060:function:/*"  
        },  
        {  
  
            "Effect": "Allow",  
            "Action": "iot:*",  
            "Resource": "*"  
        }  
    ]  
}
```

- b) Change directory to ~/assignment and run the following command:

```
aws iam put-role-policy --role-name my-iot-role --policy-name iot-policy --policy-document file://iot-policy.json
```

Section 9

The Subscriber/Server

A. Write the Code for server.py

- a) Create a Python script `server.py` with the following code:

```
sudo nano ~/assignment/server.py
```

- b) Enter the following code into `server.py`:

```
#Import all the libraries here
import gevent
import gevent.monkey
from gevent.pywsgi import WSGIServer
import datetime
import pymysql
import signal
import multiprocessing
from time import sleep
import os
import time
import json
from flask import jsonify
from gpiozero import LED
import sys
import threading
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from flask import Flask, request, Response, render_template
import RPi.GPIO as GPIO
import MFRC522
from flask import Flask, redirect, url_for
import picamera
import os

gevent.monkey.patch_all()

def getdatetime():
    return time.strftime('%Y-%m-%d %H:%M:%S')

led_status = None
def light_status(client, userdata, message):
    global led_status
    led_status = json.loads(message.payload)
    print led_status

def mqtt_connect():
    my_rpi = AWSIoTMQTTClient("subscriber_led")

    host = "a7gx67jy14o4h.iot.us-west-2.amazonaws.com"
    rootCAPath = "rootca.pem"
    certificatePath = "certificate.pem.crt"
    privateKeyPath = "private.pem.key"

    my_rpi.configureEndpoint(host, 8883)
    my_rpi.configureCredentials(rootCAPath, privateKeyPath,
                                certificatePath)
```

```

my_rpi.configureOfflinePublishQueueing(-1)
my_rpi.configureDrainingFrequency(2)
my_rpi.configureConnectDisconnectTimeout(10)
my_rpi.configureMQTTOperationTimeout(5)

my_rpi.connect()

#Subscribe
my_rpi.subscribe("led/status", 1, light_status)

def dbconnect():
    try:
        db = pymysql.connect(host = "iotca2.cmwnxcwvcz4v.us-west-
2.rds.amazonaws.com", user="admin",passwd="!QWER4321", port=3306,
db="iotca2")
        print("Successfully connected to database")
        return db
    except:
        print("Error connecting to MySQL database")

def getRFID():
    #Create an object of the class MFRC522
    mfrc522 = MFRC522.MFRC522()

    #This loop keeps checking for chips.
    #If one is near it will get the UID
    try:
        while True:
            #Scan for cards
            (status,TagType)=
            mfrc522.MFRC522_Request(mfrc522.PICC_REQIDL)

            #If a card is found
            if status == mfrc522.MI_OK:

                #Get the UID of the card
                (status,uid) = mfrc522.MFRC522_Anticoll()
                print uid
                return uid
    except:
        GPIO.cleanup()

def createNewUser():
    uid = getRFID()

def showRFID():
    db = dbconnect()
    curs = db.cursor()

    try:
        results = []
        curs.execute("SELECT id, date_time, name from rfid_accesslog L,
rfid_access A WHERE L.rfid_uid=A.rfid_uid order by L.id desc")
        data = curs.fetchall() #get all the results in database
        for row in data:
            results.append(row)
        curs.close()
        db.close()
        return results
    except MySQLdb.Error as e:
        print e

```

```

        except KeyboardInterrupt:
            curs.close()
            db.close()

app = Flask(__name__)

#Start: Declaring app route here [after app=Flask(__name__)]
@app.route("/")
def index():
    results = showRFID()
    msg = ""
    try:
        msg = json.loads(request.args['msg'])
        msg = msg[msg]
    except:
        pass
    templateData = {
        'results' : results,
        'msg' : msg
    }
    return render_template('index.html', **templateData)

@app.route("/readRFID")
def readRFID():
    uid = getRFID()
    return jsonify(uid=uid)

@app.route("/addUser", methods=['POST'])
def addUser():
    if request.method == 'POST':
        uid = request.form['uid']
        name = request.form['name']
        uid = uid.replace(',', ' ', ' ')
        uid = '['+uid+']'
        try:
            db = dbconnect()
            curs = db.cursor()
            sql = "INSERT INTO rfid_access (rfid_uid, name) VALUES (%s, %s)".format(uid, name)
            curs.execute(sql)
            msg = json.dumps({"msg": "User has been added."})
            db.commit()
        except pymysql.IntegrityError:
            msg = json.dumps({"msg": "Failed to add user. Please try again."})
        curs.close()
        db.close()

    return redirect(url_for('.index', msg=msg))

@app.route("/newUser")
def newUser():
    return render_template('newUser.html')

# [LED(18), LED(21), LED(19), LED(13)] 0-3
@app.route("/readLED/<LED>")
def readLED(LED):
    return jsonify(response=led_status[LED])

@app.route("/writeLED/<LED>/<status>")
def writeLED(LED, status):

```

```
led_data = {"led":LED, "status":status}

my_rpi = AWSIoTMQTTClient("publisher_led")

host = "a7gx67jy14o4h.iot.us-west-2.amazonaws.com"
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

my_rpi.configureEndpoint(host, 8883)
my_rpi.configureCredentials(rootCAPath, privateKeyPath,
certificatePath)

my_rpi.configureOfflinePublishQueueing(-1)
my_rpi.configureDrainingFrequency(2)
my_rpi.configureConnectDisconnectTimeout(10)
my_rpi.configureMQTTOperationTimeout(5)

my_rpi.connect()

my_rpi.publish("led/control", json.dumps(led_data), 1)

return "Error removals R' US"

@app.route("/showImage/<access_id>")
def showImage(access_id):
    sql = "SELECT date_time, name from rfid_accesslog L, rfid_access A
WHERE L.rfid_uid=A.rfid_uid AND L.id = {}".format(access_id)

    db = dbconnect()
    curs = db.cursor()
    curs.execute(sql)

    result = curs.fetchone()

    curs.close()
    db.close()

    templateData = {
        'datetime' : result[0],
        'name': result[1]
    }
    return render_template('image.html', **templateData)

@app.route("/MQTTGraphValues")
def MQTTGraphValues():
    graphValues = [[],[],[]]
    sql = "SELECT date_time, temperature, humidity FROM temp_humidity
ORDER BY date_time DESC LIMIT 10"

    db = dbconnect()
    curs = db.cursor()
    curs.execute(sql)

    curs.close()
    db.close()

    for (datetime, temperature, humidity) in curs:
        #graphValues[0].append(datetime)
        graphValues[0].insert(0, datetime)
        #graphValues[1].append(temperature)
```

```
graphValues[1].insert(0, temperature)
#graphValues[2].append(humidity)
graphValues[2].insert(0, humidity)

return jsonify(graphdata=graphValues)

if __name__ == '__main__':
    mqtt_connect()
    try:
        print "Starting Server"
        http_server = WSGIServer(('0.0.0.0', 8010), app)
        app.debug = True
        http_server.serve_forever()
    except Exception as e:
        print(e)
    except KeyboardInterrupt:
        print "Terminate Server"
```

B. Write the Code for index.html

- a) Create a HTML file index.html with the following code:

```
sudo nano ~/assignment/templates/index.html
```

- b) Enter the following code into index.html:

```
<link rel="stylesheet" href="{{ url_for('static', filename='stylesheet.css') }}">
<!DOCTYPE html>

<head>
    <title>My Smart Home</title>

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js">
</script>

    <!-- Tell the browser to be responsive to screen width -->
    <meta content="width=device-width, initial-scale=1, maximum-scale=1,
user-scalable=no" name="viewport">

    <!-- Bootstrap 3.3.7 -->
    <link rel="stylesheet"
href="/static/bower_components/bootstrap/dist/css/bootstrap.min.css">

    <!-- Theme style -->
    <link rel="stylesheet" href="/static/dist/css/AdminLTE.min.css">

    <!-- AdminLTE Skins. Choose a skin from the css/skins folder instead
of downloading all of them to reduce the load. -->
    <link rel="stylesheet" href="/static/dist/css/skins/_all-
skins.min.css">

    <script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

    <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

    <!-- Google Font -->
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,6
00,700,300italic,400italic,600italic">

    <link rel="stylesheet"
href="/static/bower_components/DataTables/datatables.min.css">

    <script
src="/static/bower_components/DataTables/datatables.min.js"></script>

    <script type="text/javascript"
src="/static/dist/js/Chart.bundle.min.js"></script>

    <script type="text/javascript"
src="/static/dist/js/moment.js"></script>

    <script>
```

```
function turn_on_led(led, state){  
    $.ajax({  
        url: "/writeLED/" + led + "/" + state  
    })  
}  
  
function rfid() {  
    $.ajax({  
        url: "writeRFID"  
    })  
}  
  
function ledupdate(){  
    $.ajax({  
        url: "/readLED/0",  
        success: function(result) {  
            if (result.response) {  
                $('input[id="red"]').prop('checked', true);  
            }  
        }  
    })  
  
    $.ajax({  
        url: "/readLED/1",  
        success: function(result) {  
            if (result.response) {  
                $('input[id="green"]').prop('checked', true);  
            }  
        }  
    })  
  
    $.ajax({  
        url: "/readLED/2",  
        success: function(result) {  
            if (result.response) {  
                $('input[id="yellow"]').prop('checked', true);  
            }  
        }  
    })  
  
    $.ajax({  
        url: "/readLED/3",  
        success: function(result) {  
            if (result.response) {  
                $('input[id="redtwo"]').prop('checked', true);  
            }  
        }  
    })  
}  
  
document.addEventListener('DOMContentLoaded', function() {  
    var checkbox = document.querySelector('input[id="red"]');  
  
    checkbox.addEventListener('change', function() {  
        if (checkbox.checked) {  
            turn_on_led(0, "ON")  
        } else {  
            turn_on_led(0, "OFF")  
        }  
    });  
});
```

```
});  
  
document.addEventListener('DOMContentLoaded', function() {  
    var checkbox = document.querySelector('input[id="green"]');  
  
    checkbox.addEventListener('change', function() {  
        if (checkbox.checked) {  
            turn_on_led(1, "ON")  
        } else {  
            turn_on_led(1, "OFF")  
        }  
    });  
});  
  
document.addEventListener('DOMContentLoaded', function() {  
    var checkbox = document.querySelector('input[id="yellow"]');  
  
    checkbox.addEventListener('change', function() {  
        if (checkbox.checked) {  
            turn_on_led(2, "ON")  
        } else {  
            turn_on_led(2, "OFF")  
        }  
    });  
});  
  
document.addEventListener('DOMContentLoaded', function() {  
    var checkbox = document.querySelector('input[id="redtwo"]');  
  
    checkbox.addEventListener('change', function() {  
        if (checkbox.checked) {  
            turn_on_led(3, "ON")  
        } else {  
            turn_on_led(3, "OFF")  
        }  
    });  
});  
  
$(document).ready(function() {  
  
    $('#example1').DataTable({  
        "pageLength": 7,  
        "dom" : "tp"  
    });  
  
    var tempchart = document.getElementById('temp-chart').getContext('2d');  
    var humiditychart = document.getElementById('humidity-chart').getContext('2d');  
  
    var config = {  
        type: 'line',  
        data: {  
            labels: [],  
            datasets: [{  
                label: 'Temperature',  
                data: [],  
                backgroundColor: "rgba(255,0,0,0.4)",  
                borderColor: "rgba(255,0,0,0.4)",  
                fill: false  
            }]  
    }  
});
```

```
        },
        options: {
            responsive: true,
            title: {
                display: true,
                text: 'Temperature graph'
            },
            scales: {
                yAxes : [
                    ticks : {
                        max : 35.0,
                        min : 15.0
                    }
                ]
            }
        }
    }

var config2 = {
    type: 'line',
    data: {
        labels: [],
        datasets: [
            label: 'Humidity',
            data: [],
            backgroundColor: "rgba(0,255,0,0.4)",
            borderColor: "rgba(0,255,0,0.4)",
            fill: false
        ]
    },
    options: {
        responsive: true,
        title: {
            display: true,
            text: 'Humidity Graph'
        },
        scales: {
            yAxes : [
                ticks : {
                    max : 80.0,
                    min : 20.0
                }
            ]
        }
    }
}

var tempG = new Chart(tempchart, config);
var humidityG = new Chart(humiditychart, config2);

function dataupdate() {
    var axisLabel = [];

    $.ajax({
        url: "MQTTGraphValues",
        success: function(result) {
            var datalist = result.graphdata;

            for (var i = 0; i < datalist[0].length; i++) {
```

```

        axisLabel.push(moment(datalist[0][i]).format('h:mm:ss
a'));
    }

    tempG.data.datasets.forEach((dataset) => {
        dataset.data = datalist[1];
    });

    humidityG.data.datasets.forEach((dataset) => {
        dataset.data = datalist[2];
    });

    tempG.data.labels = axisLabel;
    humidityG.data.labels = axisLabel;

    $('#temp').text(datalist[1][datalist[1].length - 1]);
    $('#hum').text(datalist[2][datalist[2].length - 1]);
}
))

tempG.update()
humidityG.update()
}

// Reduce the values to increase update speed
setInterval(dataupdate, 5000);
//setInterval(ledupdate, 2500);
ledupdate()
});

</script>
<script>
$(document).ready(function() {


    alert( "{{ msg.msg }} " );


})
</script>
</head>

<body class="hold-transition skin-blue layout-top-nav">
<div class="wrapper">
    <header class="main-header">
        <nav class="navbar navbar-static-top">
            <a href="/" style="float: left;">
                
                <h1 id="smarthome">SMART HOME</h1>
            </a>
            <a href="newUser"><button style="float:right; color: black; margin-
right: 9%; margin-top: 2%; border: none; border-radius: 2px; padding:
3px; padding-left: 8px; padding-right: 8px; background-color:
#E8E8E8;">Add New User</button></a>
        </nav>
    </header>
    <!-- Full Width Column -->
    <div class="content-wrapper">
        <div class="container">
            <!-- Main content -->
            <section class="content">

```

```
<div class="row">
    <div class="col-lg-6">
        <div class="box box-primary">
            <div class="box-header with-border">
                <h3 class="box-title">Home Access History</h3>
            </div>

            <div class="box-body">
                <table id="example1" class="table table-bordered table-striped dataTable" data-page-length='6'>
                    <thead>
                        <tr>
                            <th style="width:40%">Date (YYYY:MM:DD)</th>
                            <th style="width:40%">Time (HH:mm:SS)</th>
                            <th style="width:20%">User</th>
                        </tr>
                    </thead>
                    <tbody>
                        {# for row in results %}
                        <tr>
                            <td>{{ row[1][0:10] }}</td>
                            <td>{{ row[1][11:13] }}:{{ row[1][14:16] }}:{{ row[1][17:19] }}</td>
                            <td><a href="/showImage/{{ row[0] }}">{{ row[2] }}</a></td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        <br />
    </div>

    </div>
    <div class="col-lg-2">
        <div class="box" id="control">
            <div class="box-header with-border">
                <h3 class="box-title">Bed Room</h3>
            </div>
            <br />
            
            <label class="switch">
                <input type="checkbox" id="red" />
                <span class="slider round"></span>
            </label>
        </div>
        <div class="box" id="control">
            <div class="box-header with-border">
                <h3 class="box-title">Living Room</h3>
            </div>
            <br />
            
            <label class="switch">
                <input type="checkbox" id="green" />
            </label>
        </div>
    </div>
</div>
```

```
        <span class="slider round"></span>
    </label>

    </div>
</div>
<div class="col-lg-2">
    <div class="box" id="control">
        <div class="box-header with-border">
            <h3 class="box-title">Toilet</h3>
        </div>
        <br />
        
        <label class="switch">
            <input type="checkbox" id="yellow" round"></span>
            <span class="slider round"></span>
        </label>

    </div>
    <div class="box" id="control">
        <div class="box-header with-border">
            <h3 class="box-title">Kitchen</h3>
        </div>
        <br />
        
        <label class="switch">
            <input type="checkbox" id="redtwo" round"></span>
            <span class="slider round"></span>
        </label>

    </div>
</div>
<div class="col-lg-2">
    <div class="box" id="control">
        <div class="box-header with-border">
            <h3 class="box-title">Temperature (°C)</h3>
        </div>
        <br />
        <h1 id="temp"></h1>
    </div>
    <div class="box" id="control">
        <div class="box-header with-border">
            <h3 class="box-title">Humidity (%)</h3>
        </div>
        <br />
        <h1 id="hum"></h1>
    </div>
</div>
<!-- Graphs -->
<div class="row">
    <div class="col-lg-6">
        <div class="box">
            <canvas id="temp-chart"></canvas>
        </div>
    </div>
```

```
</div>
<div class="col-lg-6">
    <div class="box">
        <canvas id="humidity-chart"></canvas>
    </div>
</div>
</section>
<!-- Bootstrap 3.3.7 -->
<script
src="/static/bower_components/bootstrap/dist/js/bootstrap.min.js"></scr
ipt>
<!-- SlimScroll -->
<script src="/static/bower_components/jquery-
slimscroll/jquery.slimscroll.min.js"></script>
<!-- FastClick -->
<script
src="/static/bower_components/fastclick/lib/fastclick.js"></script>
<!-- AdminLTE App -->
<script src="/static/dist/js/adminlte.min.js"></script>
</body>

</html>
```

C. Write the Code for image.html

- a) Create a HTML file image.html with the following code:

```
sudo nano ~/assignment/templates/image.html
```

- b) Enter the following code into image.html:

```
<link rel="stylesheet" href="{{ url_for('static', filename='stylesheet.css') }}">
<!DOCTYPE html>

<head>
<title>My Smart Home</title>


<meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" name="viewport">

<!-- Bootstrap 3.3.7 --&gt;
&lt;link rel="stylesheet"
href="/static/bower_components/bootstrap/dist/css/bootstrap.min.css"&gt;

<!-- Theme style --&gt;
&lt;link rel="stylesheet" href="/static/dist/css/AdminLTE.min.css"&gt;

<!-- AdminLTE Skins. Choose a skin from the css/skins folder instead of
downloading all of them to reduce the load. --&gt;
&lt;link rel="stylesheet" href="/static/dist/css/skins/_all-skins.min.css"&gt;

&lt;script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"&gt;&lt;/script&gt;

&lt;script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"&gt;&lt;/script&gt;

<!-- Google Font --&gt;
&lt;link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600,700,300italic,400italic,600italic"&gt;

&lt;link type="text/css" rel="stylesheet" href="css/main.css"&gt;

&lt;script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"&gt;&lt;/script&gt;

&lt;script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.js"&gt;&lt;/script&gt;

&lt;/head&gt;

&lt;body class="hold-transition skin-blue layout-top-nav"&gt;
&lt;div class="wrapper"&gt;
    &lt;header class="main-header"&gt;
        &lt;nav class="navbar navbar-static-top"&gt;
            &lt;a href="/"&gt;
                &lt;img src="/static/home.jpg" alt="Home" id="home" width="50px" height="50px"&gt;</pre>
```

```
                <h1 id="smarthome">SMART HOME</h1>
            </a>
        </nav>
    </header>
    <!-- Full Width Column -->
    <div class="content-wrapper">
        <div class="container">
            <!-- Main content -->
            <section class="content">
                <div class="col-lg-12">
                    <div class="box box-primary">
                        <div class="box-header with-border" id="imageTitle">
                            <h3 class="box-title">User: <strong>{{name}}</strong></h3>
                            <h3 class="box-title">Photo Taken on <strong>{{datetime[0:10]}} {{datetime[11:13]}}:{{datetime[14:16]}}:{{datetime[17:19]}}</strong></h3>
                            <br />
                            
                            <br />
                            <a href="/"><h4 id="backHome">[ Back to Home Page ]</h4></a>
                            <br />
                        </div>
                    </div>
                </div>
            </section>
        </div>
    </div>
</body>
</html>
```

D. Write the Code for newUser.html

- a) Create a HTML file newUser.html with the following code:

```
sudo nano ~/assignment/templates(newUser.html)
```

- b) Enter the following code into **newUser.html**:

```
<link rel="stylesheet" href="{{ url_for('static', filename='stylesheet.css') }}">
<!DOCTYPE html>

<head>
<title>My Smart Home</title>


<meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" name="viewport">

<!-- Bootstrap 3.3.7 --&gt;
&lt;link rel="stylesheet"
href="/static/bower_components/bootstrap/dist/css/bootstrap.min.css"&gt;

<!-- Theme style --&gt;
&lt;link rel="stylesheet" href="/static/dist/css/AdminLTE.min.css"&gt;

<!-- AdminLTE Skins. Choose a skin from the css/skins folder instead of
downloading all of them to reduce the load. --&gt;
&lt;link rel="stylesheet" href="/static/dist/css/skins/_all-skins.min.css"&gt;

&lt;script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"&gt;&lt;/script&gt;

&lt;script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"&gt;&lt;/script&gt;

<!-- Google Font --&gt;
&lt;link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600,700,300italic,400italic,600italic"&gt;

&lt;link type="text/css" rel="stylesheet" href="css/main.css"&gt;
&lt;script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"&gt;
&lt;/script&gt;
&lt;script
src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"&gt;&lt;/script&gt;
&lt;style&gt;
.center {
    display: block;
    margin-left: auto;
    margin-right: auto;
    width: 50%;
}

.close {
    color: #aaa;
    float: right;
}</pre>
```

```

        font-size: 28px;
        font-weight: bold;
    }

.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
}
.modal-content {
    background-color: #fefefe;
    margin: 15% auto; /* 15% from the top and centered */
    padding: 20px;
    border: 1px solid #888;
    width: 20%; /* Could be more or less, depending on screen size */
    height: 30%;
    text-align: center;
}
</style>
<script>
$(document).ready(function() {
    // Modal
    var span = document.getElementsByClassName("close")[0];
    var modal = document.getElementById('myModal');
    span.onclick = function() {
        $('#myModal').modal('hide');
    }
    window.onclick = function(event) {
        if (event.target == modal) {
            $('#myModal').modal('hide');
        }
    }
    $.get( "readRFID", function( data ) {
        $('#myModal').modal('show');
        alert( "The UID detected is: " + data.uid );
        document.getElementById("uid").value = data.uid;
    });
})
</script>
</head>

<body class="hold-transition skin-blue layout-top-nav">
<div class="wrapper">
    <header class="main-header">
        <nav class="navbar navbar-static-top">
            <a href="/">
                
                <h1 id="smarthome">SMART HOME</h1>
            </a>
        </nav>
    </header>
    <!-- Full Width Column -->
    <div class="content-wrapper">
        <div class="container">
            <!-- Main content -->
            <section class="content">
                <div class="col-lg-12">
                    <div class="box box-primary">

```

```
<div class="box-
header with-border" id="imageTitle">
    <h3 class="box-
title">Scan your Card</strong></h3>
    <br />
    
    <br />
    <a href="/"><h4 id="backHome">[ Back to Home Page ]</h4></a>
    <br />
    </div>
</div>

<div id="myModal" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <form method="POST" action="addUser">
            <br />
            Username:<br><br>
            <input type="text" name="name">
            <br /> <br />
            <input type="hidden" id="uid" name="uid" value="">
            <input type="submit" value="Submit">
        </form>
    </div>
</div>
</body>
</html>
```

E. Expected Outcome

- a) Change directory to ~/assignment:

```
cd ~/assignment
```

- b) Run the following to start server.py:

```
python ./server.py
```

- c) In your terminal, you should see the following

```
pi@joylaw:~/Desktop/iotiot $ python ./server.py
Starting Server
Successfully connected to database
169.254.103.49 - - [2018-02-20 08:19:33] "GET /MQTTGraphValues HTTP/1.1" 200 591
1.241310
Successfully connected to database
169.254.103.49 - - [2018-02-20 08:19:38] "GET /MQTTGraphValues HTTP/1.1" 200 591
1.117392
Successfully connected to database
169.254.103.49 - - [2018-02-20 08:19:43] "GET /MQTTGraphValues HTTP/1.1" 200 591
1.127225
Successfully connected to database
169.254.103.49 - - [2018-02-20 08:19:46] "GET / HTTP/1.1" 200 69943 1.437121
```

Section 10

The Publisher

F. Write the Code for pub.py

- a) Create a Python script pub.py with the following code:

```
sudo nano ~/assignment/pub.py
```

- b) Enter the following code into pub.py:

```
# Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from time import sleep
import time, datetime
import json
import Adafruit_DHT
import multiprocessing
from signal import pause
from gpiozero import LED
import MFRC522
import re
import boto3
import botocore
import RPi.GPIO as GPIO
from picamera import PiCamera
import os
import pymysql

# Returns tuple: (Humidity, Temperature)
def gethumiditytemp():
    return Adafruit_DHT.read_retry(11, 4)

def getdatetime():
    return time.strftime('%Y-%m-%d %H:%M:%S')

s3 = boto3.resource('s3')

def prepareImage():
    camera = PiCamera()
    datetime = time.strftime('%Y-%m-%d_%H-%M-%S') #Get the current date and time
    full_path = os.getcwd() + "/" + datetime + ".jpg"
    file_name = datetime + '.jpg'
    camera.capture(full_path)
    camera.close()
    return datetime, full_path, file_name

def setS3():
    bucket_name = 'accesssmarthome' # replace with your own unique bucket name
    exists = True
    try:
        s3.meta.client.head_bucket(Bucket=bucket_name)
    except botocore.exceptions.ClientError as e:
        error_code = int(e.response['Error']['Code'])
        if error_code == 404:
            exists = False
```

```

if exists == False:

    s3.create_bucket(Bucket=bucket_name, CreateBucketConfiguration={'LocationConstraint': 'us-west-2'})
    return bucket_name

def rfid():
    mfrc522 = MFRC522.MFRC522()
    try:
        while True:
            (status,TagType)=
            mfrc522.MFRC522_Request(mfrc522.PICC_REQIDL)
            if status == mfrc522.MI_OK:
                (status,uid) = mfrc522.MFRC522_Anticoll()
                datetime, full_path, file_name = prepareImage()
                bucket_name = sets3()
                db = pymysql.connect(host =
"iotca2.cmwnxcwvcz4v.us-west-2.rds.amazonaws.com",
user="admin",passwd="!QWER4321", port=3306, db="iotca2")
                s3.Object(bucket_name,
file_name).put(Body=open(full_path, 'rb'))
                try:
                    with db.cursor() as cur:
                        cur.execute("INSERT into rfid_accesslog
(date_time, rfid_uid) VALUES ('{0}', '{1}').format(datetime,uid)")
                        db.commit()
                except Exception, e:
                    print (e)
                db.close()
                os.remove(full_path)
                sleep(5)
    except:
        GPIO.cleanup()

LEDS = [LED(18), LED(21), LED(19), LED(13)]

def light_controller(client, userdata, message):
    data = json.loads(message.payload)

    print(data)

    if data is not None:
        led = int(data["led"])
        print led
        if led <= len(LEDS):
            if data["status"] == "ON":
                LEDS[led].on()
            elif data["status"] == "OFF":
                LEDS[led].off()
            else:
                print("INVALID COMMAND")

def light_status():
    led_status = {}
    for led in LEDS:
        led_status.update({LEDS.index(led):led.is_lit})
    return led_status

def mqtt_connect():

    host = "a7gx67jy14o4h.iot.us-west-2.amazonaws.com"

```

```

rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

my_rpi = AWSIoTMQTTClient("publisher")
my_rpi.configureEndpoint(host, 8883)
my_rpi.configureCredentials(rootCAPath, privateKeyPath,
certificatePath)

my_rpi.configureOfflinePublishQueueing(-1)
my_rpi.configureDrainingFrequency(2)
my_rpi.configureConnectDisconnectTimeout(10)
my_rpi.configureMQTTOperationTimeout(5)

my_rpi.connect()

#Subscribe
my_rpi.subscribe("led/control", 1, light_controller)

while True:
    HT = gethumiditytemp()
    iot_data = {"datetime":str(getdatetime()), "temperature":str(HT[1]), "humidity":str(HT[0])}
    my_rpi.publish("sensors/temp_humid", json.dumps(iot_data), 1)
    print("published ", iot_data)
    light_stat = light_status()
    my_rpi.publish("led/status", json.dumps(light_stat), 1)
    print("published ", light_stat)
    sleep(5)

if __name__ == '__main__':
    multiprocessing.Process(target= mqtt_connect).start()
    multiprocessing.Process(target= rfid).start()
    pause()

```

G. Expected Outcome

- a) Change directory to ~/assignment:

```
cd ~/assignment
```

- b) Run the following to start server.py:

```
python ./pub.py
```

- c) In your terminal, you should see the following:

```

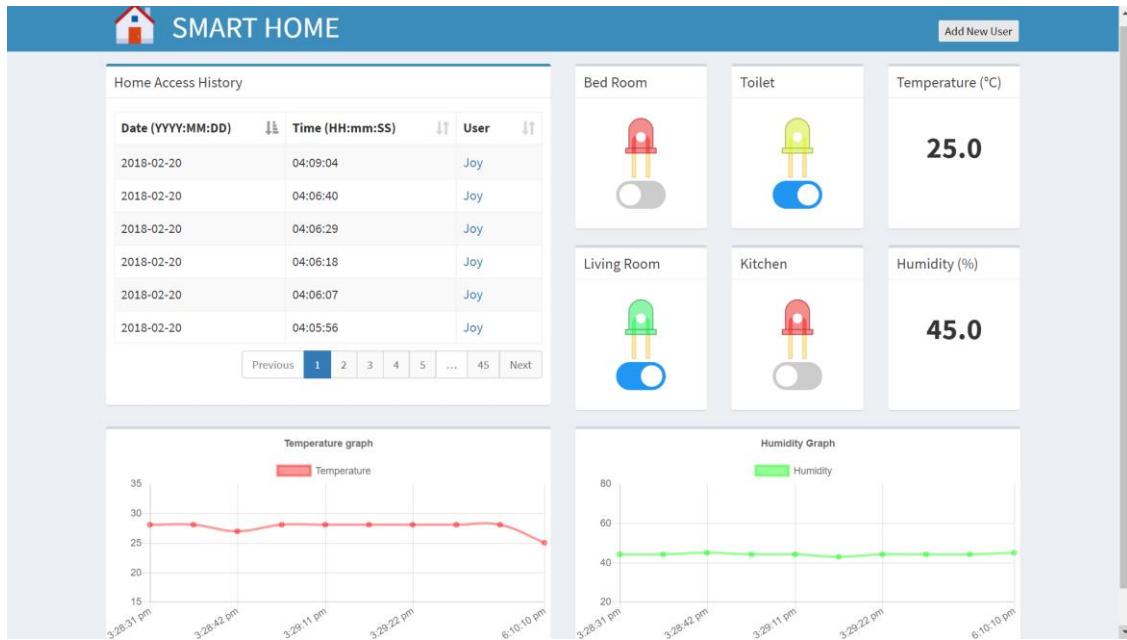
pi@joylaw:~/Desktop/iotiot $ python ./pub.py
('published ', {'humidity': '29.0', 'temperature': '27.0', 'datetime': '2018-02-20 08:21:49'})
('published ', {0: False, 1: False, 2: False, 3: False})
('published ', {'humidity': '29.0', 'temperature': '27.0', 'datetime': '2018-02-20 08:21:57'})
('published ', {0: False, 1: False, 2: False, 3: False})
('published ', {'humidity': '28.0', 'temperature': '27.0', 'datetime': '2018-02-20 08:22:05'})
('published ', {0: False, 1: False, 2: False, 3: False})

```

H. Final Outcome (Dashboard)

- a) On the Raspberry Pi that is running the server.py, using a web browser go to the following link where 169.254.134.247 is the IP address of your Raspberry Pi and 8010 is the port set in your server.py

<http://169.254.134.247:8010/>



Section 11

Task List

A. Task List

Task	Name
All RFID, Camera functions, documentation	Joy Law
All LED, temperature & humidity functions	Sean Phang

-- End of CA2 Step-by-step tutorial --