SCHOOL OF COMPUTING (SOC)

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:	23/02/2020
Prepared for:	Ms Dora Chua
Class:	DISM/FT/3A/41

Submitted by:

Student ID	**Name**
1727191	Low Shu Ting
1703148	Poh Haw Jin

Table of Contents

Section 1
Overview of project

A. Tutorial Links

Youtube	https://youtu.be/uWQ59Pcpb20
Public tutorial link	https://github.com/shutingx/MoniCam.git

B. Introduction

This application is a home security monitoring system, which consists of a Pi Camera that is motion activated using a motion sensor, and also a RFID reader to 'allow access' through the use of RFID cards. When an authenticated RFID is used, facial recognition will confirm that the current user is the card owner. On the other hand, when there is 3 unsuccessful attempts to 'get access' by using unknown RFID id cards, it will alert the user through telegram alert with a picture and sound the alarm for 30 seconds. Moreover, when the light intensity falls below a certain level, the LED will light up to assist doing facial recognition and taking pictures.

The user can use the web interface to sound alarm, see the status of the motion sensor, real time and historical data of motion sensed and attempts, and add/modify/delete users.

The owner can send various commands to the telegram bot such as getting it to take pictures, ring alarms, chat with guests who use the bot too and displaying text on the lcd display.

C. Final RPi and Arduino Set-Up

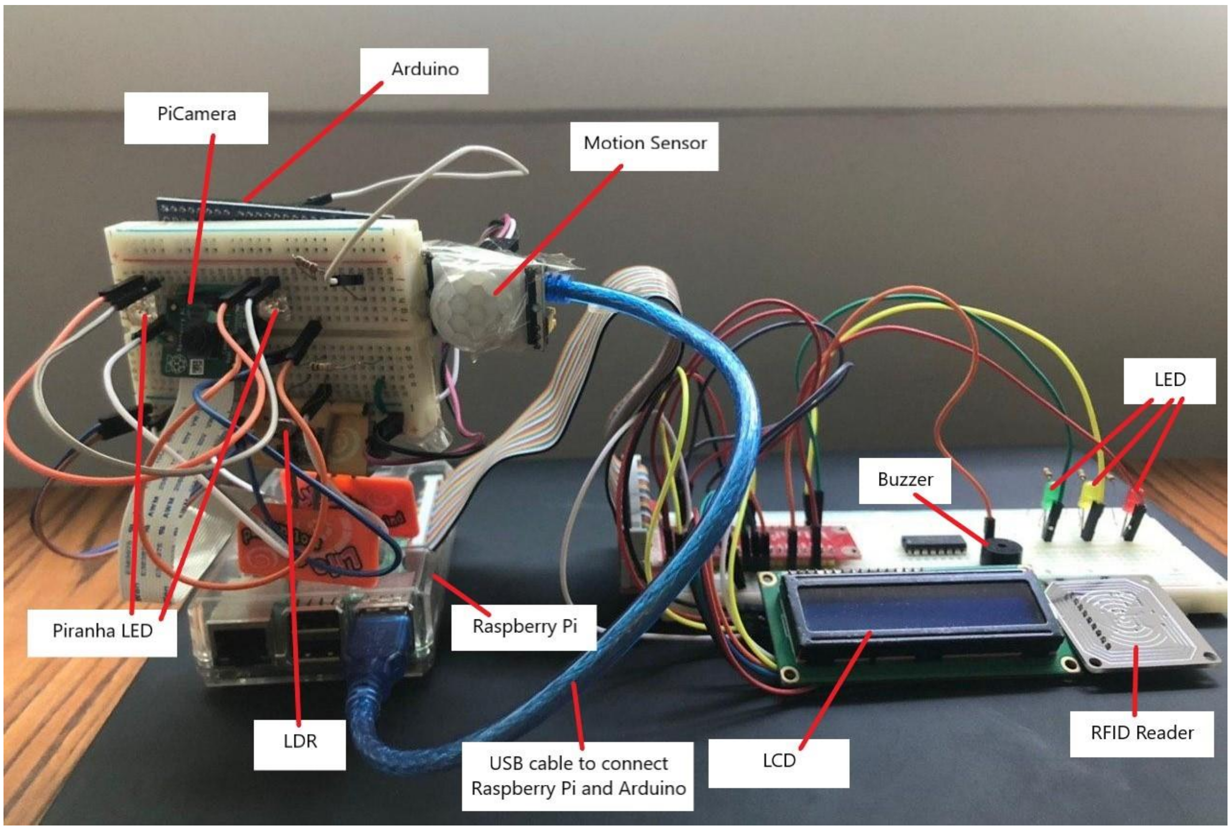


Figure 1: Full Set-Up of Hardware

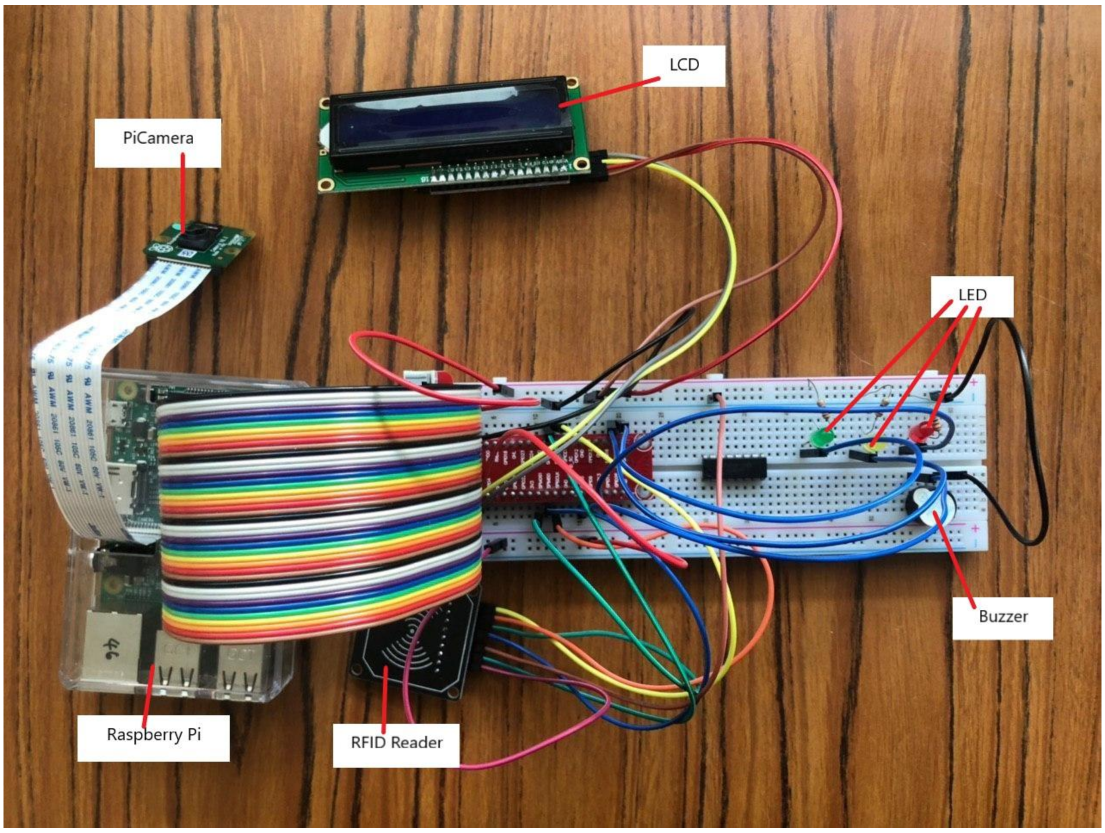


Figure 2: Close-Up of Raspberry Pi Set-Up

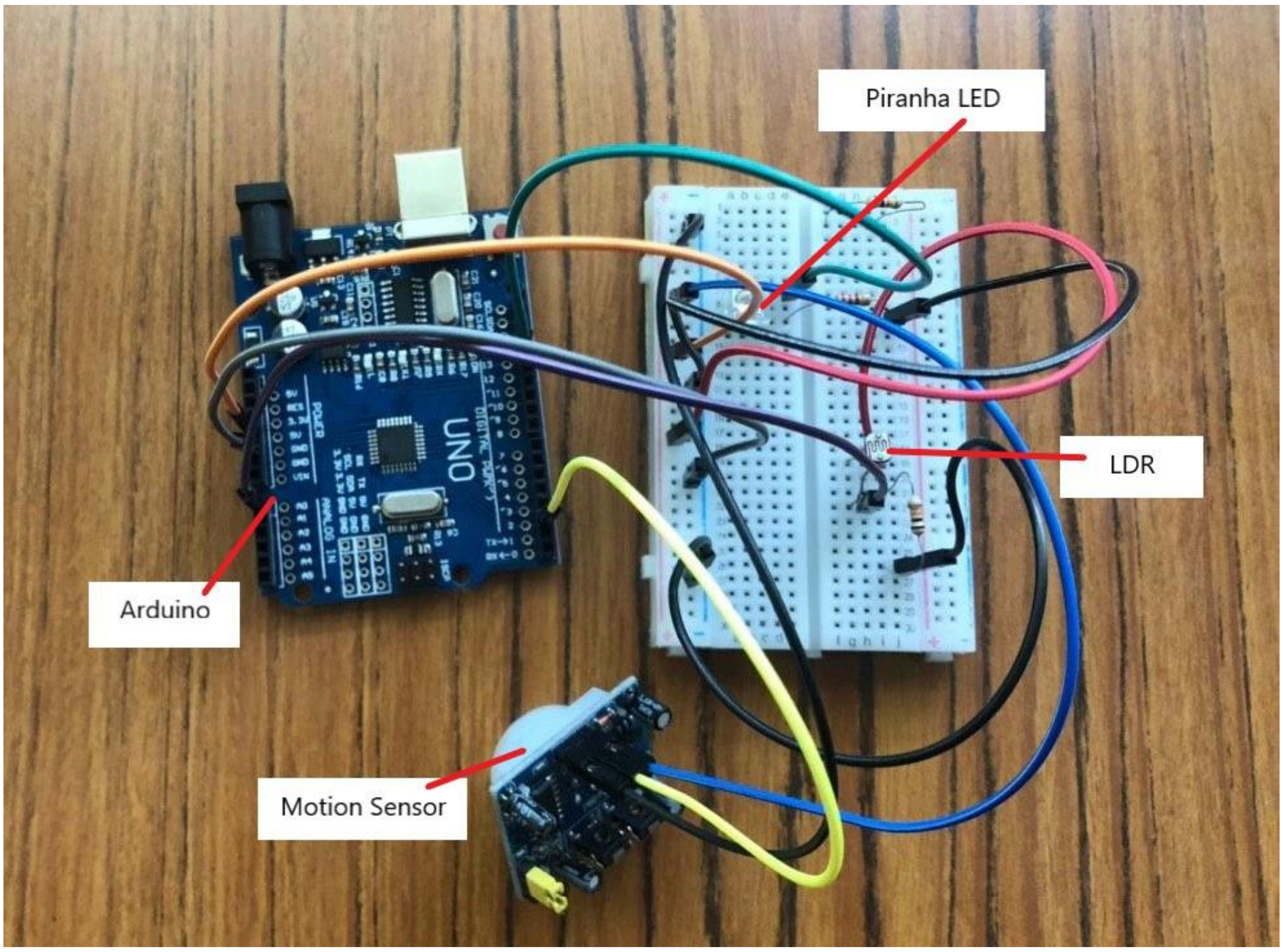


Figure 3: Close-Up of Arduino Set-Up

D. Web Interface

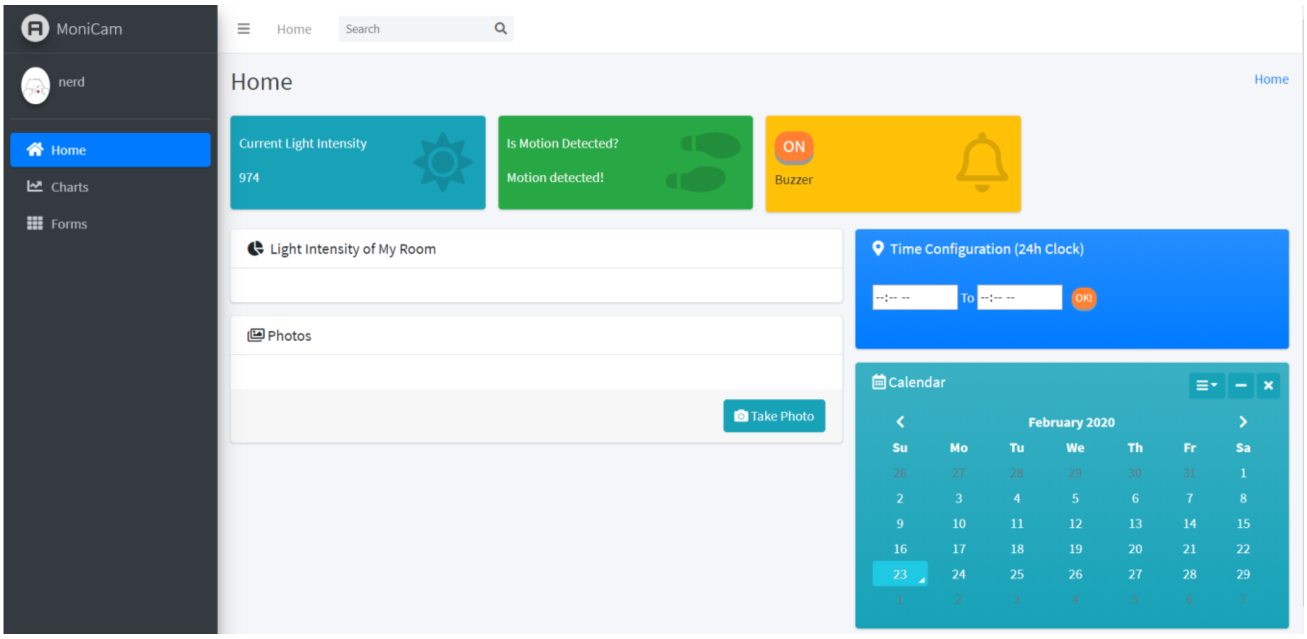
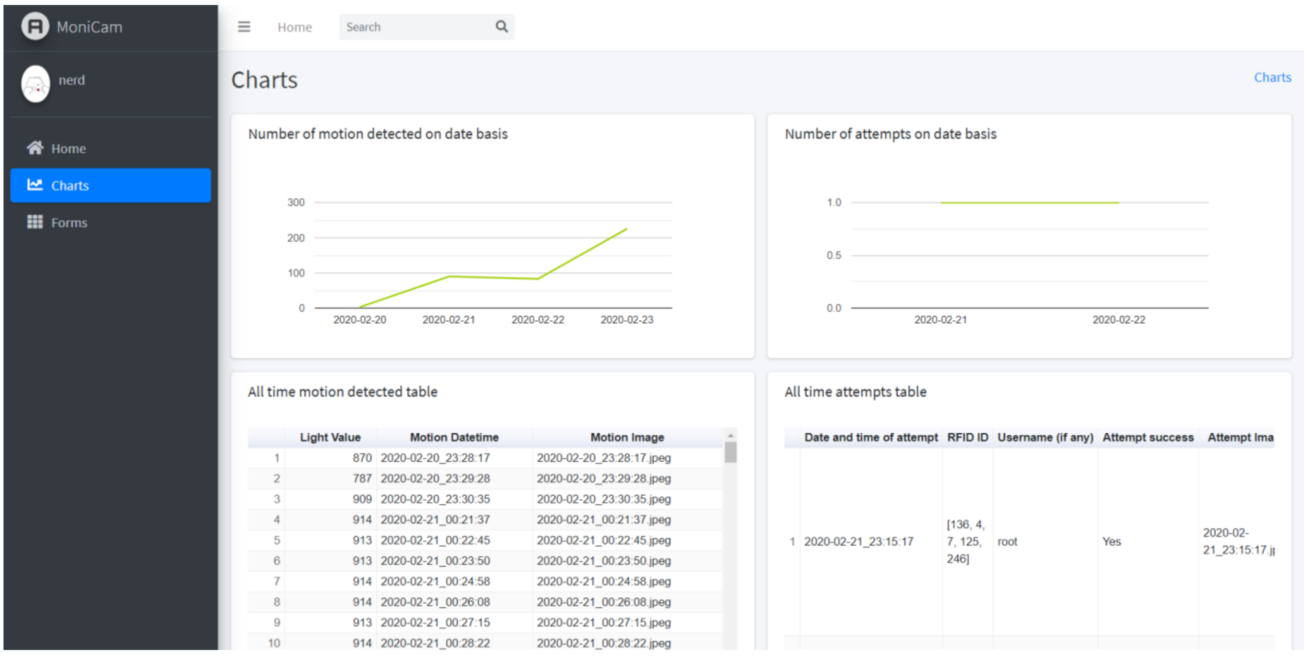


Figure 4: index.html



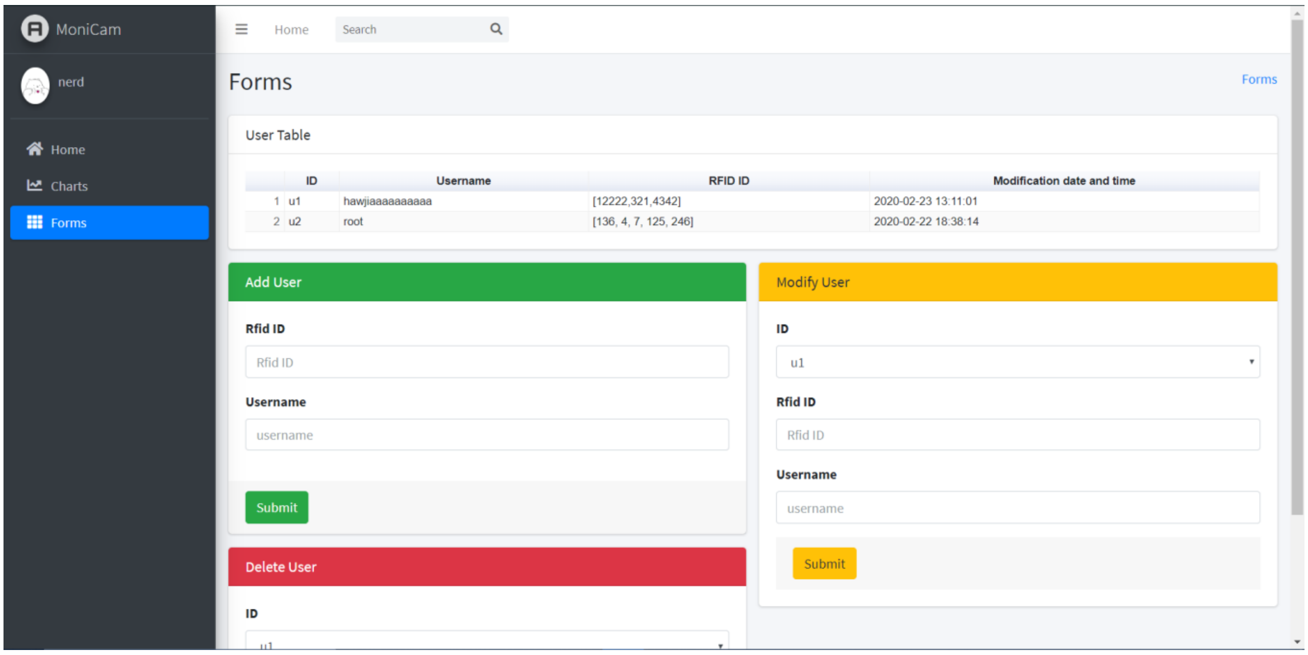
All time motion detected table

	Light Value	Motion Datetime	Motion Image
1	870	2020-02-20_23:28:17	2020-02-20_23:28:17.jpeg
2	787	2020-02-20_23:29:28	2020-02-20_23:29:28.jpeg
3	909	2020-02-20_23:30:35	2020-02-20_23:30:35.jpeg
4	914	2020-02-21_00:21:37	2020-02-21_00:21:37.jpeg
5	913	2020-02-21_00:22:45	2020-02-21_00:22:45.jpeg
6	913	2020-02-21_00:23:50	2020-02-21_00:23:50.jpeg
7	914	2020-02-21_00:24:58	2020-02-21_00:24:58.jpeg
8	914	2020-02-21_00:26:08	2020-02-21_00:26:08.jpeg
9	913	2020-02-21_00:27:15	2020-02-21_00:27:15.jpeg
10	914	2020-02-21_00:28:22	2020-02-21_00:28:22.jpeg

All time attempts table

	Date and time of attempt	RFID ID	Username (if any)	Attempt success	Attempt Ima
1	2020-02-21_23:15:17	[136, 4, 7, 125, 246]	root	Yes	2020-02-21_23:15:17.j

Figure 5: charts.html



User Table

ID		Username	RFID ID	Modification date and time
1	u1	hawjiaaaaaaaaaa	[12222,321,4342]	2020-02-23 13:11:01
2	u2	root	[136, 4, 7, 125, 246]	2020-02-22 18:38:14

Figure 6: forms.html

E. System architecture of MoniCam

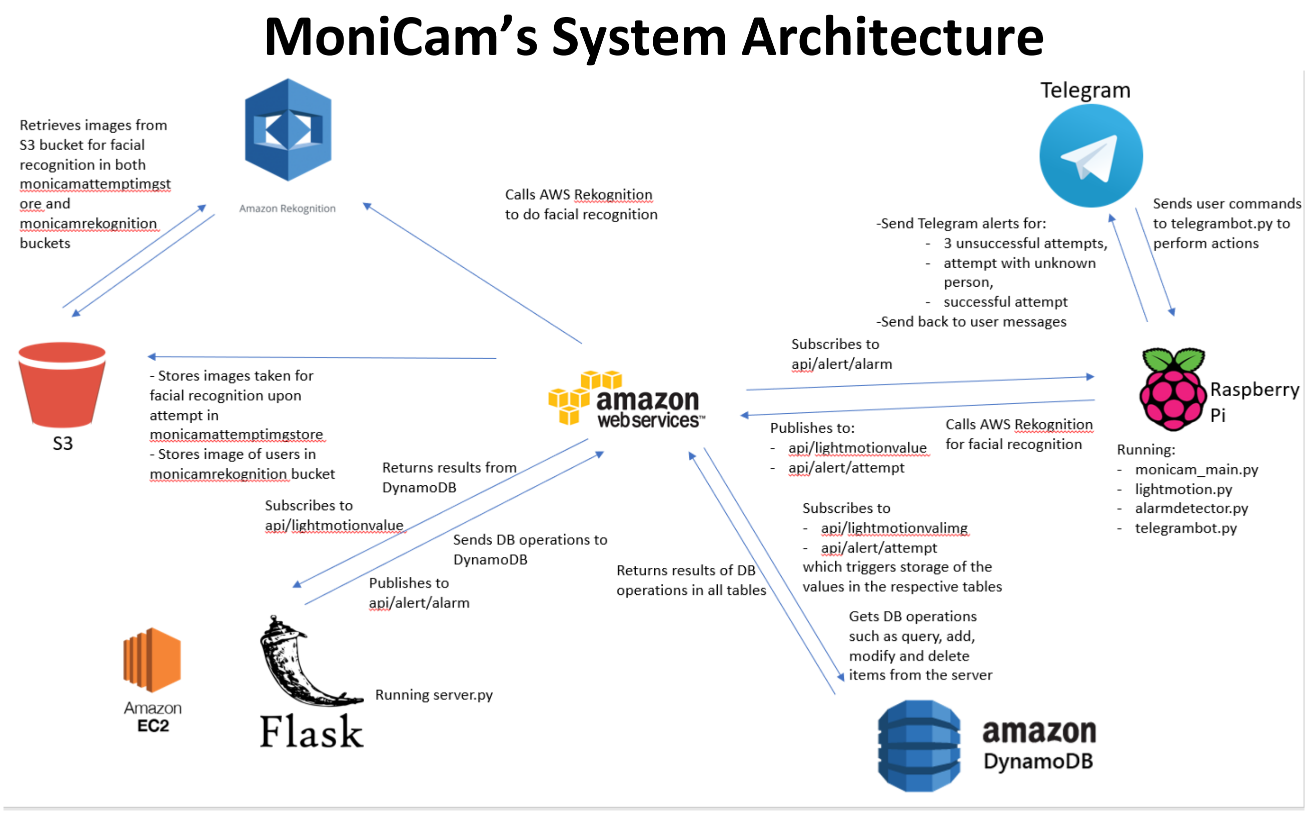

MoniCam's System Architecture
Retrieves images from S3 bucket for facial recognition in both monicamattemptimgstore and monicamrekognition buckets
Amazon Rekognition
Calls AWS Rekognition to do facial recognition
Telegram
-Send Telegram alerts for:
- 3 unsuccessful attempts,
- attempt with unknown person,
- successful attempt
-Send back to user messages
Sends user commands to telegrambot.py to perform actions
S3
- Stores images taken for facial recognition upon attempt in monicamattemptimgstore
- Stores image of users in monicamrekognition bucket
amazon web services
Subscribes to api/alert/alarm
Publishes to:
- api/lightmotionvalue
- api/alert/attempt
Calls AWS Rekognition for facial recognition
Raspberry Pi
Running:
- monicam_main.py
- lightmotion.py
- alarmdetector.py
- telegrambot.py
Returns results from DynamoDB
Subscribes to api/lightmotionvalue
Sends DB operations to DynamoDB
Publishes to api/alert/alarm
Returns results of DB operations in all tables
Subscribes to
- api/lightmotionvalimg
- api/alert/attempt
which triggers storage of the values in the respective tables
Gets DB operations such as query, add, modify and delete items from the server
Amazon EC2
Flask
Running server.py
amazon DynamoDB

Figure 7: Monicam's System Architecture

F. Evidence that we have met basic requirements

Requirement	Evidence
Used three sensors and actuators	Used Sensors: • Motion Sensor • LDR (Light Sensor) • RFID Reader • PiCamera Used Actuators: • LEDs • Piranha LEDs • LCD • Buzzer

Used MQTT	Our MQTT endpoint --> a1sihgkt17tgrd-ats.iot.us-east-1.amazonaws.com
	Example of data sent through MQTT :
	{
	"deviceid": deviceid,
	"motion_datetime": date_time,
	"lightvalue": light_val,
	"motionvalue": motion_sig,
	"motion_image": motion_img
	}

Stored data in cloud	Stored all text data in DynamoDB
Used cloud service	Use AWS Rekognition, hosted web server on EC2, S3 for storing images
Provide real-time sensor value / status	Show the real-time value of light sensor and motion sensor
Provide historical sensor value/ status	Show historical value of motion sensor
Control actuator	Placed button on webpage to control buzzer

G. Bonus features on top of basic requirements

1. ~~Log in system that zzzzz~~Used Arduino for light sensor (analog sensor) that checks the light intensity

2. Used Arduino for motion sensor (digital sensor) that checks the presence of motion

3. Telegram bot that:
   1. Can be a proxy to communicate with guests
   2. Send text to be displayed on lcd display
   3. Display real time picture
   4. Ring the alarm
   5. Get the chat id with the bot
   6. Get the current guest id when communicating with the guest
   7. Only owner can use certain commands guests cannot use
   8. Guests can request to communicate with the owner

4. Web interface that:
   1. Can add, modify and delete user entry
   2. Grouping of individual data into date or time range to allow easier understanding of the data and more meaningful data presentation

5. LCD display to provide information to guests

6. Facial Recognition is used to scan faces of the user who tap the authenticated RFID card. This is to ensure that the scanned RFID card belongs to the user.

7. EC2 instance is implemented for remote access to web interface.

E. Quick-start guide (Readme first)

1) First connect hardware as in Section 2 and the Fritzing diagram.
2) Go through all the instructions in Section 3.
3) Then run the server.py file for web server.
4) Run the telegrambot.py for Telegram bot.
5) Run the monicam_main.py for the main program.
6) Run the lightmotion.py
7) Run the alarmdetector.py
8) Access the webpage by typing <your AWS public IPv4 address>:5000

Section 2
Hardware requirements

Hardware checklist

1. 1 LDR (light sensor)
2. 1 PIR motion sensor
3. 1 RFID Card reader + 3 RFID cards
4. I2C 16x2 LCD display
5. 3 LEDs (1 red, yellow and green LED)
6. 2 Piranha LEDs
7. 1 Buzzer
8. Raspberry Pi camera (piCam)
9. 4 330Ω resistors (for 3 LEDs and 2 piranha LEDS)
10. 1 10kΩ resistor (for light sensor)
11. Arduino
12. Raspberry Pi
13. USB cable (for connecting Arduino and Raspberry Pi)
14. T-Cobbler breadboard
15. Half breadboard

Hardware setup instructions

The followings are the hardware set-up on Rasbperry Pi:

For the RFID card reader, connect the pins on the MFRCF522 card reader to the RPi as indicated below.

Jumper color	MFRCF522pin	RPi pin
Yellow	SDA	CE0
Orange	SCK	SCLK
Green	MOSI	MOSI
Blue	MISO	MISO
NIL	IDR	NIL
Black	GND	GND
White	RST	GPIO25
Red	3.3V	3.3V
NIL	5V	NIL

For the LCD display, connect the pins on the LCD to the RPi as follows:

Jumper color	LCD pin	RPi pin
White	SCL	SCL
Yellow	SDA	SDA
Black	GND	GND
Red	Vcc	5V

For the two wires that is near the word LED, either use a jumper, a female to female cable, or bend them together so that they touch each other, so as to connect them to one another for your LCD display to have backlight.

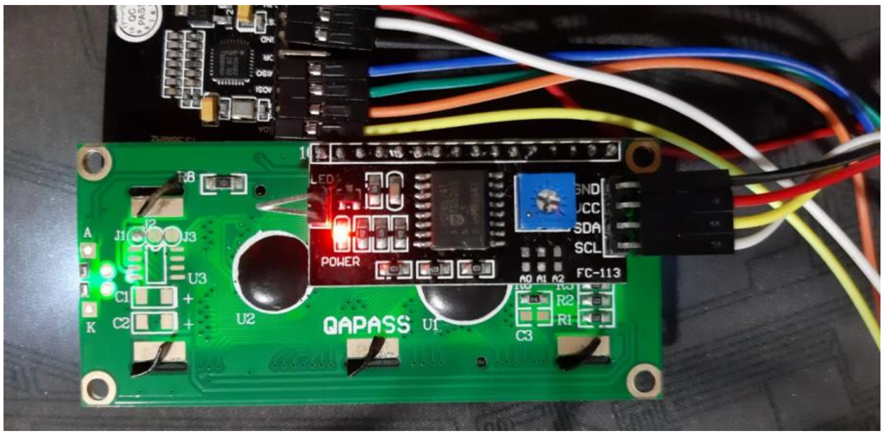


Figure 8: the two LED wires connected together

For the LEDs, connect the positive end of the LEDs to the RPi GPIO pins as indicated:

LED	Pin number
Red	19
Yellow	20
Green	21

The negative ends should be connected to the 330 ohms resistors, which are directly connected to the ground power rail.

The buzzer is connected to RPi GPIO pin 5, where the negative end of the buzzer is directly connected to the ground power rail.

The following is the hardware set-up on Arduino:

For the PIR motion sensor, connect it as follows:

Jumper color	PIR pin	Arduino pin
Red	VCC	5V
Black	GND	GND
Yellow	VOUT	2

The motion sensor might need some tuning to calibrate to the desired sensitivity, so follow the image below on calibrating the PIR motion sensor.

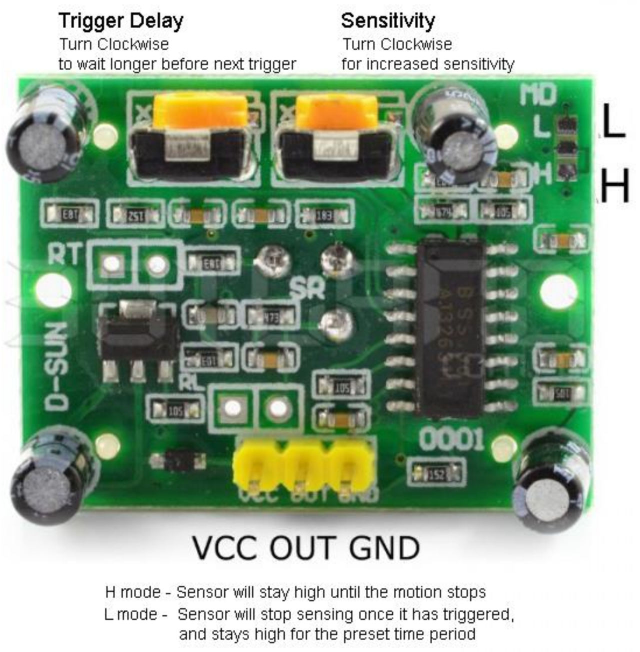

Trigger Delay
Turn Clockwise
to wait longer before next trigger
Sensitivity
Turn Clockwise
for increased sensitivity
L
H
VCC OUT GND
H mode - Sensor will stay high until the motion stops
L mode - Sensor will stop sensing once it has triggered, and stays high for the preset time period

Figure 9: How to calibrate a motion sensor

For the LDR light sensor, connects as indicated:

Jumper color	PIR pin	Arduino pin
Red	VCC	5V
Black	GND	GND
Yellow	VOUT	A0

For the piranha LEDs, we will be using 330Ω resistor to moderate the current flowing through the LEDs. You can use as many LEDs as you like as we are wiring the LEDs in parellel. Connect one end of the wire to the pin 13, and another end to the part of the breadboard that has sufficient space and not connected to anything.

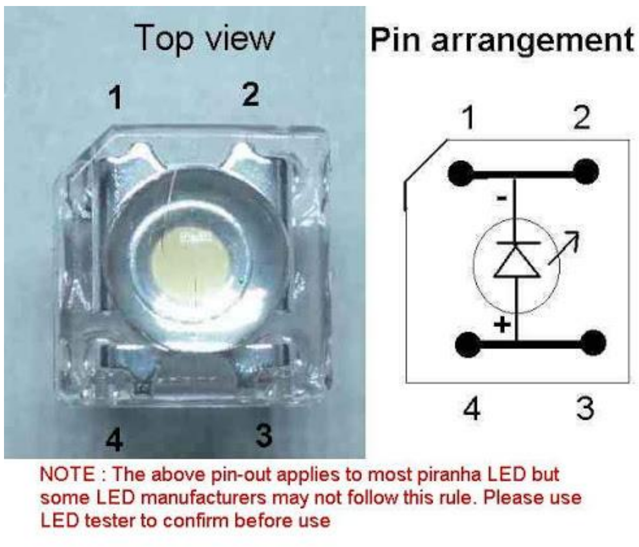

Top view
1
2
4
3
Pin arrangement
1
2
-
+
4
3
NOTE : The above pin-out applies to most piranha LED but some LED manufacturers may not follow this rule. Please use LED tester to confirm before use

Figure 10: Piranha LED Diagram

Connect a 330Ω resistor to the other end of the wire. Connect the positive end of the led to the resistor using a wire on the breadboard. Take another wire and connect to the breadboard row that is connected to the led on the positive end. Connect the other end of the wire to another LED. Connect a wire to the ground power rail, and the end of that wire to the one of the LED's negative end. Connect a wire between the two LED's negative ends.

Lastly, simply connect the Raspberry Pi and the Arduino with a USB cable.

Fritzing Diagram

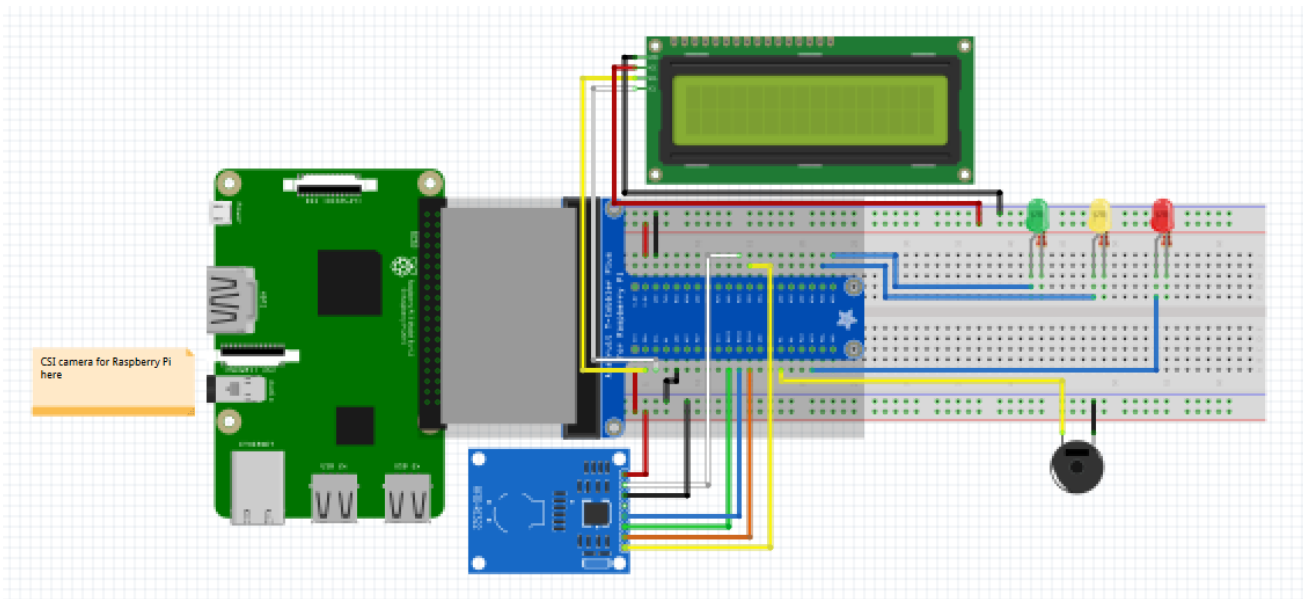


Figure 11: Fritzing Diagram of Raspberry Pi

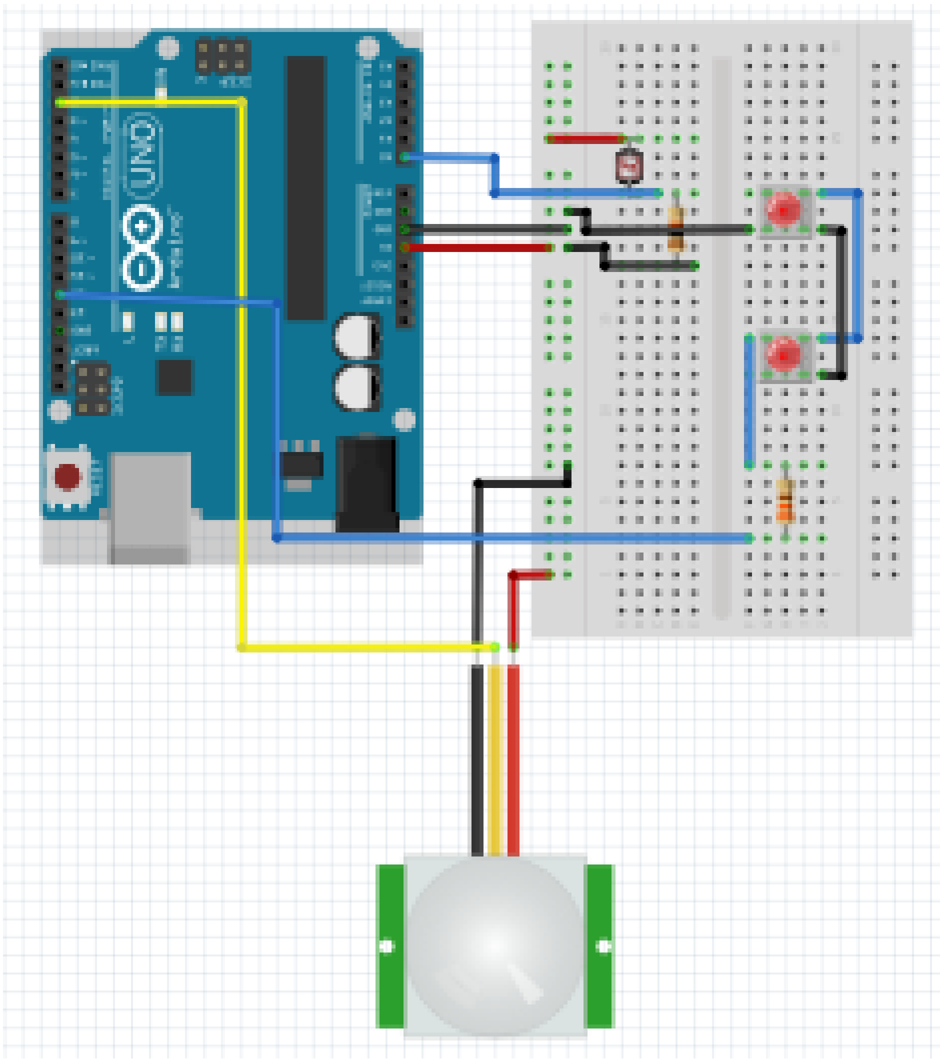


Figure 12: Fritzing Diagram of Arduino

Section 3
Software Requirements

Software checklist

This is the list of libraries required for this project:

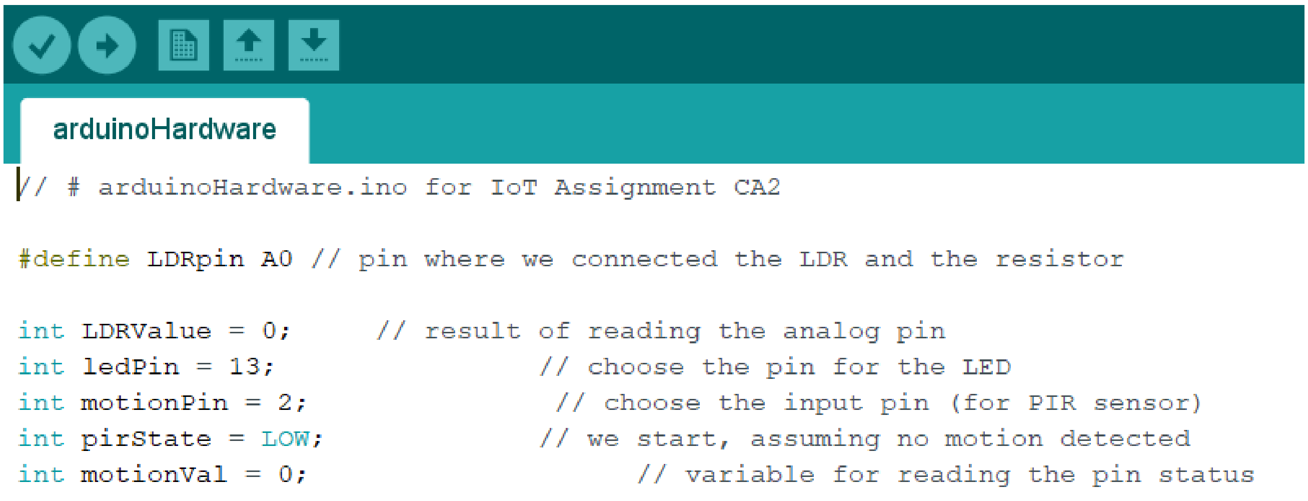
1. boto3 library
2. botocore library
3. json library
4. telepot library
5. MFRC522 library
6. rpi_lcd library
7. AWSIoTPythonSDK library
8. paho-mqtt library

Software setup instructions

Arduino

1. After writing LDR.ino in the Arduino software as shown in Section 4, connect the Arduino to the computer with a USB cable.

~~1.~~2. Upload the script to Arduino by clicking the right arrow at the top of the software.



```
// # arduinoHardware.ino for IoT Assignment CA2

#define LDRpin A0 // pin where we connected the LDR and the resistor

int LDRValue = 0;      // result of reading the analog pin
int ledPin = 13;                // choose the pin for the LED
int motionPin = 2;               // choose the input pin (for PIR sensor)
int pirState = LOW;             // we start, assuming no motion detected
int motionVal = 0;                    // variable for reading the pin status
```

Figure 13: Uploading Script in Arduino

Connection between Raspberry Pi and Arduino

1. In your Raspberry Pi interface, navigate to the start menu and select "Preferences" and then "Raspberry Pi Configuration".

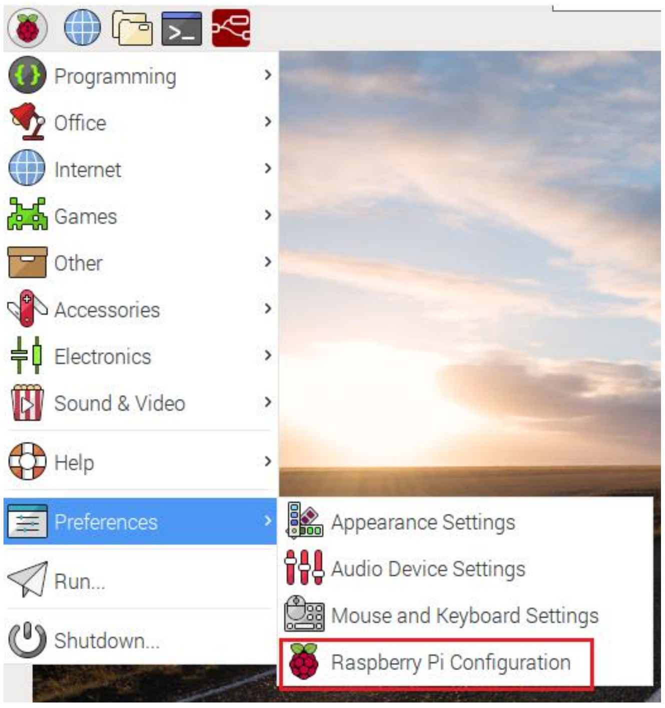
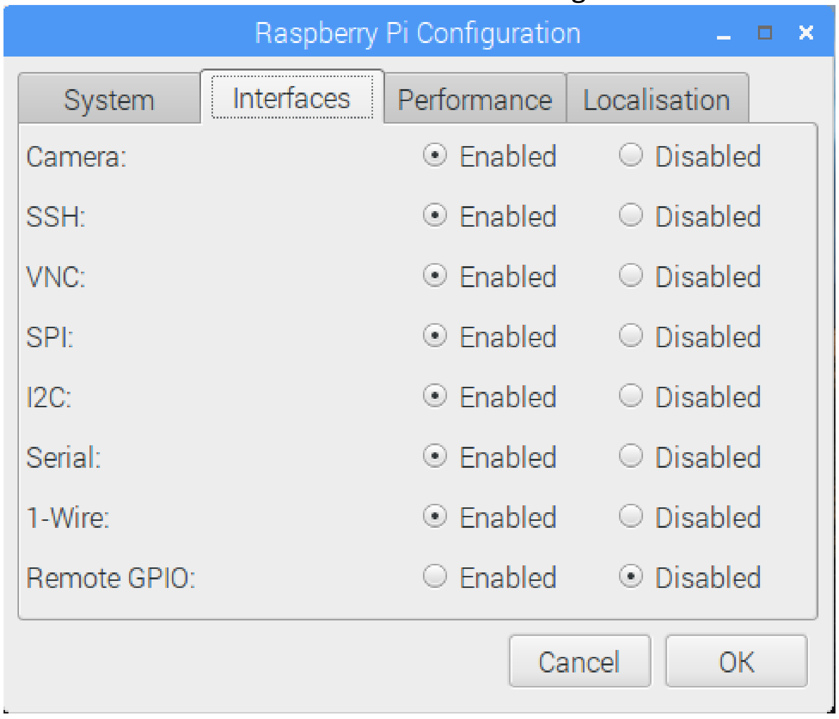


Figure 14: Raspberry Pi's Start Menu

2. Ensure that I2C and Serial are enabled as shown in the figure below.



Camera: [✓ Enabled] Disabled
SSH: [✓ Enabled] Disabled
VNC: [✓ Enabled] Disabled
SPI: [✓ Enabled] Disabled
I2C: [✓ Enabled] Disabled
Serial: [✓ Enabled] Disabled
1-Wire: [✓ Enabled] Disabled
Remote GPIO: Enabled [✓ Disabled]

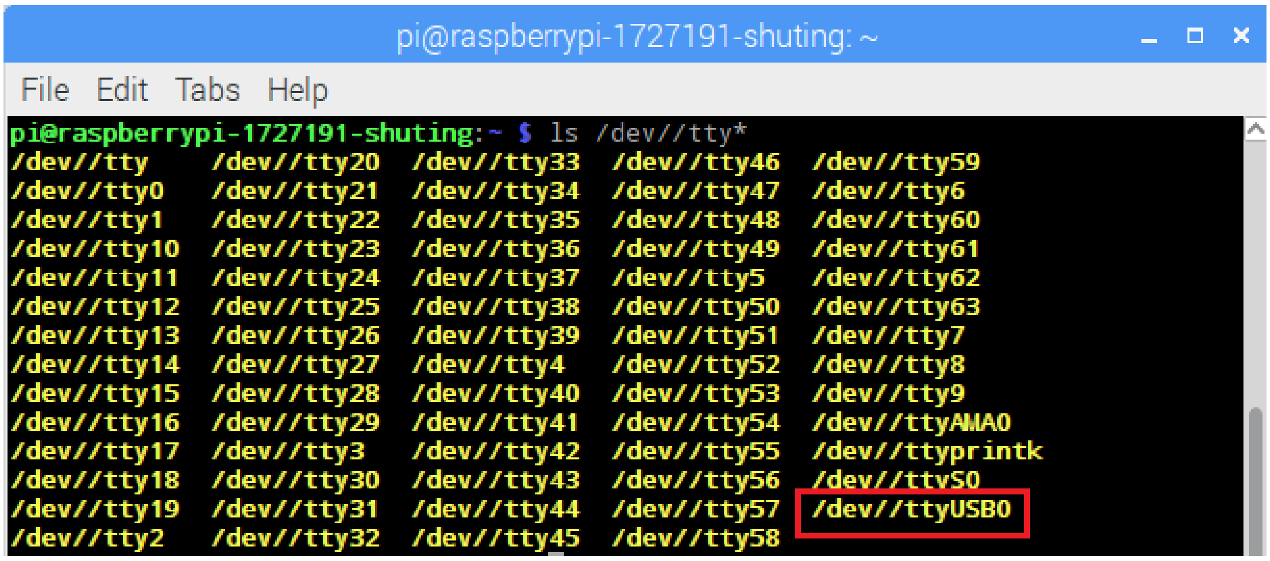
Figure 15: Raspberry Pi Configuration

3. After doing the above, restart your Raspberry Pi. Then, open a terminal console and execute the following commands:

```
sudo apt-get install python-serial
sudo pip install pyserial
```

4. Ensure your Raspberry Pi and Arduino are connected via a USB cable. The serial port that is receiving the Arduino data must be known for them to communicate. To find out the serial port, simply execute the following command:

```
ls /dev/tty*
```



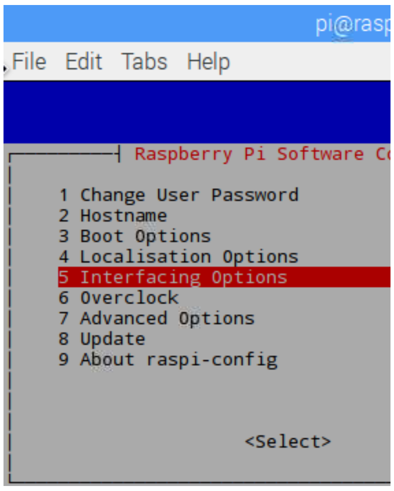
```
pi@raspberrypi-1727191-shuting:~ $ ls /dev//tty*
/dev//tty    /dev//tty20  /dev//tty33  /dev//tty46  /dev//tty59
/dev//tty0   /dev//tty21  /dev//tty34  /dev//tty47  /dev//tty6
/dev//tty1   /dev//tty22  /dev//tty35  /dev//tty48  /dev//tty60
/dev//tty10  /dev//tty23  /dev//tty36  /dev//tty49  /dev//tty61
/dev//tty11  /dev//tty24  /dev//tty37  /dev//tty5   /dev//tty62
/dev//tty12  /dev//tty25  /dev//tty38  /dev//tty50  /dev//tty63
/dev//tty13  /dev//tty26  /dev//tty39  /dev//tty51  /dev//tty7
/dev//tty14  /dev//tty27  /dev//tty4   /dev//tty52  /dev//tty8
/dev//tty15  /dev//tty28  /dev//tty40  /dev//tty53  /dev//tty9
/dev//tty16  /dev//tty29  /dev//tty41  /dev//tty54  /dev//ttyAMA0
/dev//tty17  /dev//tty3   /dev//tty42  /dev//tty55  /dev//ttyprintk
/dev//tty18  /dev//tty30  /dev//tty43  /dev//tty56  /dev//ttyS0
/dev//tty19  /dev//tty31  /dev//tty44  /dev//tty57  /dev//ttyUSB0
/dev//tty2   /dev//tty32  /dev//tty45  /dev//tty58
```

*Figure 16: ls /dev/tty**

In this project, the used USB port is /dev/ttyUSB0 and it may vary depending on which USB port you have plugged into. It will typically be /dev/ttyUSBx, for x is a number. This port number will be used in the scripts.

Enable Raspberry Pi Camera

1. If the SPI and Camera interface is not enabled, you need to enable it through raspi-config. You can get to raspi-config by running *sudo rasp-config* command in the terminal, where under interfacing options, you will see an option allowing you to enable the SPI and Camera interface.

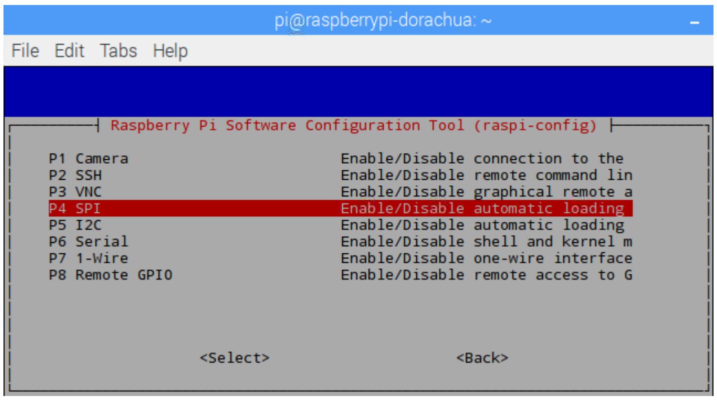


```
pi@rasp
File  Edit  Tabs  Help

Raspberry Pi Software Co

  1 Change User Password
  2 Hostname
  3 Boot Options
  4 Localisation Options
  5 Interfacing Options
  6 Overclock
  7 Advanced Options
  8 Update
  9 About raspi-config

                  <Select>
```



```
pi@raspberrypi-dorachua: ~
File  Edit  Tabs  Help

Raspberry Pi Software Configuration Tool (raspi-config)

  P1 Camera              Enable/Disable connection to the
  P2 SSH                 Enable/Disable remote command lin
  P3 VNC                 Enable/Disable graphical remote a
  P4 SPI                 Enable/Disable automatic loading
  P5 I2C                 Enable/Disable automatic loading
  P6 Serial              Enable/Disable shell and kernel m
  P7 1-Wire              Enable/Disable one-wire interface
  P8 Remote GPIO         Enable/Disable remote access to G

          <Select>                <Back>
```

Figure 17: raspi-config

The screenshots here may differ depending on your raspberry version and hardware.

You also need to modify the /boot/config.txt file to include the following lines.

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

Install Necessary Libraries

1. Run the setup.sh file in the terminal by typing the following commands:
   a. chmod +x setup.sh (make it executable)
   b. ./setup.sh (run the bash script)

The bash script will help you install the required libraries.

Telegram Bot API Key

1. You could request for a Telegram bot API key, or use the current Telegram bot API key. The current bot name is called monicam_bot. For creating your own Telegram bot API key, you could refer to this link or other online resources that will guide you through the creation of the Telegram bot API key.

```
my_bot_token =
```

Figure 18: Replace the existing Telegram API key with your own key in telegrambot.py

2. Find the monicam_bot and start it, but if you are using your own telegram bot, find it in telegram and start it. After starting the bot, type chatid to request for your chatid with the bot, and replace the existing owner_chat_id number with your chatid number the bot returns you, as this allows the bot to send messages to you as the owner.

```
owner_chat_id = 446446269
```

Figure 19: Replace the above with your own chatid in telegrambot.py

Section 4
AWS

AWS Console

1. Login into your AWS account from the link below.
   https://www.awseducate.com/signin/SiteLogin
2. After signing in, click "My Classroom" at the top navigation bar.

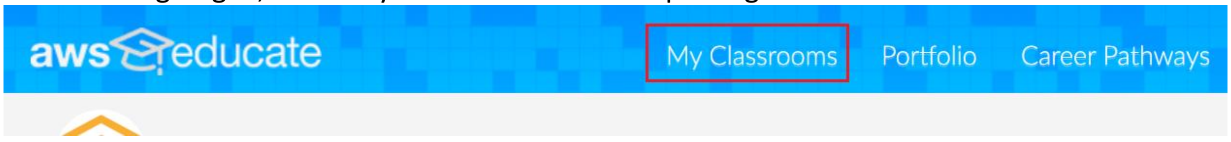
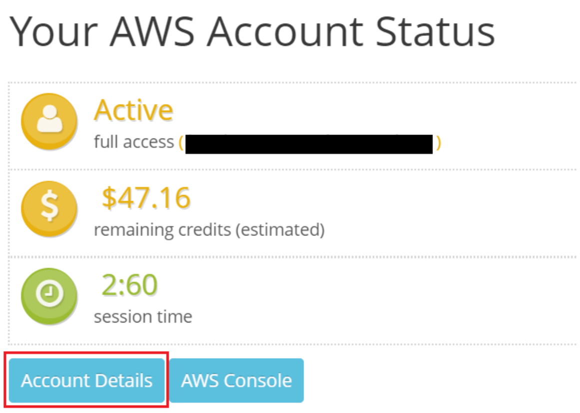


Figure 20: Select "My Classrooms"

3. In "My Classrooms", click "Go to classroom"

4. Under AWS account status, click "Account Details" to obtain your AWS credentials. Keep a copy of it as it will be needed later.



Your AWS Account Status

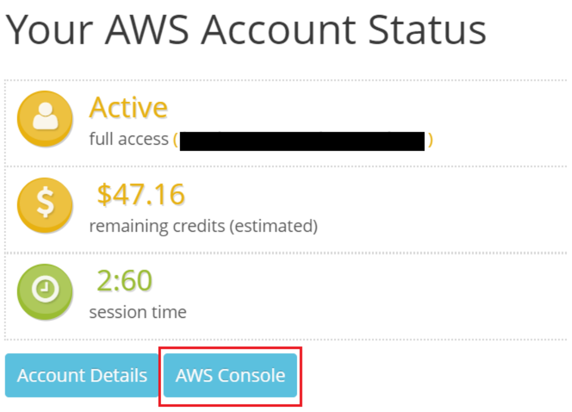
Active
full access ([redacted])

$47.16
remaining credits (estimated)

2:60
session time

Figure 21: Account Details

5. Now, click "AWS Console" under AWS account status and proceed to the next set-up.



Your AWS Account Status

Active
full access ([redacted])

$47.16
remaining credits (estimated)

2:60
session time

Figure 22: AWS Console

Register your Raspberry Pi as a Thing

A thing represents a device whose status or data is stored in the AWS cloud and in this case, it will be the Raspberry Pi.

1. In AWS Console, search for "IoT Core".

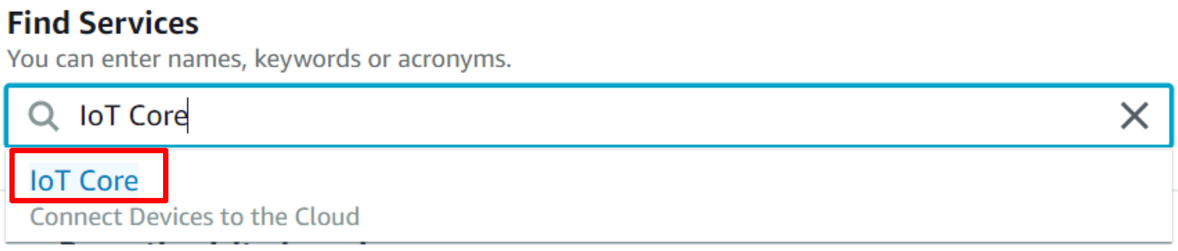

Find Services
You can enter names, keywords or acronyms.
IoT Core
IoT Core
Connect Devices to the Cloud

Figure 23: Iot Core Service

2. In IoT Core's side navigation, navigate to "Manage" and then click "Things".
3. On top right of the page, click "Create".

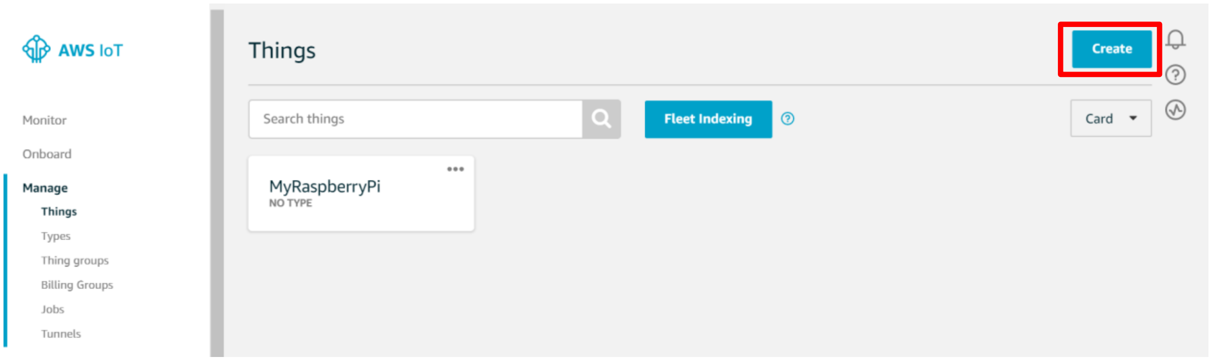
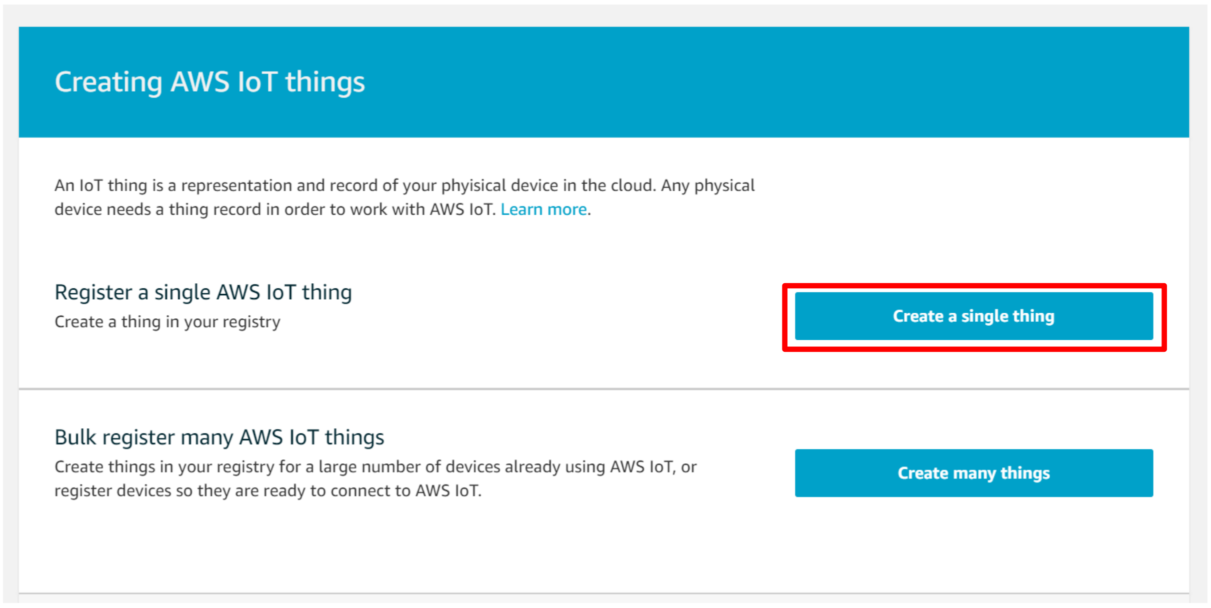


Figure 24: Create a Thing

4. Select "Create a single thing"


Creating AWS IoT things

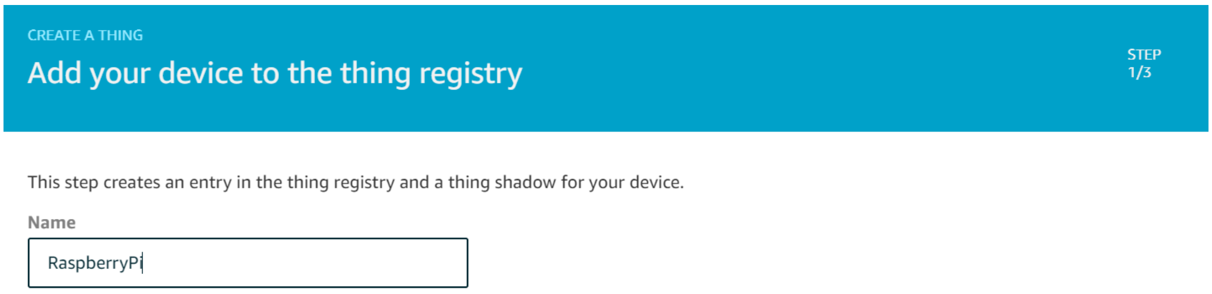
An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. Learn more.

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

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.

Figure 25: Create a Single Thing

5. name your thing as "RaspberryPi" and click "Next" at the bottom right of the page.


CREATE A THING

Add your device to the thing registry

STEP 1/3

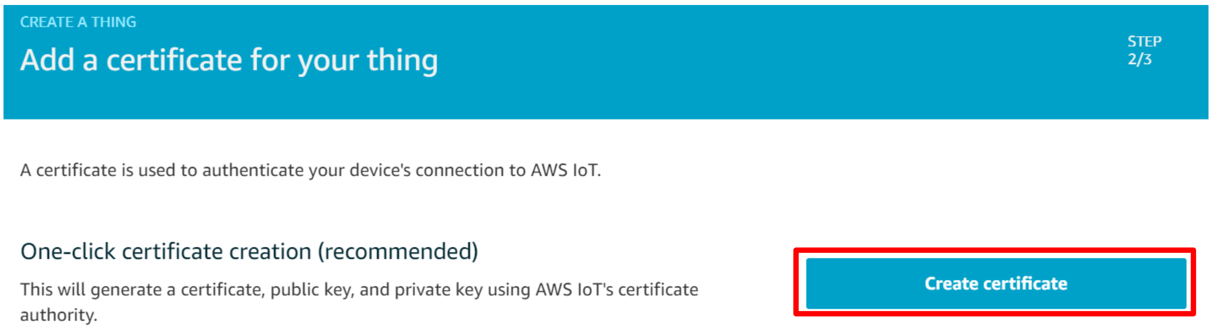
This step creates an entry in the thing registry and a thing shadow for your device.

Name

RaspberryPi

Figure 26: Name your Thing

6. Now we will proceed to creating the certificate for the thing. Click "Create certificate" on the following page.


CREATE A THING

Add a certificate for your thing

STEP 2/3

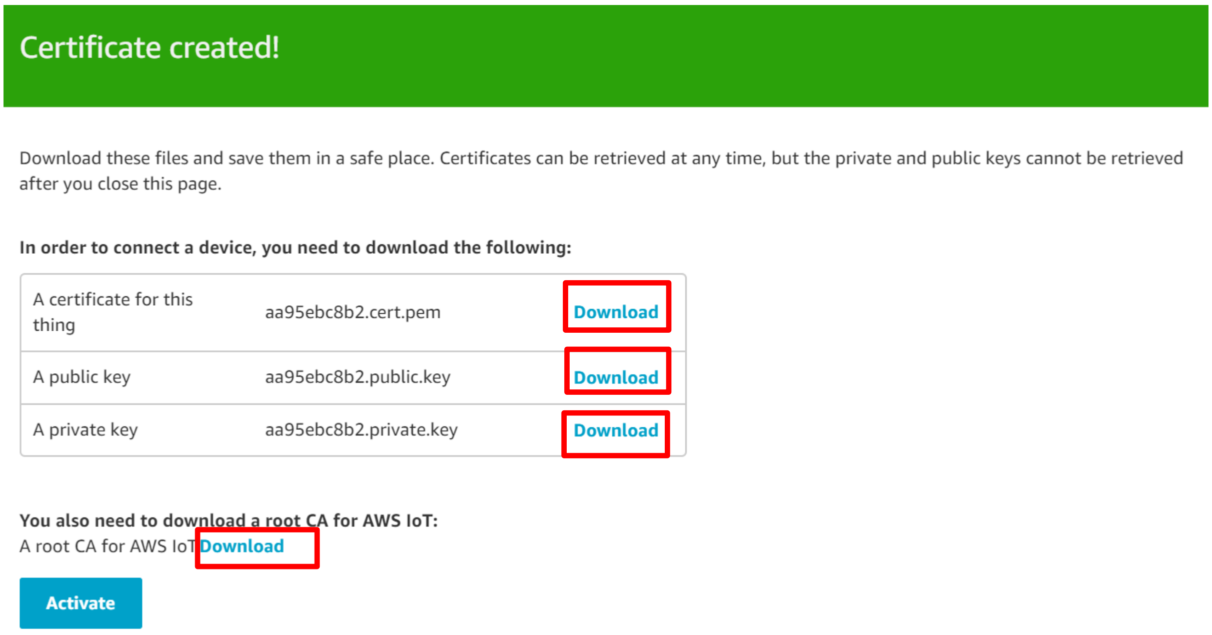
A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)

This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

Figure 27: Create Certificate

7. You will be brought to a page that states that the certifcate is created. Download the following:
    a. Certificate for the thing
    b. Public key
    c. Private key
    d. Root CA
        i. For the root CA, do a right-click at "Amazon Root CA 1" and select "Save As". You can just name it as "rootca.pem".


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.

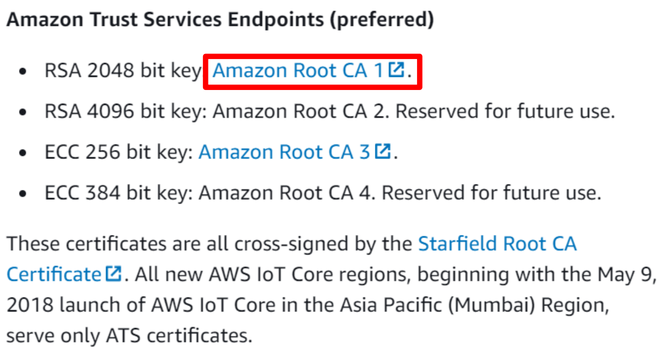
In order to connect a device, you need to download the following:

A certificate for this thing	aa95ebc8b2.cert.pem	Download
A public key	aa95ebc8b2.public.key	Download
A private key	aa95ebc8b2.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT Download

Activate

Figure 28: Download Files

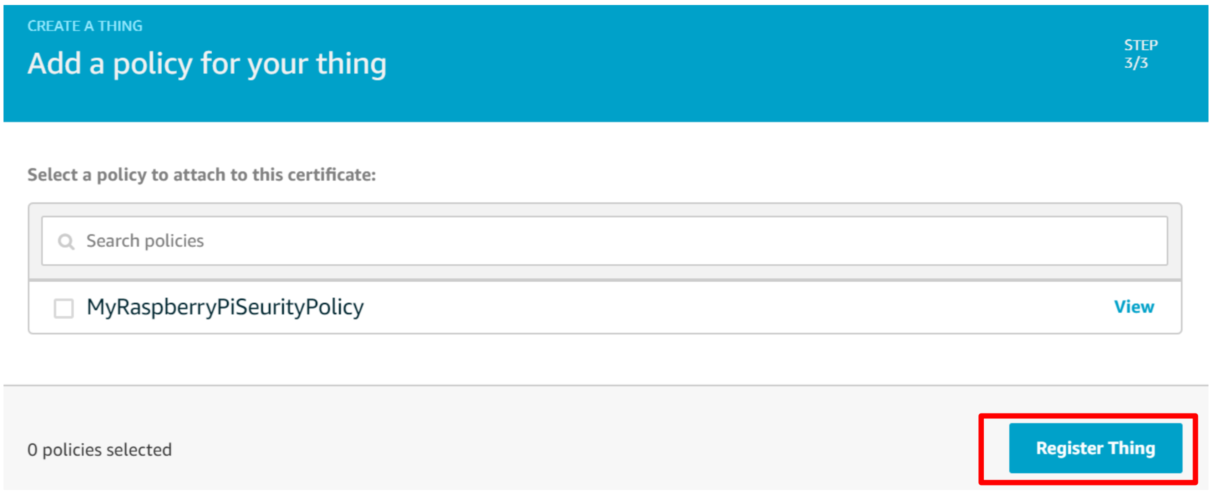

Amazon Trust Services Endpoints (preferred)

- RSA 2048 bit key: Amazon Root CA 1.
- RSA 4096 bit key: Amazon Root CA 2. Reserved for future use.
- ECC 256 bit key: Amazon Root CA 3.
- ECC 384 bit key: Amazon Root CA 4. Reserved for future use.

These certificates are all cross-signed by the Starfield Root CA Certificate. All new AWS IoT Core regions, beginning with the May 9, 2018 launch of AWS IoT Core in the Asia Pacific (Mumbai) Region, serve only ATS certificates.

Figure 29: Save the Amazon Root CA 1

8. After downloading the required files, click “Attach a policy” at the bottom right of the page. You can leave the “Search policies” empty and just click “Register Thing”.


CREATE A THING
Add a policy for your thing
STEP 3/3

Select a policy to attach to this certificate:

Search policies

☐ MyRaspberryPiSeurityPolicy View

0 policies selected

Register Thing

Figure 30: Register the Thing

Copy REST API endpoint of the Thing

1. Return the the "Thing" page under "Manage". Click on the thing you have just created. On the next screen, click "Interact".

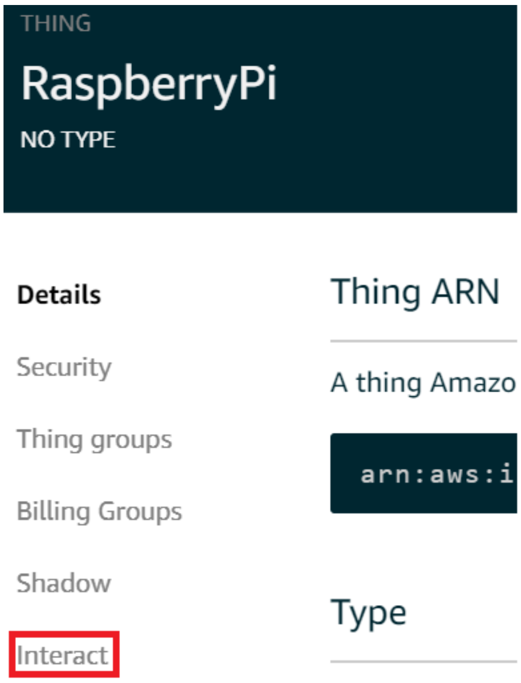
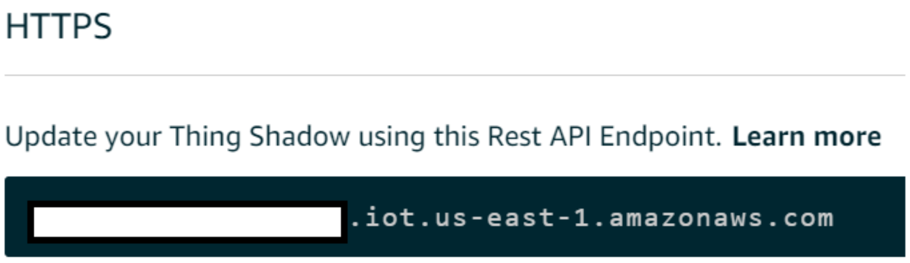


Figure 31: Selected Thing

2. Copy the REST API endpoint as it will be needed later.



HTTPS

Update your Thing Shadow using this Rest API Endpoint. **Learn more**

```
[redacted].iot.us-east-1.amazonaws.com
```

Figure 32: REST API Endpoint

Create Security Policy for the Raspberry Pi

1. In Iot Core page, go to "Secure" then "Policies" at the side navigation. Then click "Create" at the top right of the page.

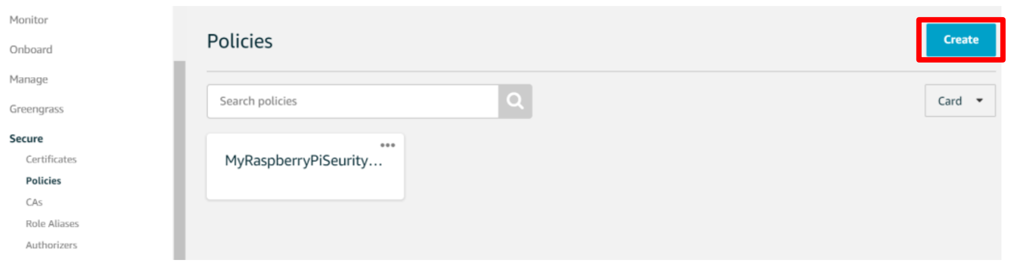
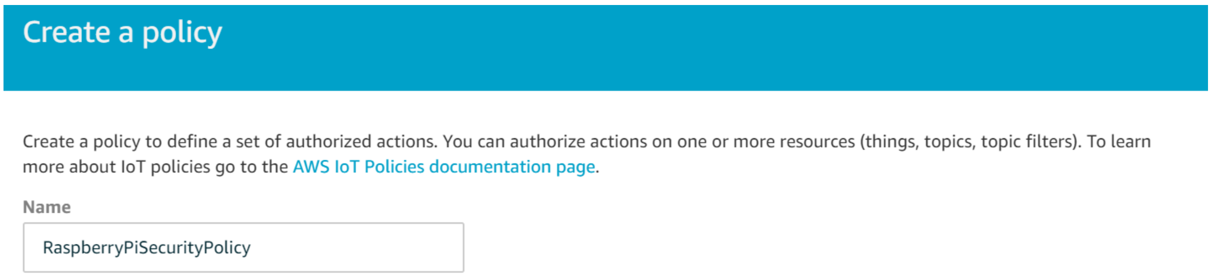


Figure 33: Create Security Policy

2. Key in the respective values and click “Create” at the bottom of the page:

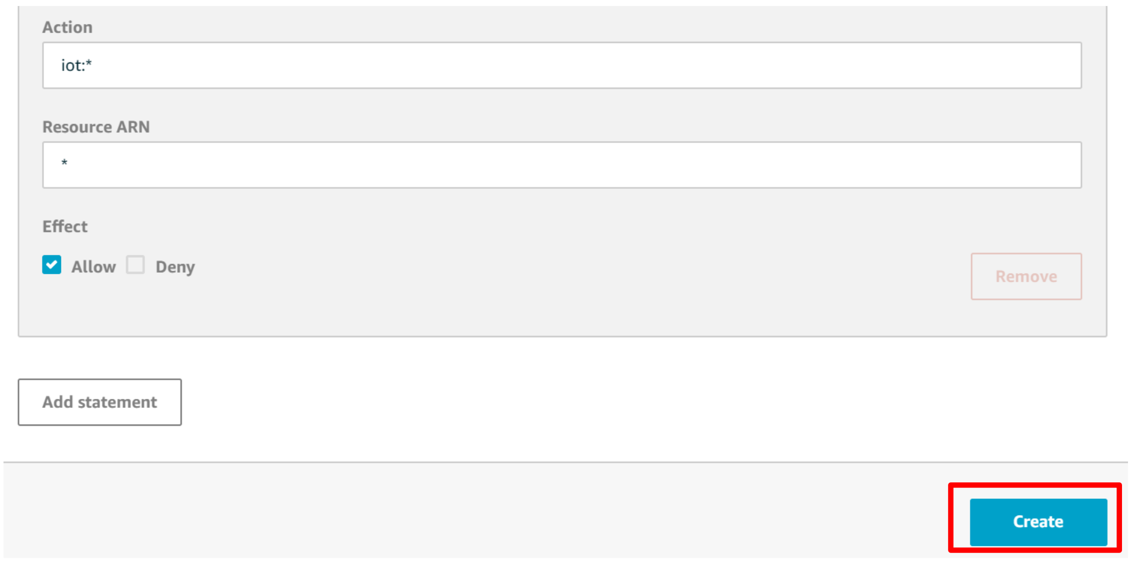
Name	RaspberryPiSecurityPolicy
Action	Iot:*
Resource ARN	*
Effect	Check “Allow”


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: RaspberryPiSecurityPolicy

Figure 34: Values for Policy


Action: iot:*

Resource ARN: *

Effect: [✓ Allow] Deny

Figure 35: Values for Policy

Attach Security Policy to your Certificate

1. In Iot Core page, go to “Secure” then “Certificates” at the side navigation. Check the certificate that you have just created. Then click “Actions” at the top right and select “Attach policy”.

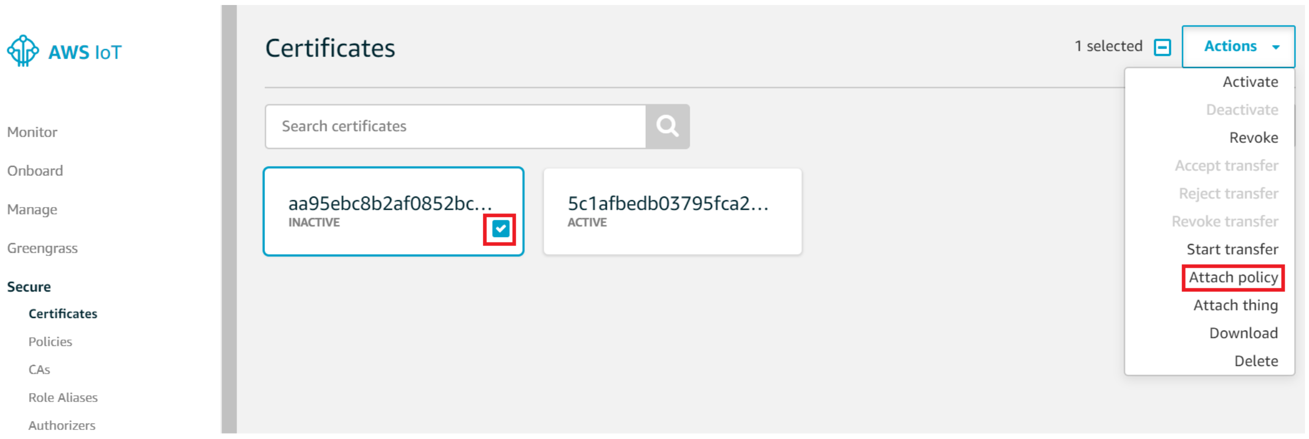
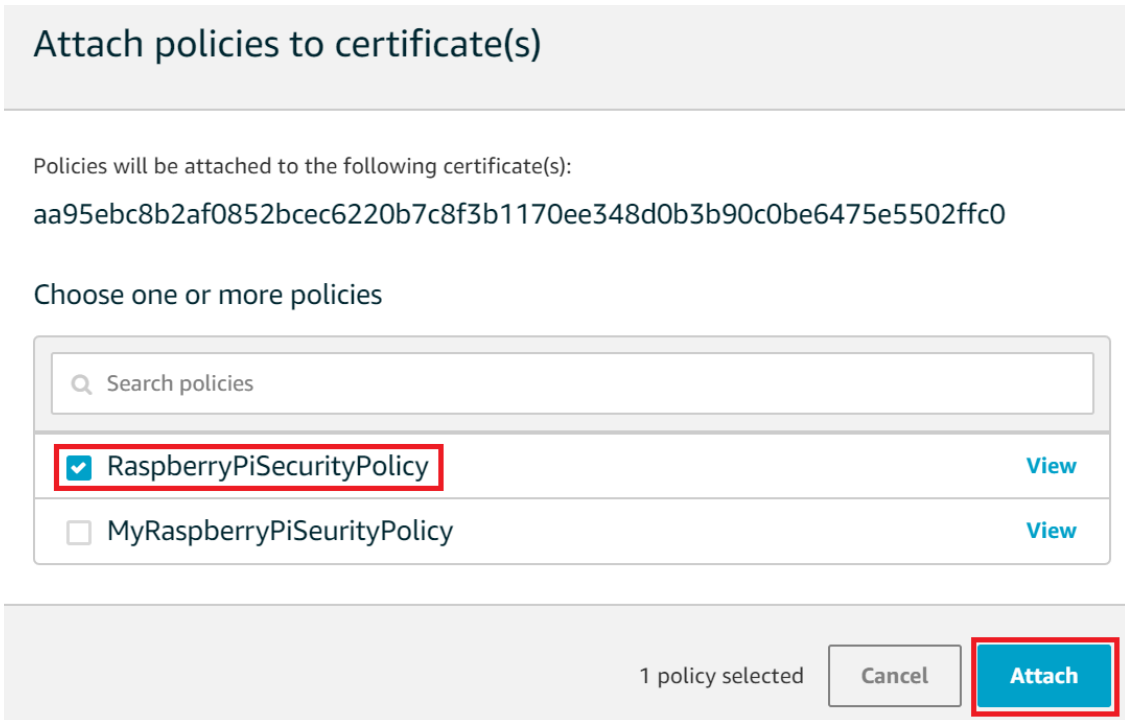


Figure 36: Attach Policy

2. Select the security policy that you have just created and click “Attach”.


Attach policies to certificate(s)

Policies will be attached to the following certificate(s):

aa95ebc8b2af0852bcec6220b7c8f3b1170ee348d0b3b90c0be6475e5502ffc0

Choose one or more policies

Figure 37: Select the Security Policy

Attach the Thing to your Certificate

1. In Iot Core page, go to “Secure” then “Certificates” at the side navigation. Check the certificate that you have just created. Then click “Actions” at the top right and select “Attach thing”.

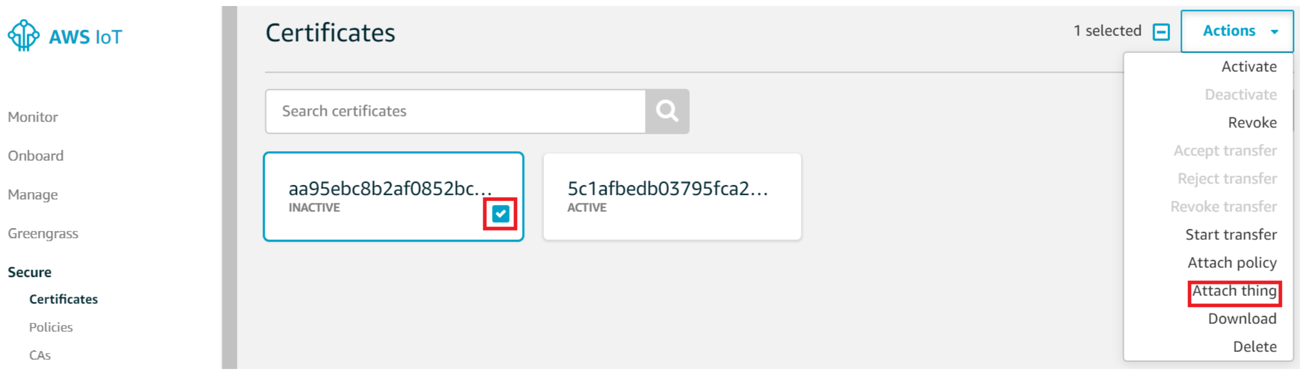
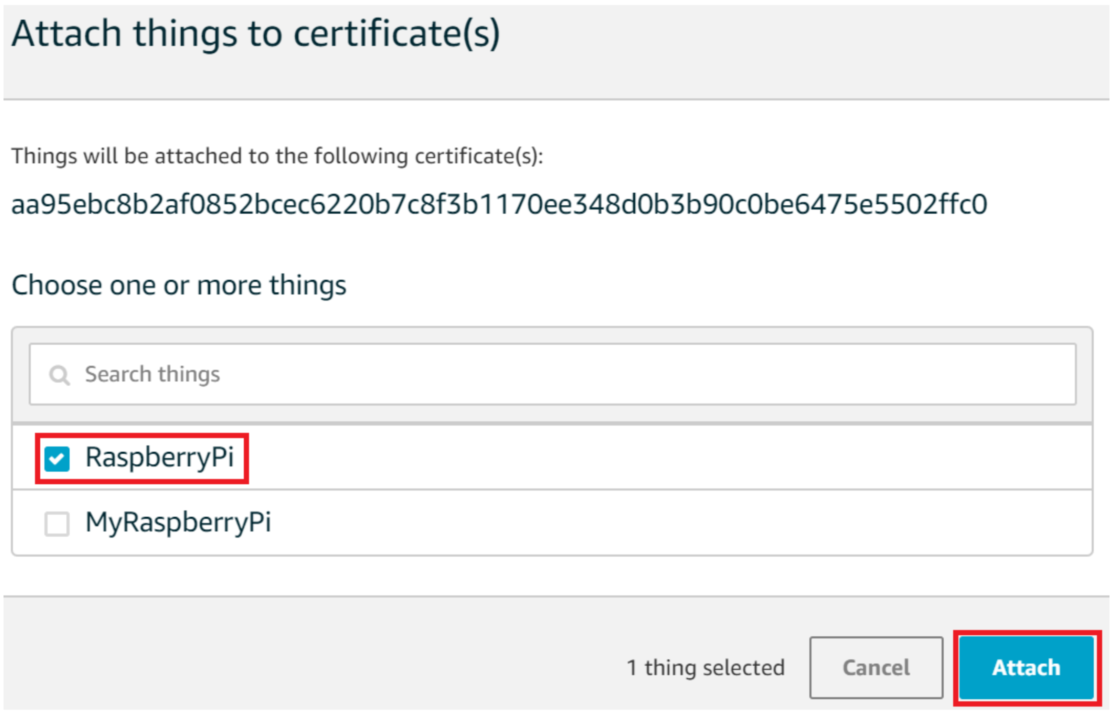


Figure 38: Attach Thing

2. Select the thing that you have just created and click “Attach”.


Attach things to certificate(s)

Things will be attached to the following certificate(s):

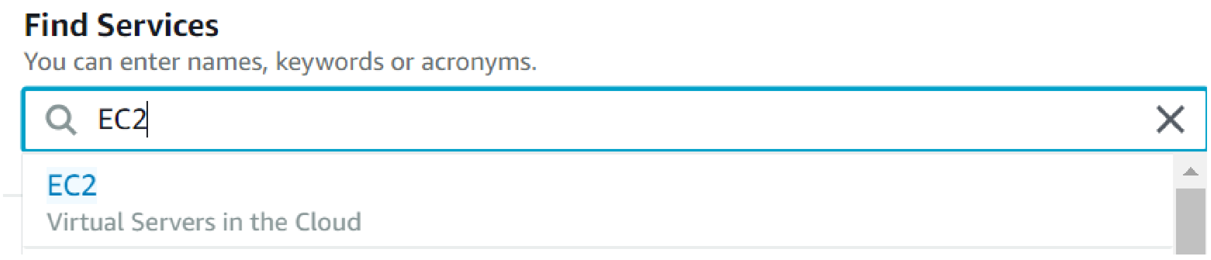
aa95ebc8b2af0852bcec6220b7c8f3b1170ee348d0b3b90c0be6475e5502ffc0

Choose one or more things

Figure 39: Select the Thing

EC2 Server Set-Up

1. In AWS Console, search for “EC2”.



Find Services

You can enter names, keywords or acronyms.

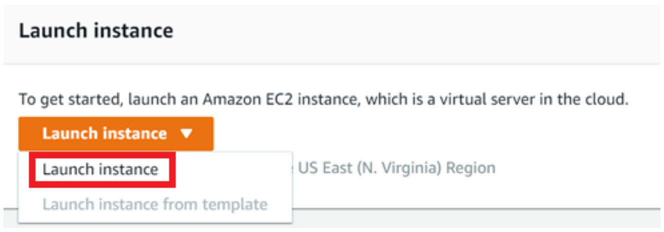
EC2

EC2

Virtual Servers in the Cloud

Figure 40: Search for EC2 Service

2. In EC2, scroll down until you see the following:



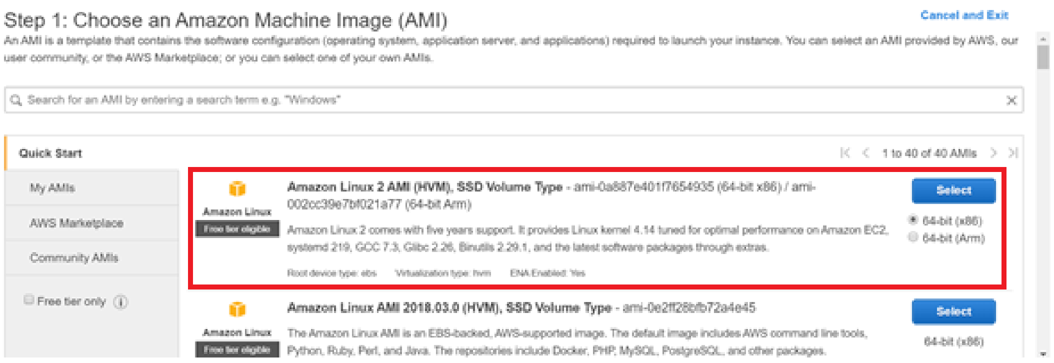
Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Figure 41: Launch Instance

Click the “launch instance” dropdown and select “Launch instance”.

3. In the next screen, select the first image.



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0a887e401f7654935 (64-bit x86) / ami-002cc39e7bf021a77 (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.

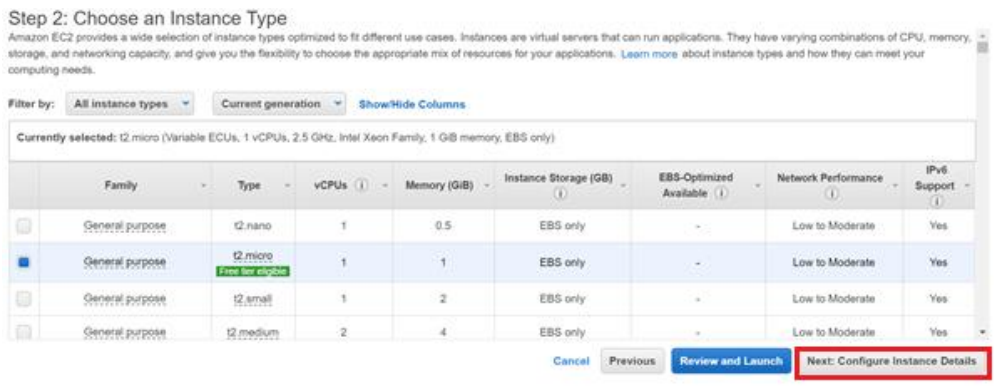
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-0e2ff28bfb72a4e45

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Figure 42: Choosing an Amazon Machine Image (AMI)

4. Then select the default instance type and click “Next Configure Instance Details”.



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

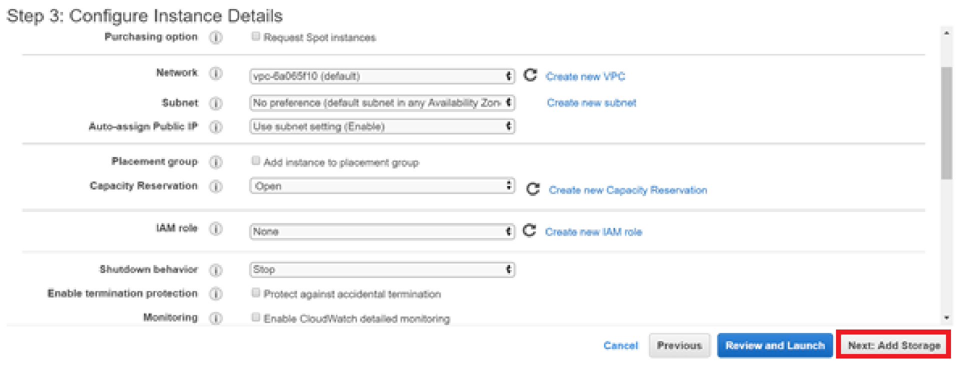
Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes

Figure 43: Choosing an Instance Type

5. Leave the configure instance details as default and click “Next: Add Storage”.

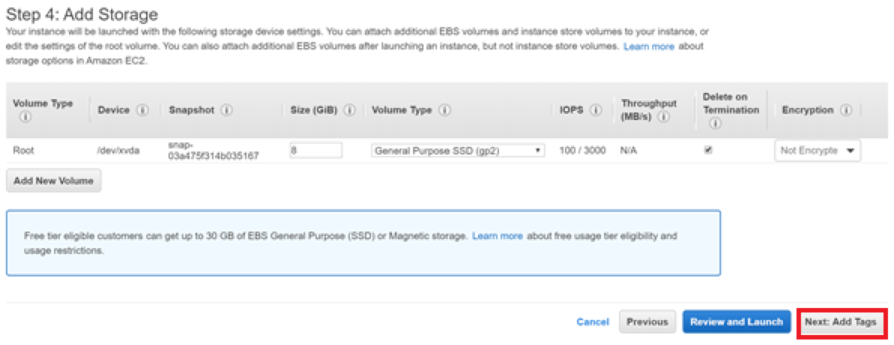


Step 3: Configure Instance Details

Purchasing option	☐ Request Spot instances
Network	vpc-6a065f10 (default)
Subnet	No preference (default subnet in any Availability Zon
Auto-assign Public IP	Use subnet setting (Enable)
Placement group	☐ Add instance to placement group
Capacity Reservation	Open
IAM role	None
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring

Figure 44: Configure Instance Details

6. Leave it as default again and click “Next: Add Tags”.



Step 4: Add Storage

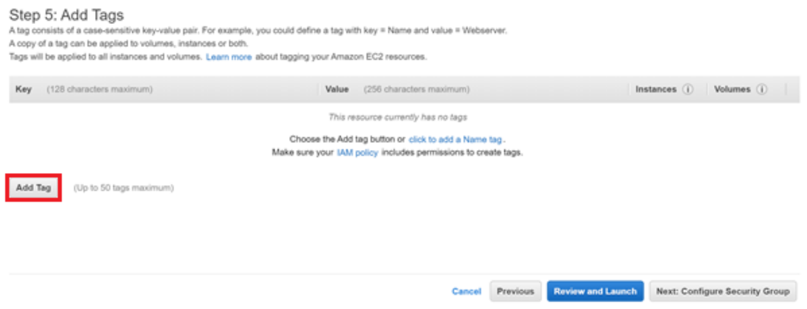
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-03a475f314b035167	8	General Purpose SSD (gp2)	100 / 3000	N/A	✓	Not Encrypte

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

Figure 45: Add Storage

7. Click “Add Tag”.



Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances	Volumes

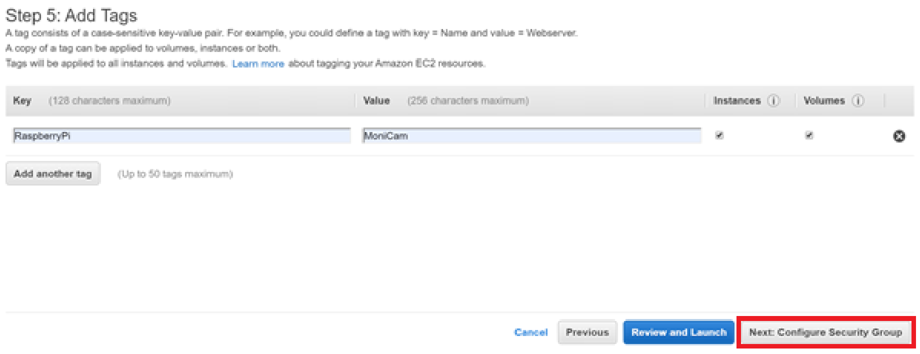
This resource currently has no tags

Choose the Add tag button or click to add a Name tag.
Make sure your IAM policy includes permissions to create tags.

Figure 46: Add Tags

8. Key in the following details and click “Next: Configure Security Group”:

Key	RaspberryPi
Value	MoniCam



Step 5: Add Tags

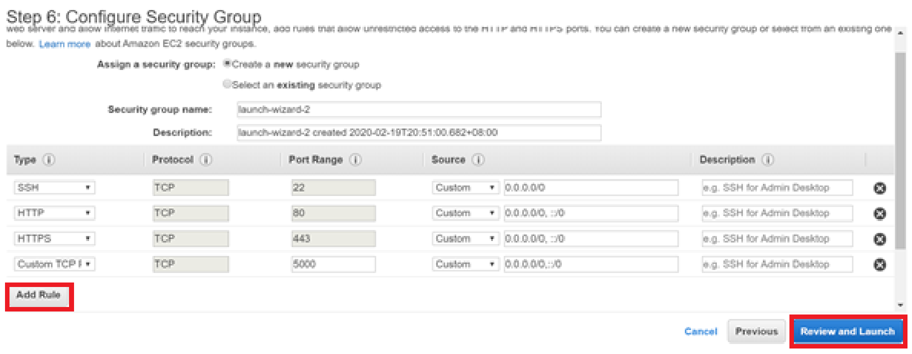
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances	Volumes
RaspberryPi	MoniCam	✓	✓

Figure 47: Tags Information

9. In the next screen, click “Add Rule” and add the following rules shown in the table below. Then click “Review and Launch”.

Type	**Protocol**	**Port Range**	**Source**
SSH	TCP	22	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0, :/0
HTTPS	TCP	443	0.0.0.0/0, :/0
Custom TCP Rule	TCP	5000	0.0.0.0/0, :/0



Step 6: Configure Security Group

web server and allow internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

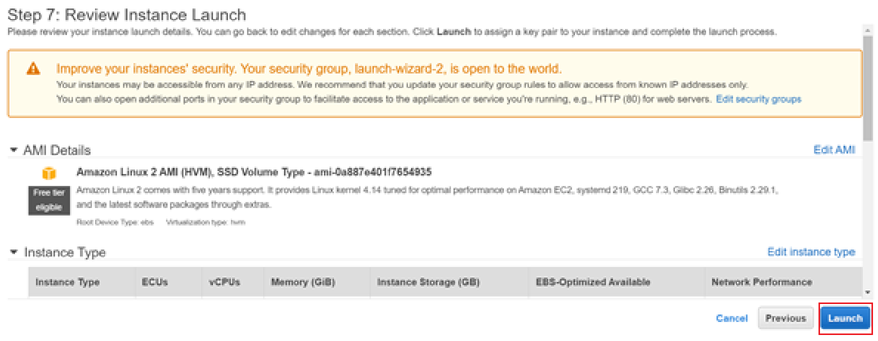
Security group name: launch-wizard-2

Description: launch-wizard-2 created 2020-02-19T20:51:00.682+08:00

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
Custom TCP I	TCP	5000	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Figure 48: Configure Security Group

10. Then, click “Launch”.



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your security group, launch-wizard-2, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

AMI Details

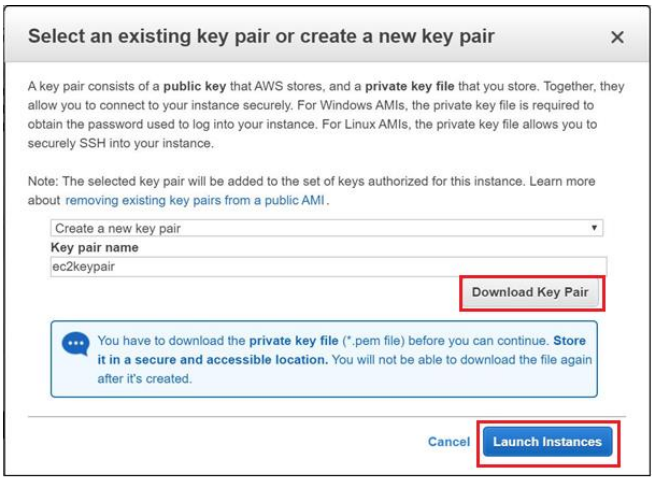
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0a887e401f7654935
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.
Root Device Type: ebs Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance

Figure 49: Review Instance Launch

11. You will be prompted to either select an existing key pair or create a new key pair. We will be creating a new key pair. You can name it whatever you want. Remember to download the key pair as you will need it to establish a connection with the EC2 instance. Lastly, click "Launch Instances".


Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

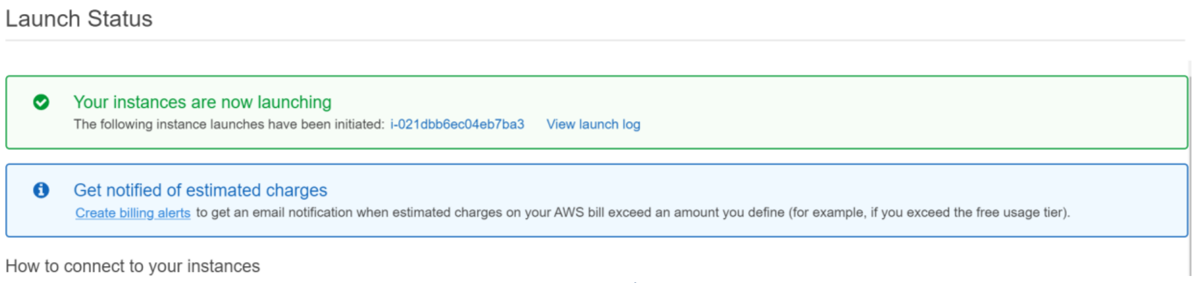
Key pair name

ec2keypair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Figure 50: Create Key Pair

12. If the following page appears, it means you have successfully launched an EC2 instance.


Launch Status

Your instances are now launching
The following instance launches have been initiated: i-021dbb6ec04eb7ba3 View launch log

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

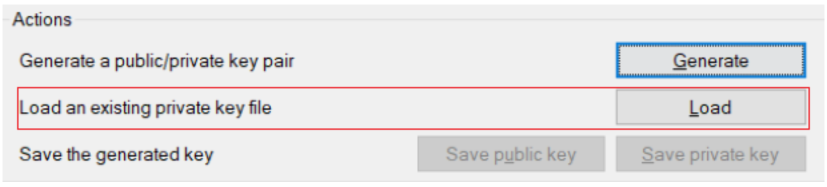
Figure 51: Launch Status

Connecting to EC2 Instance in WINSCP and Transferring Files

1. Install PuTTY and WinSCP before doing the following steps by downloading them from their respective websites.

2. After installing PuTTY with default settings, go to Windows Search and open PuTTYgen, which should be installed alongside with the installation of PuTTY.

3. After opening PuTTYgen, there will be a 'Load an existing private key file' option. Load your private key (.pem) for the EC2 instance obtained from the previous steps.


Actions

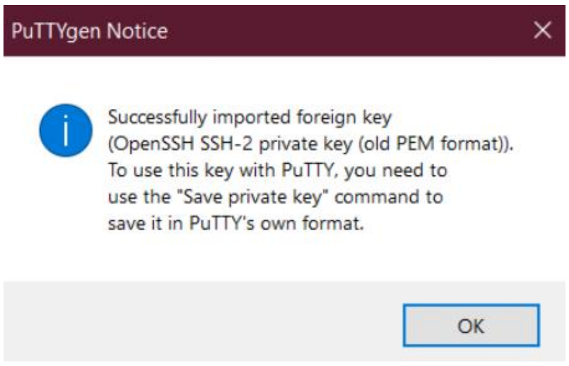
Generate a public/private key pair

Load an existing private key file

Save the generated key

Figure 52: Load existing private key file

4. A popup showing the following will appear. Click "OK".

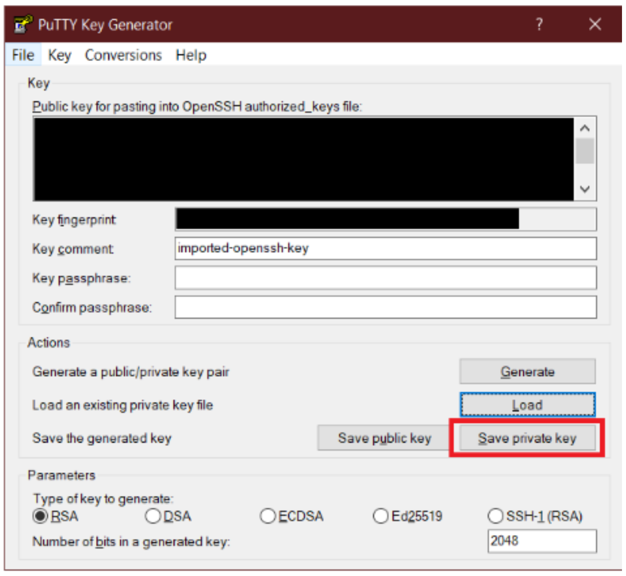


PuTTYgen Notice

Successfully imported foreign key
(OpenSSH SSH-2 private key (old PEM format)).
To use this key with PuTTY, you need to
use the "Save private key" command to
save it in PuTTY's own format.

Figure 53: PuTTygen Notice

5. You should see the following after you have imported your key. Save this key by clicking 'Save private key'.



PuTTY Key Generator

Public key for pasting into OpenSSH authorized_keys file: [redacted]

Key fingerprint: [redacted]

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

Type of key to generate: [✓ RSA] DSA ECDSA Ed25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

File Key Conversions Help
Key
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key
Parameters

Figure 54: Save Private Key

6. Name it the same as the key you used for this, but append a .ppk file extension format behind the file name. Ensure that you save as a .ppk file. You can close the PuTTYgen program now.

7. Open WinSCP. At the top left-hand corner of the program, you should see a 'New Session' tab. Click on it.

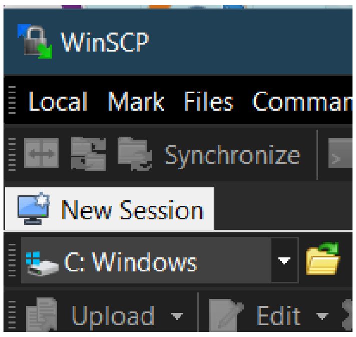
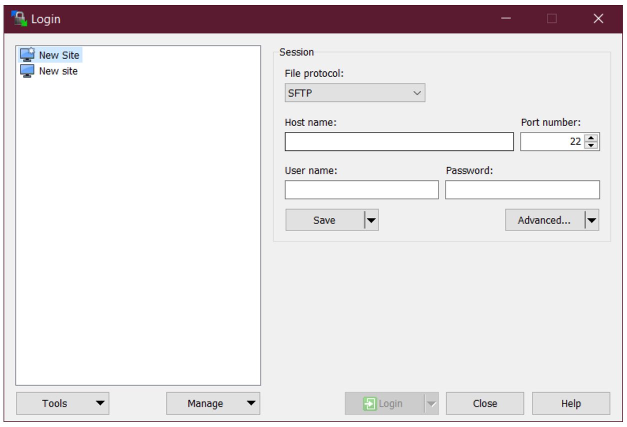


Figure 55: Creating New Session

8. The popup below will appear.



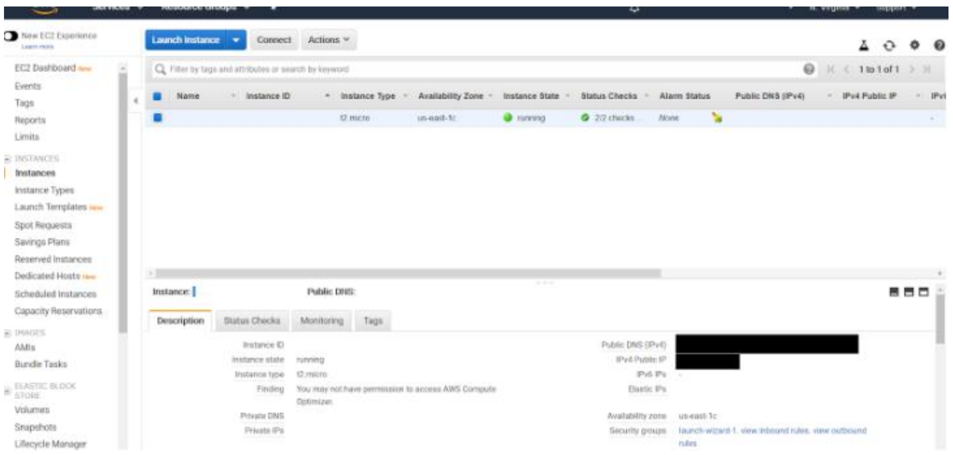
Login

New Site
New site

Session
File protocol: SFTP
Host name:
Port number: 22
User name:
Password:

Figure 56: WinSCP

9. For the hostname above, you can use either the instance's DNS name or the IP address. To obtain the IP address of your EC2 instance, go to your EC2 console, and click on your instance to see the IP address and DNS name (in the blacked-out parts).



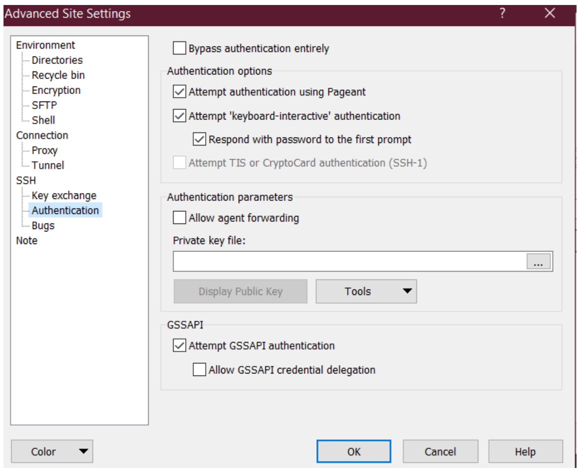
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
		t2.micro	us-east-1c	running	2/2 checks ...	None		

Instance:
Public DNS:

Instance ID
Instance state: running
Instance type: t2.micro
Finding: You may not have permission to access AWS Compute Optimizer
Private DNS
Private IPs
Public DNS (IPv4): [redacted]
IPv4 Public IP: [redacted]
IPv6 IPs: -
Elastic IPs
Availability zone: us-east-1c
Security groups: launch-wizard-1. view inbound rules. view outbound rules

Figure 57: EC2 Instance's IP Address and DNS Name

10. After you have gotten the hostname, click on the Advanced button under the Password input. You will see the following popup.



Advanced Site Settings

☐ Bypass authentication entirely

Authentication options
☑ Attempt authentication using Pageant
☑ Attempt 'keyboard-interactive' authentication
☑ Respond with password to the first prompt
☐ Attempt TIS or CryptoCard authentication (SSH-1)

Authentication parameters
☐ Allow agent forwarding
Private key file:

GSSAPI
☑ Attempt GSSAPI authentication
☐ Allow GSSAPI credential delegation

Figure 58: Advanced Site Settings

11. Under the SSH section, go to Authentication. You will be using the Private key file for authentication instead of a password. Browse for your private key file under Authentication parameters. This private key file (.ppk) is the file you have just generated from PuTTYgen. Click "OK" after loading your private key file to close the popup.

12. For the User name, put as ec2-user and click Login.

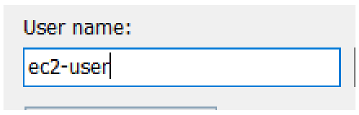
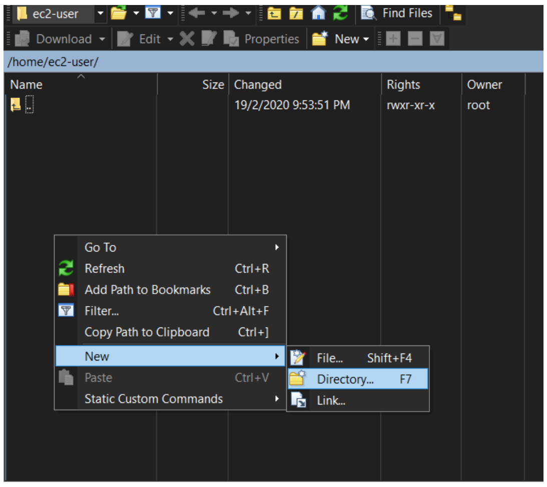

User name: ec2-user

Figure 59: User Name in WinSCP

13. You should be logged into your ec2 instance now, and WinSCP will show you the ec2-user folder. Create a directory under the ec2-user directory. Right-click in the directory, under New, there should be a Directory option.

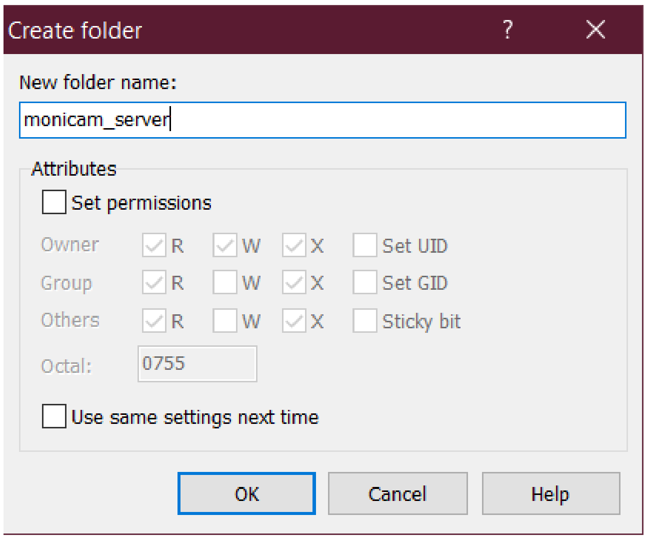


/home/ec2-user/

Name	Size	Changed	Rights	Owner
..		19/2/2020 9:53:51 PM	rwxr-xr-x	root

Figure 60: Create Directory

14. Name the folder as something identifiable or as monicam_server.


Create folder

New folder name: monicam_server

Attributes

[ ] Set permissions

Owner	✓ R	✓ W	✓ X	Set UID
Group	✓ R	W	✓ X	Set GID
Others	✓ R	W	✓ X	Sticky bit

Octal: 0755

[ ] Use same settings next time

Figure 61: Naming Folder

15. Drag server.py, requirements.txt, private.pem.key file, certificate.pem.crt, rootca.pem, static and template folders into the directory you have just created.

16. After all the files have been transferred, go to the ec2-user directory, and create a .aws folder there. Create a file called credentials and open it.

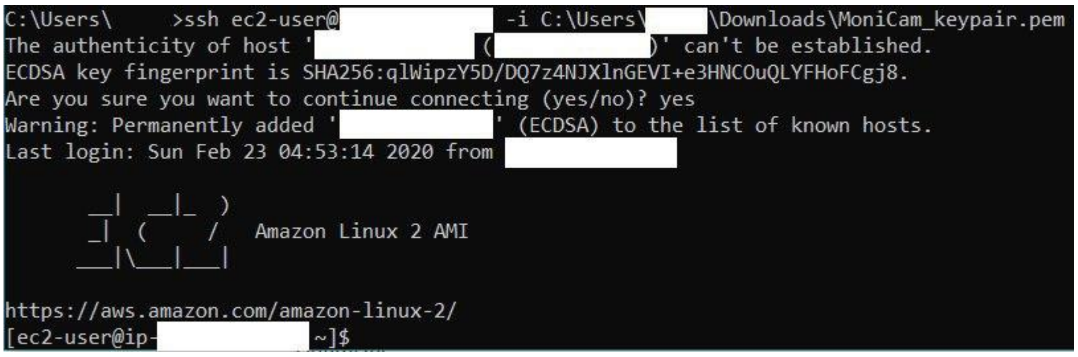
17. Paste the credentials you have copied in “AWS Console” section and save the file.

Set Up to EC2 in Raspberry Pi

1. Open a new terminal in your Raspberry Pi.

2. Execute the following command to SSH into your EC2 instance.

```
ssh ec2-user@<your EC2 public IP address> -i <location of your
.pem key pair file>
```



```
C:\Users\     >ssh ec2-user@[redacted] -i C:\Users\[redacted]\Downloads\MoniCam_keypair.pem
The authenticity of host '[redacted] ([redacted])' can't be established.
ECDSA key fingerprint is SHA256:qlWipzY5D/DQ7z4NJXlnGEVI+e3HNCOuQLYFHoFCgj8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[redacted]' (ECDSA) to the list of known hosts.
Last login: Sun Feb 23 04:53:14 2020 from [redacted]

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-[redacted] ~]$
```

Figure 62: SSH into EC2 Instance

If you are prompted whether to trust the source or not, type “Yes”.

3. cd into your project folder directory and run the following commands.

```
sudo yum update
sudo yum install python-pip
pip install -r requirements.txt
```

4. Your server is all set up!
You can run the server program by using <python server.py> and access the web interface at <your_EC2_ip_addr>:5000

Creating DynamoDB Tables

1. In AWS Console, search for “DynamoDB”.

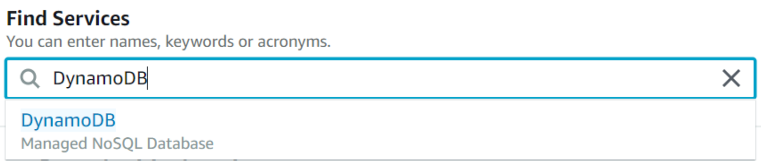
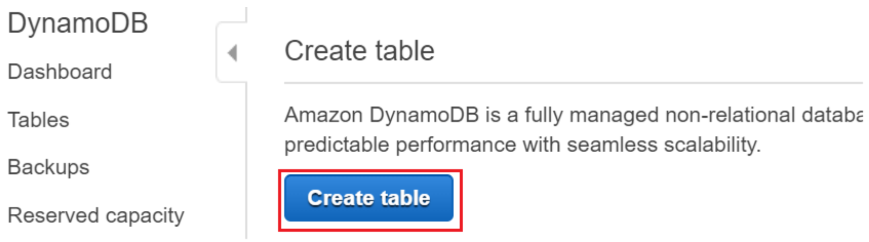


Figure 63: AWS Console

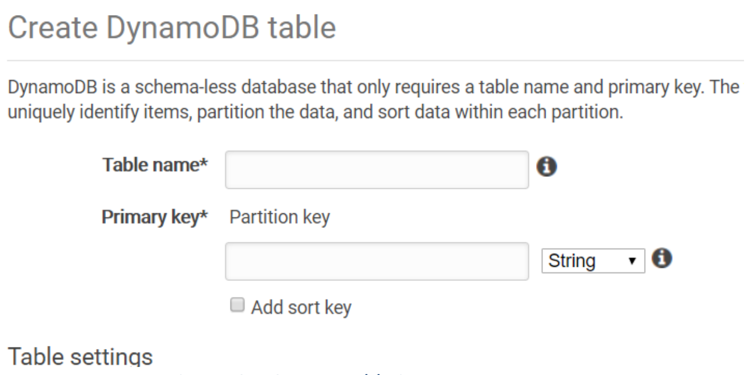
2. In DynamoDB, click “Create table”.



Amazon DynamoDB is a fully managed non-relational databa
predictable performance with seamless scalability.

Figure 64: Create Table in DynamoDB

3. You will come to the following screen. Create the following tables with the primary and sort keys. Check the add sort key box to be able to add the sort keys. Column variable types are all string. Once you are done with popluating the fields, click create to create the table. You are done with the creation of tables!



Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The
uniquely identify items, partition the data, and sort data within each partition.

Table name*

Primary key* Partition key

String

Add sort key

Table settings

Figure 65: Create Table in DynamoDB

	lightmotion_tb	attempt_tb	user_info
Partition Key	deviceid	deviceid	deviceid
Sort Key	motion_datetime	attempt_datetime	id

Section 5
Source codes

server.py

```
from flask import Flask, render_template, jsonify, request,Response,redirect,
send_from_directory
import sys

import json
import numpy

import decimal

import gevent
import gevent.monkey

from gevent.pywsgi import WSGIServer
from datetime import datetime
from time import sleep
# Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import boto3
from boto3.dynamodb.conditions import Key, Attr
gevent.monkey.patch_all()

import string, random

deviceid = "monicam"
#######################################################################################
# for mqtt and db functions

host = "" # <replace> with your AWS Endpoint
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

callb_msgpayload = {}



# random string generator
def rand_str_gen(size=10):
  lettersal = ''.join(random.choice(string.ascii_letters) for i in range(size))
```

```
    lettersd = ''.join(random.choice(string.digits) for i in range(size))
    lettersp = ''.join(random.choice(string.punctuation) for i in range(size))
    letter = str(lettersal) + str(lettersd) + str(lettersp)
    return ''.join(random.choice(letter) for i in range(size))

# Custom MQTT message callback
def customCallback(client, userdata, message):
    global callb_msgpayload
    callb_msgpayload = json.loads(message.payload)

# --------------------------------------------------------#

def publisher(topic, message):
    server_mqtt = AWSIoTMQTTClient("monicamMQTT_pub" + rand_str_gen())
    server_mqtt.configureEndpoint(host, 8883)
    server_mqtt.configureCredentials(rootCAPath, privateKeyPath, certificatePath)
    server_mqtt.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
    server_mqtt.configureDrainingFrequency(2)  # Draining: 2 Hz
    server_mqtt.configureConnectDisconnectTimeout(10)  # 10 sec
    server_mqtt.configureMQTTOperationTimeout(5)  # 5 sec
    server_mqtt.connect()
    server_mqtt.publish(topic , json.dumps(message), 1)
    sleep(2)
    server_mqtt.disconnect()

def subscriber(topic):
    server_mqtt = AWSIoTMQTTClient("monicamMQTT_sub" + rand_str_gen())
    server_mqtt.configureEndpoint(host, 8883)
    server_mqtt.configureCredentials(rootCAPath, privateKeyPath, certificatePath)
    server_mqtt.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
    server_mqtt.configureDrainingFrequency(2)  # Draining: 2 Hz
    server_mqtt.configureConnectDisconnectTimeout(10)  # 10 sec
    server_mqtt.configureMQTTOperationTimeout(5)  # 5 sec
    server_mqtt.connect()
    server_mqtt.subscribe(topic, 1, customCallback)
    sleep(2)
    server_mqtt.unsubscribe(topic)
    server_mqtt.disconnect()

# Helper class to convert a DynamoDB item to JSON.
class DecimalEncoder(json.JSONEncoder):
    def default(self, o):
        if isinstance(o, decimal.Decimal):
            if o % 1 > 0:
                return float(o)
            else:
                return int(o)
        return super(DecimalEncoder, self).default(o)

class GenericEncoder(json.JSONEncoder):

    def default(self, obj):
        if isinstance(obj, numpy.generic):
            return numpy.asscalar(obj)
        elif isinstance(obj, decimal.Decimal):
            return str(obj)
        elif isinstance(obj, datetime):
            return obj.strftime('%Y-%m-%d %H:%M:%S')
        elif isinstance(obj, decimal.Decimal):
            return float(obj)
        else:
            return json.JSONEncoder.default(self, obj)

def data_to_json(data):
    json_data = json.dumps(data,cls=GenericEncoder)
    print(json_data)
    return json_data

# https://stackoverflow.com/a/55734992
# converts unicode dict to utf-8
def utfy_dict(dic):
    if isinstance(dic,unicode):
        return(dic.encode("utf-8"))
    elif isinstance(dic,dict):
        for key in dic:
            dic[key] = utfy_dict(dic[key])
        return(dic)
    elif isinstance(dic,list):
        new_l = []
        for e in dic:
            new_l.append(utfy_dict(e))
        return(new_l)
    else:
        return(dic)

class DynamoDB_class():
    def __init__(self):

            self.dynamodb = boto3.resource('dynamodb', region_name='us-east-1')

    # def get_item(self, condition, tb_name, part_name, col_name, search_str, limit=0):
    def get_item(self, tb_name, keycondexpress, limit=0):
        try:
            print("get_item function")
            print(tb_name)
            print(limit)

            table = self.dynamodb.Table(tb_name)

            # keycondexpress = ""
            # replace deviceid with your partition key name
            # if condition == "equals":
            #           # keycondexpress=Key('deviceid').eq(deviceid) & Key(col_name).eq(search_str)
            #
            # elif condition == "begins_with":
            #           # keycondexpress=Key('deviceid').eq(deviceid) &
Key(col_name).begins_with(search_str)
            #
            # elif condition == "between":
            #           # keycondexpress=Key('deviceid').eq(deviceid) &
Key(col_name).between(search_str)

            # Query requires a partition key (aka (usually) your first column in the table) (for smol amts
of data)
            # BatchGetItems requires a primary key (aka your partition key + sort key or just partition
key if dh sort key) (for large amts of data)
            # Further reading: https://stackoverflow.com/questions/30749560/whats-the-difference-
between-batchgetitem-and-query-in-dynamodb/30772172
            response = table.query(
                KeyConditionExpression=keycondexpress,
                ScanIndexForward=False
            )
            items = response['Items']
            if limit == 0:
                data = items
            else:
                data = items[:limit] # limit to specified amt of items
            data_reversed = data[::-1]
            data_reversed_c = []
            for i in data_reversed:
              data_reversed_c.append(utfy_dict(i))
            return data_reversed_c
        except:
            import sys
            print(sys.exc_info()[0])
            print(sys.exc_info()[1])

    def add_item(self, tb_name, item):
            table = self.dynamodb.Table(tb_name)

            response = table.put_item(
              Item=item
            )

            print("PutItem succeeded:")
            print(json.dumps(response, indent=4, cls=DecimalEncoder))

    def updating_item(self, tb_name, key, update_express, express_attr_values):
            table = self.dynamodb.Table(tb_name)
        # Usage:
        # key = {
        #    "deviceid": "deviceid_dorachua",
        #    "datetimeid": "2020-01-22T18:53:30.459146"
        # }
        # update_express = 'set item_value = :v'
        # express_attr_values = {
        #    ':v': 100
        # }
        # if express_attr_names == '':
        response = table.update_item(
            Key=key,
            UpdateExpression=update_express,
            ExpressionAttributeValues=express_attr_values
        )

        # Usage:
        # key = {
        #    "deviceid": "deviceid_dorachua",
        #    "datetimeid": "2020-01-22T18:53:30.459146"
        # }
        # update_express = 'set #v = :v'
        # express_attr_values = {
        #    ':v': 100
        # }
        # express_attr_names = {
        #    '#v': "value"
        # }
        # else:
        #     response = table.update_item(
        #         Key=key,
        #         UpdateExpression=update_express,
        #         ExpressionAttributeValues=express_attr_values,
        #         ExpressionAttributeNames=express_attr_names
        #     )

            print("UpdateItem succeeded:")
            print(json.dumps(response, indent=4, cls=DecimalEncoder))
        def deleting_item(self, tb_name, del_key):
                print("Attempting a conditional delete...")
                try:
                        table = self.dynamodb.Table(tb_name)

                        # Usage
                        # key = {
                        #       "deviceid": "deviceid_dorachua",
                        #       "datetimeid": "2020-01-22T18:53:31.756004"
                        # }

                        response = table.delete_item(
                    Key=del_key
                    # ConditionExpression="info.rating <= :val", # do not delete item if doesn't
meet condition
                    # ExpressionAttributeValues= {
                    #    col_name: del_key
                    # }
                  )
                except ClientError as e:
                  if e.response['Error']['Code'] == "ConditionalCheckFailedException":
                    print(e.response['Error']['Message'])
                  else:
                    raise Exception
                else:
                  print("DeleteItem succeeded:")
                  print(json.dumps(response, indent=4, cls=DecimalEncoder))

######################################################################################
#
db = DynamoDB_class()

def get_userid():
  print("getting userid")
  user_info_list = []
  dict = {}
  user_list = []
  try:
    keycondexpress = Key("deviceid").eq(deviceid)
    user_info_list = db.get_item("user_info", keycondexpress)
    # print(user_info_list)
    # for i in user_info: # converting to utf-8 dict from unicode
    #    user_info_list.append(utfy_dict(i))
    for user in user_info_list:
      user_list.append(user["id"])
    # print(user_info_list)
    # print(user_list)
```

```
        return user_list
    except:
        print(sys.exc_info()[0])
        print(sys.exc_info()[1])


app = Flask(__name__)

########################################################################################
# for getting historical data

# for serving the graph page
@app.route("/charts.html")
def charts():
    print("getting chart webpage")
    return render_template("charts.html")

# for the all time motion graph data
@app.route("/api/get_alltime_motion_graph_data",methods = ['POST', 'GET'])
def apidata_getalltimemotiongraphdata():
    print("getting all time motion data")
    if request.method == 'GET':
        try:
            keycondexpress=Key('deviceid').eq(deviceid)
            dbdata = db.get_item("lightmotion_tb", keycondexpress)
            prev_motion_date = ''
            cur_motion_date = ''
            motion_count = 0
            dict = {}
            motion_array = []
            for i in dbdata:
                cur_motion_date = (i.get('motion_datetime', None))[0:10]
                if prev_motion_date == '':
                    prev_motion_date = cur_motion_date
                    motion_count = 1
                elif prev_motion_date == cur_motion_date:
                    motion_count += 1
                else:
                    dict = {
                        'date': prev_motion_date,
                        'motion': motion_count
                    }
                    motion_array.append(dict)
                    prev_motion_date = cur_motion_date
                    motion_count = 1
            dict = {
                'date': prev_motion_date,
            'motion': motion_count
          }
          motion_array.append(dict)
          data = {'chart_data': data_to_json(motion_array), 'title': "Historical LightMotion Graph
Data"}
          return jsonify(data)
       except:
          print(sys.exc_info()[0])
          print(sys.exc_info()[1])

# for the all time motion table data
@app.route("/api/get_alltime_motion_table_data",methods = ['POST', 'GET'])
def apidata_getalltimemotiontabledata():
    print("getting all time motion data")
    if request.method == 'GET':
       try:
          keycondexpress=Key('deviceid').eq(deviceid)
          dbdata = data_to_json(db.get_item("lightmotion_tb", keycondexpress))
          data = {'chart_data': dbdata, 'title': "Historical LightMotion Table Data"}
          return jsonify(data)
       except:
          print(sys.exc_info()[0])
          print(sys.exc_info()[1])

# for the all time attempt graph
@app.route("/api/get_alltime_attempt_graph_data",methods = ['POST', 'GET'])
def apidata_getattemptgraphdata():
    print("getting all time attempt data")
    if request.method == 'GET':
       try:
          keycondexpress=Key('deviceid').eq(deviceid)
          dbdata = db.get_item("attempt_tb", keycondexpress)
          prev_attempt_date = ''
          cur_attempt_date = ''
          attempt_count = 0
          dict = {}
          attempt_array = []
          for i in dbdata:
             cur_attempt_date = (i.get('attempt_datetime', None))[0:10]
             if prev_attempt_date == '':
                prev_attempt_date = cur_attempt_date
                attempt_count = 1
             elif prev_attempt_date == cur_attempt_date:
                attempt_count += 1
             else:
                dict = {
                   'date': prev_attempt_date,
                   'attempt': attempt_count
                }
                attempt_array.append(dict)
                prev_attempt_date = cur_attempt_date
                attempt_count = 1
          dict = {
             'date': prev_attempt_date,
             'attempt': attempt_count
          }
          attempt_array.append(dict)
          data = {'chart_data': data_to_json(attempt_array), 'title': "Historical Attempt Graph Data"}
          return jsonify(data)
       except:
          print(sys.exc_info()[0])
          print(sys.exc_info()[1])

# for the all time attempt table
@app.route("/api/get_alltime_attempt_table_data",methods = ['POST', 'GET'])
def apidata_getattempttabledata():
    print("getting all time attempt data")
    if request.method == 'GET':
       try:
          keycondexpress=Key('deviceid').eq(deviceid)
          dbdata = data_to_json(db.get_item("attempt_tb", keycondexpress))
          data = {'chart_data': dbdata, 'title': "Historical Attempt Table Data"}
          return jsonify(data)
       except:
          print(sys.exc_info()[0])
          print(sys.exc_info()[1])

##########################################################################################
# for rfid forms

@app.route("/forms.html",methods = ['POST', 'GET'])
def username():
    userid = get_userid()
    return render_template("forms.html", userid=userid)

@app.route("/api/get_user_data", methods = ['POST', 'GET'])
def get_user():
    print("getting user data for table")
    if request.method == 'POST':
       try:
          keycondexpress=Key('deviceid').eq(deviceid)
          dbdata = data_to_json(db.get_item("user_info", keycondexpress))
          data = {'chart_data': dbdata, 'title': "User Info Table"}
          return jsonify(data)
```

```
        except:
            print(sys.exc_info()[0])
            print(sys.exc_info()[1])

@app.route("/form_add_user", methods = ['POST', 'GET'])
def add_user():
    print("recieving data from add user form")
    add_feedback = "adding user data was not successful!"
    if request.method == "POST":
        print(request.form)
        missing = list()

        for k, v in request.form.items():
            if v == "":
                missing.append(k)

        if missing:
            missing_str = ""
            for i in missing:
                missing_str = missing_str + ", " + i
            feedback = "Missing field(s) for {}".format(missing_str)
        else:
            Rfid_ID = request.form.get("add_Rfid_ID")
            username = request.form.get("add_username")
            userid = get_userid()
            print(userid)
            if userid == []:
                userid = 'u1'
            else:
                userid = 'u' + str(len(userid) + 1)
            date_time = str(datetime.now().replace(microsecond=0))
            item = {}
            item["deviceid"] = deviceid
            item["id"] = userid
            item["rfid_id"] = Rfid_ID
            item["username"] = username
            item["modification_date"] = date_time
            db.add_item("user_info", item)
            add_feedback = "Successfully added user entry!"
    userid = get_userid()
    return render_template("forms.html", add_feedback=add_feedback, userid=userid)

@app.route("/form_modify_user", methods = ['POST', 'GET'])
def modify_user():
    print("recieving data from modify user form")
    modify_feedback = "modifying user data was not successful!"
    if request.method == "POST":
        id = request.form.get("mod_id")
        Rfid_ID = request.form.get("mod_Rfid_ID")
        username = request.form.get("chg_username")
        date_time = str(datetime.now().replace(microsecond=0))

        missing = list()

        for k, v in request.form.items():
            if v == "":
                missing.append(k)

        if missing:
            missing_str = ""
            for i in missing:
                missing_str = missing_str + ", " + i
            feedback = "Missing field(s) for {}".format(missing_str)
        else:
            key={
             "deviceid": deviceid,
             "id": id,
            }
            update_express='set rfid_id = :rid, username = :uname, modification_date = :mod_date'
            express_attr_values={
             ':rid': Rfid_ID,
             ':uname': username,
                ':mod_date': date_time
            }
            db.updating_item("user_info", key, update_express, express_attr_values)
            modify_feedback = "Successfully edited user entry!"
    userid = get_userid()
    return render_template("forms.html", modify_feedback=modify_feedback, userid=userid)

@app.route("/form_delete_user", methods = ['POST', 'GET'])
def delete_user():
    print("recieving data from delete user form")
    delete_feedback = "deleting user data was not successful!"
    if request.method == "POST":
        id = request.form.get("del_id")

        missing = list()

        for k, v in request.form.items():
            if v == "":
                missing.append(k)

        if missing:
            missing_str = ""
            for i in missing:
                missing_str = missing_str + ", " + i
            delete_feedback = "Missing field(s) for {}".format(missing_str)
        else:
            key = {
             'deviceid': deviceid,
                'id': id
            }
            print("Attempting a conditional delete...")
            dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
            try:
             table = dynamodb.Table("user_info")

             # Usage
             # key = {
             #         "deviceid": "deviceid_dorachua",
             #         "datetimeid": "2020-01-22T18:53:31.756004"
             # }

             response = table.delete_item(
                  Key=key
                )
            except ClientError as e:
                if e.response['Error']['Code'] == "ConditionalCheckFailedException":
                  print(e.response['Error']['Message'])
                else:
                  raise Exception
            else:
                print("DeleteItem succeeded:")
                print(json.dumps(response, indent=4, cls=DecimalEncoder))
            delete_feedback = "Successfully deleted user entry!"
    userid = get_userid()
    return render_template("forms.html", delete_feedback=delete_feedback, userid=userid)

##########################################################################################
#
# for index page content

# for displaying the index page
@app.route("/")
@app.route("/index.html")
def index():
    print("getting the index page")
    return render_template('index.html')

############################################################
# Get time configuration
# to show the most recent setting of start time range for the day
@app.route('/api/startTimeRange', methods=['POST', 'GET'])
def getStartTimeRange():
   timerange = []
   if request.method == 'GET':
      try:
         keycondexpress=Key("deviceid").eq(deviceid) &
Key("timeconf_datetime").begins_with(datetime.today().strftime('%Y-%m-%d'))
         startrange = db.get_item("timeconfig", keycondexpress, limit=1)
         if startrange == []:
            return jsonify("None")
         else:
            return jsonify(startrange[0]["starttime"])
      except:
         print(sys.exc_info()[0])
         print(sys.exc_info()[1])

# to show the most recent setting of end time range for the day
@app.route('/api/endTimeRange', methods=['POST', 'GET'])
def getEndTimeRange():
   timerange = []
   if request.method == 'GET':
      try:
         keycondexpress=Key("deviceid").eq(deviceid) &
Key("timeconf_datetime").begins_with(datetime.today().strftime('%Y-%m-%d'))
         endrange = db.get_item("timeconfig", keycondexpress, limit=1)
         if endrange == []:
            return jsonify("None")
         else:
            return jsonify(endrange[0]["endtime"])
      except:
         print(sys.exc_info()[0])
         print(sys.exc_info()[1])

# to get user input from HTML and store into DB
@app.route('/api/getTimeConfig', methods=['POST', 'GET'])
def getTimeConfig():
   if request.method == 'POST':
      starttime = str(request.form.get('starttime'))
      endtime = str(request.form.get('endtime'))
      timeconf_datetime = str(datetime.now().replace(microsecond=0)).replace(" ", "_")
      item={
         "deviceid": deviceid,
         "timeconf_datetime": timeconf_datetime,
         "starttime": starttime,
         "endtime": endtime
```

```
        }
        db.add_item("timeconfig", item)
        return redirect('/')
    else:
        abort(405)


##############################################
# For mqtt calls for alarm and live data

# for ringing the alarm
@app.route("/api/alert/alarm", methods=['POST'])
def apialarm():
    try:
        print("ringing the alarm")
        message={}
        message["alarm"] = 1
        publisher("api/alert/alarm", message)
    except:
        print(sys.exc_info()[0])
        print(sys.exc_info()[1])


# for getting the motion status
@app.route("/api/motionstatus", methods=['GET'])
def apimotionstatus():
    try:
        print("checking motion status")
        subscriber("sensors/lightmotionvalue")
        global callb_msgpayload
        motion = callb_msgpayload.get("motionvalue", "nth")
        print(motion)
        status_str = ''
        if motion == '1':
            status_str = "Motion detected!"
        else:
            status_str = "No motion detected."
        return jsonify({'status': status_str})
    except:
        print(sys.exc_info()[0])
        print(sys.exc_info()[1])


# for getting the light intensity
@app.route("/api/lightvalue", methods=['GET'])
def apilightstatus():
    try:
        print("checking light intensity")
        subscriber("sensors/lightmotionvalue")
        global callb_msgpayload
        light_value = callb_msgpayload.get("lightvalue", "nth")
        status_str = ''
        if light_value == "nth":
            status_str = "No light value"
        else:
            status_str = light_value
        return jsonify({'status': status_str})
    except:
        print(sys.exc_info()[0])
        print(sys.exc_info()[1])

##########################################################################################
# main
def main():
    try:
         http_server = WSGIServer(('0.0.0.0', 5000), app)
         app.debug = True
         http_server.serve_forever()
         print('Server waiting for requests')
    except KeyboardInterrupt:
        sys.exit()
    except:
         print("Exception")
         print(sys.exc_info())

if __name__ == '__main__':
   main()
```

monicam_main.py

```
import serial # for communication with arduino
import mysql.connector # for communication with phpmysql db
from time import sleep # for sleep (delay) function
import time # for timing operations
from datetime import datetime # for getting current date and time
import sys # for any system functions such as error msgs
import RPi.GPIO as GPIO # for operations requiring GPIO pins
import MFRC522 # for RFID reader
import signal # to capture keyboard interrupts
from gpiozero import Buzzer, MotionSensor, LED # for buzzer, motion sensor and led
from io import BytesIO # an object to hold image bytes
import os # for working with files and directories
import re # for working with regular expressions
from utils import camera, buzzer, lcd_display, rfid_reader, DynamoDB_class, rand_str_gen # for
all the shared functions
from telegrambot import send_user_Msg # for sending user image msg
# for setting queries
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from boto3.dynamodb.conditions import Key, Attr
import boto, botocore, boto3
import json
from multiprocessing import Process
import serial
# declaring the LED GPIO pins
yellowLED = LED(20)
greenLED = LED(21)

deviceid = "monicam"
prog_run = False
#################################################################################
#
# For communication with the server.py via MQTT

# Setting AWS endpoint and filepaths of CAs and key
host = "" # <replace> with your AWS Endpoint
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

callb_msgpayload = {} # global variable that stores the current callback message

# Custom MQTT message callback
def customCallback(client, userdata, message):
   global callb_msgpayload
   callb_msgpayload = json.loads(message.payload)
   # print("cust callback")
   # print(callb_msgpayload)

# ----------------------------------------------------------#

def publisher(topic, message):
   monicam_rpi = AWSIoTMQTTClient("monicamMQTT_pub" + rand_str_gen())
   monicam_rpi.configureEndpoint(host, 8883)
   monicam_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)
   monicam_rpi.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
   monicam_rpi.configureDrainingFrequency(2)  # Draining: 2 Hz
   monicam_rpi.configureConnectDisconnectTimeout(10)  # 10 sec
   monicam_rpi.configureMQTTOperationTimeout(5)  # 5 sec
   monicam_rpi.connect()
```

```
    monicam_rpi.publish(topic , json.dumps(message), 1)
    sleep(2)
    monicam_rpi.disconnect()

def subscriber(topic):
    monicam_rpi = AWSIoTMQTTClient("monicamMQTT_sub" + rand_str_gen())
    monicam_rpi.configureEndpoint(host, 8883)
    monicam_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)
    monicam_rpi.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
    monicam_rpi.configureDrainingFrequency(2)  # Draining: 2 Hz
    monicam_rpi.configureConnectDisconnectTimeout(10)  # 10 sec
    monicam_rpi.configureMQTTOperationTimeout(5)  # 5 sec
    monicam_rpi.connect()
    monicam_rpi.subscribe(topic, 1, customCallback)
    sleep(2)
    monicam_rpi.unsubscribe(topic)
    monicam_rpi.disconnect()
#####################################################################################
#
# The following chunk of code is for image recognition

dir_path = "/home/pi/motion_captures/"
# To store images to s3
def s3_imgstore(filepath):
    global img_bucket
    global dir_path
    # Create an S3 resource
    s3 = boto3.resource('s3', region_name='us-east-1')

    # Set the bucket name
    bucket = '' # <replace> with the bucket containing the attempts by people
    exists = True

    try:
        s3.meta.client.head_bucket(Bucket=bucket)
    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,CreateBucketConfiguration={
        'LocationConstraint': 'us-east-1'})

    # Upload a new file
    s3.Object(bucket, filepath).put(Body=open(dir_path + filepath, 'rb'))
    print("File uploaded")
```

```
# Sends the request to AWS Rekognition to get the specified photos and then compare them to
see if they match or not
def s3RekogniseFace(username, attempt_img):
    global img_bucket
    global dir_path
    client = boto3.client('rekognition', region_name='us-east-1')
    face_matched = False
    response = client.compare_faces(
        SimilarityThreshold=90,
        SourceImage={
            'S3Object': {
                'Bucket': '', # <replace> with the bucket containing your authenticated users' faces
                'Name': username + '.jpeg'
            }
        },
        TargetImage={
            'S3Object': {
                'Bucket': '', # <replace> with the bucket containing the attempts by people
                'Name': attempt_img
            }
        }
    )
    if (not response['FaceMatches']):
        face_matched = False
    else:
        face_matched = True
        for faceMatch in response['FaceMatches']:
            position = faceMatch['Face']['BoundingBox']
            similarity = str(faceMatch['Similarity'])
            print('The face at ' +
                   str(position['Left']) + ' ' +
                   str(position['Top']) +
                   ' matches with ' + username + ' with ' + similarity + '% confidence')

    return face_matched, response

#####################################################################################
#
# To control the led lighting
def ledControl(yellow, green):
    if 'off' in yellow:
        yellowLED.off()
    elif 'on' in yellow:
        yellowLED.on()

    if 'off' in green:
        greenLED.off()
    elif 'on' in green:
        greenLED.on()

####################################################################################

# for telegram alert
def telebotalert(command, data):
    print "sending telegram alert"
    send_user_Msg(command, data)

####################################################################################

# for logging the access attempt by publishing message to api/alert/attempt
def attempt_logger(attempt_img, date_time, rfid_id, username, success):
    print "logging attempt"
    if str(username).find('[]') != -1:
        username = ""
    message = {}
    message["deviceid"] = deviceid
    message["attempt_datetime"] = date_time
    message["rfid_id"] = rfid_id
    message["username"] = username
    message["success"] = success
    message["attempt_img_path"] = attempt_img
    publisher("api/alert/attempt", message)

# query for the existence of the rfid id
def query_rfid(rfid_id):
    print "querying rfid id"
    keycondexpress=Key('deviceid').eq(deviceid)
    db = DynamoDB_class()
    users = db.get_item("user_info", keycondexpress)
    user_rfid_id = ''
    username = None
    if users != None:
        for i in users:
            user_rfid_id = i.get('rfid_id', None)
            if user_rfid_id == rfid_id:
                username = i.get('username', None)
    return username

# check if rfid card is valid
def rfid_checker():
    global prog_run
    print "checking rfid\n"
    lcd_display(["Please present", "RFID card"])
```

```
attempt = 0
username = None
success = 'No'
date_time = ''
attempt_img = ''
while username == None or str(username).find('[]') != -1 or success == 'No':
    # check for number of attempts
    # if over 3 failed attempts ring buzzer
    if attempt < 4:
        attempt = attempt + 1
        rfid_id = rfid_reader()
        # call motion_sensor function if no card in 60 seconds
        if rfid_id == "timeout":
            print("rfid read timeout!")
            prog_run = False
            break
        else:
            date_time = str(datetime.now().replace(microsecond=0)).replace(" ", "_")
            username = query_rfid(rfid_id)
            print username
            if username == None or str(username).find('[]') != -1:
                if attempt < 4:
                    lcd_display(["Please try", "again!"])
            else:
                lcd_display(["Please wait.", "Scanning face."])
                attempt_img = camera(date_time)
                s3_imgstore(attempt_img)
                sleep(10)
                face_matched, response = s3RekogniseFace(username, attempt_img)
                if face_matched:
                    success = 'Yes'
                    print("User " + username + " has authenticated successfully!")
                else:
                    lcd_display(["Unauthorised", "User!"])
                    print("WARNING - UNAUTHORIZED USER is using " + username + "\'s card! ")
                    telebotalert("uauth_user", username)
            attempt_logger(attempt_img, date_time, rfid_id, username, success)
    elif attempt == 4:
        telebotalert("failed_attempts", '')
        lcd_display(["UNAUTHORIZED!", "ALERT TRIGGERED!"])
        buzzer() # make buzzer sound
        prog_run = False
        break
print "username is => " + str(username)
if success == 'Yes':
    ledControl('off', 'on')
    telebotalert("success", username)
```

```
        lcd_display(["Welcome home", str(username) + "!"])
        sleep(5)
        prog_run = False

# sends message to mqtt topic sensors/lightmotionvalimg
def add_motion_to_db(light_val, motion_sig, date_time):
    print "adding motion to db\n"
    motion_img = camera(date_time)
    message={
        "deviceid": deviceid,
        "motion_datetime": date_time,
        "lightvalue": light_val,
        "motionvalue": motion_sig,
        "motion_image": motion_img
    }
    publisher("sensors/lightmotionvalimg", message)

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    print "Ctrl+C captured, ending program."
    lcd_display(["clear"])
    GPIO.cleanup()
    sys.exit()

# check for motion by subscribing to the sensors/lightmotionvalue topic
def motion_checker():
    subscriber("sensors/lightmotionvalue")
    global prog_run
    global callb_msgpayload
    if callb_msgpayload.get("motionvalue", 0) == '1' and prog_run == False:
        prog_run = True
        ledControl('on', 'off')
        date_time = callb_msgpayload.get("datetimeid", "00-00-00_00:00:00")
        print "motion detected at " + date_time + "\n"
        add_motion_to_db(callb_msgpayload.get("lightvalue", 0), 1, date_time)
        rfid_checker()

def main():
    while True:
        lcd_display(["clear"])
        print "sensing for motion"
        ledControl('off', 'off')
        motion_checker()

if __name__ == '__main__':
    print "libaries import finished"
    signal.signal(signal.SIGINT, end_read)
    main()
```

lightmotion.py

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import json
from datetime import datetime
import serial
from time import sleep
from utils import buzzer, rand_str_gen

import string, random

from multiprocessing import Process

host = "" # <replace> with your AWS Endpoint
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

callb_msgpayload = {}

# Custom MQTT message callback
def customCallback(client, userdata, message):
    global callb_msgpayload
    callb_msgpayload = json.loads(message.payload)

# ---------------------------------------------------------#

# gets motion and light value from Arduino
def arduinoData():
    ser = serial.Serial("/dev/ttyUSB0", 9600) # open serial port
    # ser.baudrate = 9600
    data = ser.readline() # light value and motion value
    return data

def sendMotionLight():
    monicam_rpi = AWSIoTMQTTClient("monicamMQTT_pub" + rand_str_gen())
    monicam_rpi.configureEndpoint(host, 8883)
    monicam_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)
    monicam_rpi.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
    monicam_rpi.configureDrainingFrequency(2)  # Draining: 2 Hz
    monicam_rpi.configureConnectDisconnectTimeout(10)  # 10 sec
    monicam_rpi.configureMQTTOperationTimeout(5)  # 5 sec
    monicam_rpi.connect()
    while True:
        data = arduinoData().split(",")
        # print(data)
        lightValue = data[0]
        motionValue = data[1].strip()
        # lightValue = 404
        # motionValue = 1
        date_time = str(datetime.now().replace(microsecond=0)).replace(" ", "_")
        message = {}
        message["deviceid"] = "monicam"
        message["datetimeid"] = date_time
        message["lightvalue"] = lightValue
        message["motionvalue"] = motionValue
        print(message)
        monicam_rpi.publish("sensors/lightmotionvalue" , json.dumps(message), 1)


def main():
    sendMotionLight_proc = Process(name='sendMotionLight', target=sendMotionLight)
    sendMotionLight_proc.start()
    sendMotionLight_proc.join()

if __name__ == '__main__':
    main()
```

alarmdetector.py

```
import json
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from utils import rand_str_gen, buzzer
from multiprocessing import Process
from time import sleep

host = "" # <replace> with your AWS Endpoint
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

def alarmCallback(client, userdata, message):
    print(message.payload)
    if (json.loads(message.payload)).get("alarm", 0) == 1:
        buzzer()

def getAlarmStat():
    monicam_rpi = AWSIoTMQTTClient("monicamMQTT_sub" + rand_str_gen())
    monicam_rpi.configureEndpoint(host, 8883)
    monicam_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)
    monicam_rpi.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
    monicam_rpi.configureDrainingFrequency(2)  # Draining: 2 Hz
    monicam_rpi.configureConnectDisconnectTimeout(10)  # 10 sec
    monicam_rpi.configureMQTTOperationTimeout(5)  # 5 sec
    monicam_rpi.connect()
    monicam_rpi.subscribe("api/alert/alarm", 1, alarmCallback)
    while True:
        print("Alarm detection in progress...")
        sleep(5)

if __name__ == '__main__':
    getAlarmStat_proc = Process(name='getAlarmStat', target=getAlarmStat)
    getAlarmStat_proc.start()
    getAlarmStat_proc.join()
```

telegrambot.py

```
import telepot
from gpiozero import LED
from time import sleep
from datetime import datetime
import RPi.GPIO as GPIO # for operations requiring GPIO pins
import glob
from utils import camera, buzzer, lcd_display
import wget

my_bot_token = '' # <replace> with your bot token
bot = telepot.Bot(my_bot_token)
display = False
proxy_chat = False
proxy_chat_status = 'Inactive'
guest_chat_id = 0
owner_chat_id = # <replace> with your own chat id

def dateTimefunc():
    date_time = str(datetime.now()).replace(" ", "_")
    return date_time

def send_user_Msg(command, data=None):
    import telepot
    global owner_chat_id
    bot = telepot.Bot(my_bot_token)

    if command == 'failed_attempts':
        bot.sendMessage(owner_chat_id, '3 failed attempts have been detected!')
        bot.sendPhoto(owner_chat_id, photo=open("/home/pi/motion_captures/" +
camera(dateTimefunc()), 'rb'))
        print('Latest picture has been sent.')
    elif command == 'success':
        bot.sendMessage(owner_chat_id, data + ' is at home.')
    elif command == 'uauth_user':
        bot.sendMessage(owner_chat_id, 'Unauthorised user of ' + data + '\'s RFID card!')
        bot.sendPhoto(owner_chat_id, photo=open("/home/pi/motion_captures/" +
camera(dateTimefunc()), 'rb'))

def respondToMsg(msg):
    print(msg)
    file_id = ''
    chat_id = msg['chat']['id']
    command = (msg.get('text', 'msg_is_img')).lower()
    if command == 'msg_is_img':
        file_id = msg['photo'][0]['file_id']
    global display
    global guest_chat_id
    global proxy_chat
    global proxy_chat_status
    global owner_chat_id

    print('Got chat_id: {}'.format(chat_id))
    print('Got command: {}'.format(command))

    if 'Active' in proxy_chat_status:
        if chat_id == guest_chat_id:
            if 'exit()' in command:
                bot.sendMessage(owner_chat_id, 'Guest has left the chat.')
                bot.sendMessage(guest_chat_id, 'You have left the chat.')
                proxy_chat = False
                proxy_chat_status = 'Inactive'
                guest_chat_id = 0
            else:
                bot.sendMessage(owner_chat_id, command)
        elif chat_id == owner_chat_id:
            if 'exit()' in command:
                bot.sendMessage(guest_chat_id, 'Owner has left the chat.')
                bot.sendMessage(owner_chat_id, 'You have left the chat.')
                proxy_chat = False
                proxy_chat_status = 'Inactive'
                guest_chat_id = 0
            elif 'guestid' in command:
                bot.sendMessage(owner_chat_id, 'Current guest chat id is {}'.format(guest_chat_id))
            else:
                bot.sendMessage(guest_chat_id, command)
```

```
  elif proxy_chat == True and 'Inactive' in proxy_chat_status:
    if 'yes' in command.lower():
      bot.sendMessage(owner_chat_id, 'Connecting you to guest now...\n'+
      'To end convo with guest, please type \'exit()\'.\n'+
      'Type guestid to get the chat id of the current guest.')
      bot.sendMessage(guest_chat_id, 'Owner has accepted your chat request.\n'+
      'To end convo with owner, please type \'exit()\'.')
      proxy_chat_status = 'Active'
    elif 'no' in command.lower():
      bot.sendMessage(owner_chat_id, 'Communication request by guest declined.')
      bot.sendMessage(guest_chat_id, 'Communication request declined by owner.')
      guest_chat_id = 0
      proxy_chat = False
  # display help message to users
  elif command == '/help':
    bot.sendMessage(chat_id, 'Hello!\nI am a surveillance camera bot')
    if chat_id == owner_chat_id:
      bot.sendMessage(owner_chat_id, 'Here is a list of available commands you can ask me to
do:\n'+
      '/help - displays this help message.\n'+
      'pic / picture / image - takes a real time picture.\n'+
      'display / lcd - displays a message on the lcd screen.\n'+
      'alarm - sounds the alarm.\n'+
      'chat_id - gets your chat id with the bot.')
    else:
      bot.sendMessage(chat_id, 'Here is a list of available commands you can ask me to do:\n'+
      'chat_id - gets your chat id with the bot.\n'+
      'talk / owner / msg / message - talk to the owner.')
  # for all the owner allowed commands
  elif chat_id == owner_chat_id:
    if 'pic' in command or 'picture' in command or 'image' in command:
      print('Taking an image')
      bot.sendPhoto(owner_chat_id, photo=open("/home/pi/motion_captures/" +
camera(dateTimefunc()), 'rb'))
      bot.sendMessage(owner_chat_id, 'This is the real time ' + command)
    # for owner to display text on lcd with text split into 2 lines using newline
    # as a delimiter
    elif display == True:
      txt = command.split("\n")
      if len(txt) == 1:
        txt.append('')
      if 'exit()' in txt[0]:
        display = False
        bot.sendMessage(owner_chat_id, 'Exited lcd display mode.')
      else:
        lcd_display(txt)
    # for owner to display text on lcd display
```

```
        elif 'display' in command or 'lcd' in command:
            bot.sendMessage(owner_chat_id,
            'Please enter what you want to say to the visitor.\n' +
            'Only 16 characters per line; Maximum of 2 lines.\n' +
            'Press the enter key to split your strings into two lines\n' +
            'Eg.\n Hello\nWorld\n'+
            'Type exit() to exit sending text to lcd display.')
            display = True
        # for owner to sound the alarm
        elif 'alarm' in command:
            bot.sendMessage(owner_chat_id, 'Sounding the alarm now!\n'+
            'Please wait for alarm to stop ringing before inputting any more commands.')
            buzzer()
            bot.sendMessage(owner_chat_id, 'Alarm has finished ringing.')
        elif ('talk' in command or "owner" in command or 'msg' in command or 'message' in
command):
            bot.sendMessage(owner_chat_id, 'Displaying proxy chat instructions to guest')
            for x in range(2):
                lcd_display(['Hello guest!', 'Please use the'])
                sleep(2)
                lcd_display(['Telegram bot', 'monicam_bot'])
                sleep(2)
                lcd_display(['to communicate', 'with the owner'])
                sleep(2)
                lcd_display(['Start the bot', 'and type chat'])
                sleep(2)
                lcd_display(['to chat with', 'the owner'])
                sleep(2)
                lcd_display(['clear'])
        elif 'chat_id' in command:
            bot.sendMessage(chat_id, 'Your chat id is {}'.format(chat_id))
        # for all the other commands not here
        else:
            bot.sendMessage(chat_id, 'Unknown command. Please try again.\n' +
            'For a list of available commands, type /help.')
    # for getting the chat id of the current user
    elif 'chat_id' in command:
        bot.sendMessage(chat_id, 'Your chat id is {}'.format(chat_id))
    # for guest to communicate with the owner
    elif ('talk' in command or "owner" in command or 'msg' in command or 'message' in command)
and chat_id != owner_chat_id:
        # if chat_id == owner_chat_id:
        #     bot.sendMessage(owner_chat_id, 'Enter the chat id of the guest.')
        guest_chat_id = chat_id
        bot.sendMessage(owner_chat_id, 'guest with chat_id ' + str(guest_chat_id) +
        ' wants to talk to you. Do you accept or not? (Yes or No)')
        proxy_chat = True
    # for all the other commands not here
    else:
        bot.sendMessage(chat_id, 'Unknown command. Please try again.\n' +
        'For a list of available commands, type /help.')


def main():
    bot.message_loop(respondToMsg)
    # bot.sendMessage(chat_id, 'nanana')
    print('Listening for RPi commands...')

    while True:
        sleep(10)

if __name__ == '__main__':
    main()
```

utils.py

```
# this python file is for shared functions amongst other python files
from gpiozero import Buzzer, LED  # for buzzer
import RPi.GPIO as GPIO # for operations requiring GPIO pins
import os
from picamera import PiCamera # for picam
from rpi_lcd import LCD # for lcd
import MFRC522 # for RFID reader
import time # for timing operations
from time import sleep
import serial # to read arduino
import json

# Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import boto3
from boto3.dynamodb.conditions import Key, Attr

import string, random

def rand_str_gen(size=20):
    lettersal = ''.join(random.choice(string.ascii_letters) for i in range(size))
    lettersd = ''.join(random.choice(string.digits) for i in range(size))
    lettersp = ''.join(random.choice(string.punctuation) for i in range(size))
    letter = str(lettersal) + str(lettersd) + str(lettersp)
    return ''.join(random.choice(letter) for i in range(size))

# make the buzzer and led on and off to give the pulsating and blinking effect for 30 seconds
```

```
def buzzer():
    print "MAKE SOME NOISE!!!"
    bz = Buzzer(5) # buzzer at gpio 5
    redled = LED(19)
    t_end = time.time() + 30
    while time.time() < t_end:
        try:
            print "on buzzer and led"
            bz.on()
            redled.on()
            sleep(0.25)
            print "off buzzer and led"
            bz.off()
            redled.off()
            sleep(0.25)
        except Exception:
            bz.off()
            redled.off()

def camera(date_time):
    print "getting picture\n"
    # cam_stream = BytesIO(b"")
    camera = PiCamera()
    # camera.start_preview()
    sleep(2)
    # camera.capture(cam_stream, 'jpeg')
    dir_path = "/home/pi/motion_captures/"
    image_name = date_time + ".jpeg"

    if os.path.isdir(dir_path) == True:
        camera.capture(dir_path + image_name)
    else:
        try:
            os.mkdir(dir_path)
        except OSError:
            print ("Creation of the directory %s failed" % dir_path)
        else:
            print ("Successfully created the directory %s " % dir_path)

        camera.capture(dir_path + image_name)

    camera.close()
    print "stored image in " + dir_path + image_name
    return image_name

# function for all lcd display operations
# function takes in a list containing two strings
# and displays them in two seperate rows
def lcd_display(text_list):

    lcd = LCD()
    print "displaying"
    print text_list
    try:
        if 'clear' in text_list[0]:
            lcd.clear()
        else:
            lcd.text(text_list[0], 1)
            lcd.text(text_list[1], 2)
        sleep(1)
    except Exception:
        lcd.clear()

# read rfid card
def rfid_reader():

    print "reading rfid card"
    uid = None

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

    # This loop keeps checking for chips over 60 seconds.
    t_end = time.time() + 60
    while time.time() <= t_end: #wait for 60 seconds for rfid card

        # 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()
            # if uid!=prev_uid:
            #    prev_uid = uid
            print("UID of card is {}".format(uid))
            return str(uid)
    return "timeout" # if no card return string timeout

# Helper class to convert a DynamoDB item to JSON.
class DecimalEncoder(json.JSONEncoder):
    def default(self, o):
        if isinstance(o, decimal.Decimal):
            if o % 1 > 0:
                return float(o)
            else:
                return int(o)
        return super(DecimalEncoder, self).default(o)


# https://stackoverflow.com/a/55734992
# converts unicode dict to utf-8
def utfy_dict(dic):
    if isinstance(dic,unicode):
        return(dic.encode("utf-8"))
    elif isinstance(dic,dict):
        for key in dic:
            dic[key] = utfy_dict(dic[key])
        return(dic)
    elif isinstance(dic,list):
        new_l = []
        for e in dic:
            new_l.append(utfy_dict(e))
        return(new_l)
    else:
        return(dic)

class DynamoDB_class():
    def __init__(self):
             self.dynamodb = boto3.resource('dynamodb', region_name='us-east-1')

    # def get_item(self, condition, tb_name, part_name, col_name, search_str, limit=0):
    def get_item(self, tb_name, keycondexpress, limit=0):
        try:
            print("get_item function")
            print(tb_name)
            print(limit)

            table = self.dynamodb.Table(tb_name)
                        # deviceid = "deviceid_dorachua"

                        # keycondexpress = ""
                        # replace deviceid with your partition key name
                        # if condition == "equals":
                        #          # keycondexpress=Key('deviceid').eq(deviceid) &
Key(col_name).eq(search_str)
                        #
                        # elif condition == "begins_with":
                        #          # keycondexpress=Key('deviceid').eq(deviceid) &
Key(col_name).begins_with(search_str)
                        #
```

```
                    # elif condition == "between":
                    #         # keycondexpress=Key('deviceid').eq(deviceid) &
Key(col_name).between(search_str)

                    # Query requires a partition key (aka (usually) your first column in the table) (for
smol amts of data)
                    # BatchGetItems requires a primary key (aka your partition key + sort key or just
partition key if dh sort key) (for large amts of data)
                    # Further reading: https://stackoverflow.com/questions/30749560/whats-the-
difference-between-batchgetitem-and-query-in-dynamodb/30772172
         response = table.query(
          KeyConditionExpression=keycondexpress,
          ScanIndexForward=False
         )
         items = response['Items']
         if limit == 0:
               data = items
         else:
               data = items[:n] # limit to last n-th items
         data_reversed = data[::-1]
         data_reversed_c = []
         for i in data_reversed:
            data_reversed_c.append(utfy_dict(i))
         print("getitemresult_utils")
         print(data_reversed_c)
         print("end of getitemresult_utils")
         return data_reversed_c
      except:
         import sys
         print(sys.exc_info()[0])
         print(sys.exc_info()[1])

   def add_item(self, tb_name, item):
          table = dynamodb.Table(tb_name)

          response = table.put_item(
            Item=item
          )

          print("PutItem succeeded:")
          print(json.dumps(response, indent=4, cls=DecimalEncoder))

   # Usage:
   # key = {
   #      "deviceid": "deviceid_dorachua",
   #      "datetimeid": "2020-01-22T18:53:30.459146"
   # }
   # update_express = 'set item_value = :v'
   # express_attr_values = {
   #       ':v': 100
   # }
   # if express_attr_names=="":
         # def update_item(self, tb_name, key, update_express, express_attr_values,
express_attr_names):
   def update_item(self, tb_name, key, update_express, express_attr_values):
      self.table = self.dynamodb.Table(tb_name)
      response = self.table.update_item(
         Key=key,
         UpdateExpression=update_express,
         ExpressionAttributeValues=express_attr_values
      )

      # Usage:
      # key = {
      #   "deviceid": "deviceid_dorachua",
      #   "datetimeid": "2020-01-22T18:53:30.459146"
      # }
      # update_express = 'set #v = :v'
      # express_attr_values = {
      #   ':v': 100
      # }
      # express_attr_names = {
      #   '#v': "value"
      # }
      # else:
      #     response = table.update_item(
      #         Key=key,
      #         UpdateExpression=update_express,
      #         ExpressionAttributeValues=express_attr_values,
      #         ExpressionAttributeNames=express_attr_names
      #     )

      print("UpdateItem succeeded:")
      print(json.dumps(response, indent=4, cls=DecimalEncoder))

   def delete_item(self, tb_name, del_key):
         print("Attempting a conditional delete...")
         try:
               table = self.dynamodb.Table(tb_name)

               # Usage
               # key = {
               #    "deviceid": "deviceid_dorachua",
               #    "datetimeid": "2020-01-22T18:53:31.756004"
```

```
                # }

                response = table.delete_item(
                Key=del_key
                # ConditionExpression="info.rating <= :val", # do not delete item if doesn't meet
condition
                # ExpressionAttributeValues= {
                #     col_name: del_key
                # }
             )
         except ClientError as e:
             if e.response['Error']['Code'] == "ConditionalCheckFailedException":
                print(e.response['Error']['Message'])
             else:
                raise Exception
         else:
             print("DeleteItem succeeded:")
             print(json.dumps(response, indent=4, cls=DecimalEncoder))
```

arduinoHardware.ino

```
// # arduinoHardware.ino for IoT Assignment CA2

#define LDRpin A0 // pin where we connected the LDR and the resistor

int LDRValue = 0;     // result of reading the analog pin
int ledPin = 13;              // choose the pin for the LED
int motionPin = 2;              // choose the input pin (for PIR sensor)
int pirState = LOW;           // we start, assuming no motion detected
int motionVal = 0;                  // variable for reading the pin status


void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT);      // declare LED as output
  pinMode(motionPin, INPUT);     // declare motion sensor as input
  Serial.begin(9600); // sets serial port for communication
}

void loop() {
  LDRValue = analogRead(LDRpin); // read the value from the LDR
//  Serial.println(LDRValue);      // print the value to the serial port
  motionVal = digitalRead(motionPin);  // read input value
  Serial.println(String(LDRValue) + "," + String(motionVal));
//  Serial.println(motionVal);
  if (motionVal == HIGH) {           // check if the input is HIGH

    if (LDRValue < 500){ // check LDRValue
      digitalWrite(ledPin, HIGH);  // turn LED ON
    } else { // end of LDRValue check
      digitalWrite(ledPin, LOW);
    }

  } else {
    digitalWrite(ledPin, LOW); // turn LED OFF
  }

  delay(1000); // delay 3s

} // end of loop
```

index.html

```
<!-- index.html for IoT Assignment CA2 -->
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>AdminLTE 3 | Dashboard</title>
  <!-- Tell the browser to be responsive to screen width -->
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <!-- Font Awesome -->
  <link rel="stylesheet" href="../static/../static/plugins/fontawesome-free/css/all.min.css">
  <!-- Ionicons -->
  <link rel="stylesheet"
href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
  <!-- Tempusdominus Bbootstrap 4 -->
  <link rel="stylesheet" href="../static/plugins/tempusdominus-bootstrap-4/css/tempusdominus-
bootstrap-4.min.css">
  <!-- iCheck -->
  <link rel="stylesheet" href="../static/plugins/icheck-bootstrap/icheck-bootstrap.min.css">
  <!-- JQVMap -->
  <link rel="stylesheet" href="../static/plugins/jqvmap/jqvmap.min.css">
  <!-- Theme style -->
  <link rel="stylesheet" href="../static/dist/css/adminlte.min.css">
  <!-- overlayScrollbars -->
  <link rel="stylesheet" href="../static/plugins/overlayScrollbars/css/OverlayScrollbars.min.css">
  <!-- Daterange picker -->
  <link rel="stylesheet" href="../static/plugins/daterangepicker/daterangepicker.css">
  <!-- summernote -->
  <link rel="stylesheet" href="../static/plugins/summernote/summernote-bs4.css">
  <!-- Google Font: Source Sans Pro -->
  <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700"
rel="stylesheet">

  <!-- jQuery -->
  <script src="../static/plugins/jquery/jquery.min.js"></script>

  <!--
    Added Scripts for MoniCam
  -->

  <style>
    .button {
      padding: 5px 10px;
      font-size: 20px;
      text-align: center;
      cursor: pointer;
      outline: none;
      color: #fff;
      background-color: #FF8033;
      border: none;
      border-radius: 15px;
      box-shadow: 0 5px #999;
    }

    .button:hover {
      background-color: #FB5D03
    }

    .button:active {
      background-color: #FB5D03;
      box-shadow: 0 2px #666;
      transform: translateY(4px);
    }

    .formbutton {
      padding: 3px 5px;
      margin: 3px 7px;
      font-size: 13px;
      text-align: center;
      cursor: pointer;
      outline: none;
      color: #fff;
      background-color: #FF8033;
      border: none;
      border-radius: 15px;
      box-shadow: 0 3px #999;
    }

    .formbutton:hover {
      background-color: #FB5D03
    }

    .formbutton:active {
      background-color: #FB5D03;
      box-shadow: 0 1px #666;
      transform: translateY(4px);
    }
  </style>

  <style> #chartDiv {width:100%;}</style>
      <title>Light Intensity Graph</title>
      <script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.js"></script>
      <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

    <script>
      // checks in real time for motion
      function motion_stat(){
        jQuery.ajax({
          url: "/api/motionstatus" ,
          type: 'GET',
          success: function(data, textStatus, xhr){
            console.log(data)
            $("#motion_status").html(data.status);
          }//end success
        });//end ajax
      } //end

      // real time data of light intensity
      function light_val(){
        jQuery.ajax({
          url: "/api/lightvalue" ,
          type: 'GET',
          success: function(data, textStatus, xhr){
            console.log(data)
            $("#light_value").html(data.status);
          }//end success
        });//end ajax
      } //end

      // getting the start time from user configuration
      function startTimeRange(){
        jQuery.ajax({
          url: "/api/startTimeRange" ,
          type: 'GET',
          success: function(ndata, textStatus, xhr){
            console.log(ndata)
            $("#startrange").html(ndata);
          }//end success
        });//end ajax
      } //end

      // getting the end time from user configuration
      function endTimeRange(){
        jQuery.ajax({
          url: "/api/endTimeRange" ,
          type: 'GET',
          success: function(ndata, textStatus, xhr){
            console.log(ndata)
            $("#endrange").html(ndata);
          }//end success
        });//end ajax
      } //end

      $(document).ready(function(){
        setInterval(function () {
          motion_stat();
          light_val();
          startTimeRange();
          endTimeRange();
        }, 3000);
      });

      // for the turning on of alarm
      function turnonalarm(){
          $.ajax({url: "/api/alert/alarm",
                  type: 'POST',
                  success: function(){
                    $("#buzzerstatus").html("Rang alarm!");
                    // setTimeout( { $("#buzzerstatus").html("Rang alarm!"); }, 30000);
                              }
                })
        }

        $(document).ready(function(){
            $("#alarmb1").click(function(){
                  turnonalarm();
              });
          });


    </script>

</head>
<body class="hold-transition sidebar-mini layout-fixed">
<div class="wrapper">

  <!-- Navbar -->
  <nav class="main-header navbar navbar-expand navbar-white navbar-light">
    <!-- Left navbar links -->
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" data-widget="pushmenu" href="#"><i class="fas fa-bars"></i></a>
      </li>
      <li class="nav-item d-none d-sm-inline-block">
        <a href="index.html" class="nav-link">Home</a>
      </li>
      <!-- <li class="nav-item d-none d-sm-inline-block">
        <a href="charts.html" class="nav-link">Charts</a>
      </li> -->
    </ul>

    <!-- SEARCH FORM -->
    <form class="form-inline ml-3">
      <div class="input-group input-group-sm">
        <input class="form-control form-control-navbar" type="search" placeholder="Search" aria-
label="Search">
        <div class="input-group-append">
          <button class="btn btn-navbar" type="submit">
            <i class="fas fa-search"></i>
          </button>
        </div>
      </div>
    </form>
  </nav>
  <!-- /.navbar -->

  <!-- Main Sidebar Container -->
  <aside class="main-sidebar sidebar-dark-primary elevation-4">
    <!-- Brand Logo -->
    <a href="index3.html" class="brand-link">
      <img src="../static/dist/img/AdminLTELogo.png" alt="AdminLTE Logo" class="brand-image
img-circle elevation-3"
         style="opacity: .8">
      <span class="brand-text font-weight-light">MoniCam</span>
    </a>

    <!-- Sidebar -->
    <div class="sidebar">
      <!-- Sidebar user panel (optional) -->
      <div class="user-panel mt-3 pb-3 mb-3 d-flex">
        <div class="image">
          <img src="../static/dist/img/displaypic.png" class="img-circle elevation-2" alt="User
Image">
        </div>
        <div class="info">
          <a href="#" class="d-block">nerd</a>
        </div>
      </div>

      <!-- Sidebar Menu -->
      <nav class="mt-2">
        <ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview" role="menu" data-
    accordion="false">
          <li class="nav-item">
            <a href="./index.html" class="nav-link active">
              <i class="nav-icon fas fa-home"></i>
              <p>
                Home
              </p>
            </a>
          </li>
          <li class="nav-item">
            <a href="./charts.html" class="nav-link">
              <i class="nav-icon fas fa-chart-line"></i>
              <p>
                Charts
              </p>
            </a>
          </li>
          <li class="nav-item">
            <a href="forms.html" class="nav-link">
              <i class="nav-icon fas fa-th"></i>
              <p>
                Forms
              </p>
            </a>
          </li>
        </ul>
      </nav>
      <!-- /.sidebar-menu -->
    </div>
    <!-- /.sidebar -->
  </aside>

  <!-- Content Wrapper. Contains page content -->
  <div class="content-wrapper">
    <!-- Content Header (Page header) -->
    <div class="content-header">
      <div class="container-fluid">
        <div class="row mb-2">
          <div class="col-sm-6">
            <h1 class="m-0 text-dark">Home</h1>
          </div><!-- /.col -->
          <div class="col-sm-6">
            <ol class="breadcrumb float-sm-right">
              <li class="breadcrumb-item"><a href="#">Home</a></li>
            </ol>
          </div><!-- /.col -->
```

```
        </div><!-- /.row -->
      </div><!-- /.container-fluid -->
    </div>
    <!-- /.content-header -->

    <!-- Main content -->
    <section class="content">
      <div class="container-fluid">
        <!-- Small boxes (Stat box) -->
        <div class="row">
          <div class="col-lg-3 col-6">
            <!-- small box -->
            <div class="small-box bg-info">
              <div class="inner">
                <h3><span ></span></h3>

                <p>Current Light Intensity</p>
                <p id="light_value"></p>
              </div>
              <div class="icon">
                <i class="fas fa-sun"></i>
              </div>
              <!-- <a href="#" class="small-box-footer">More info <i class="fas fa-arrow-circle-
right"></i></a> -->
            </div>
          </div>
          <!-- ./col -->
          <div class="col-lg-3 col-6">
            <!-- small box -->
            <div class="small-box bg-success">
              <div class="inner">
                <h3><span ></span></h3>

                <p>Is Motion Detected?</p>
                <p id="motion_status"></p>
              </div>
              <div class="icon">
                <i class="fas fa-shoe-prints"></i>
              </div>
              <!-- <a href="#" class="small-box-footer">More info <i class="fas fa-arrow-circle-
right"></i></a> -->
            </div>
          </div>
          <!-- ./col -->
          <div class="col-lg-3 col-6">
            <!-- small box -->
            <div class="small-box bg-warning">
              <div class="inner">
                <h3 id="buzzerstatus">
                  <button class="button" id="alarmb1">ON</button>
                  <!--<button class="button" id="alarmb2">OFF</button>-->
                </h3>

                <p>Buzzer</p>
              </div>
              <div class="icon">
                <i class="far fa-bell"></i>
              </div>
              <!-- <a href="#" class="small-box-footer">More info <i class="fas fa-arrow-circle-
      right"></i></a> -->
            </div>
          </div>
        </div>
        <!-- /.row -->
        <!-- Main row -->
        <div class="row">
          <!-- Left col -->
          <section class="col-lg-7 connectedSortable">
            <!-- Custom tabs (Charts with tabs)-->
            <div class="card">
              <div class="card-header">
                <h3 class="card-title" style="z-index: 4">
                  <i class="fas fa-chart-pie mr-1"></i>
                  Light Intensity of My Room
                </h3>

              </div><!-- /.card-header -->
              <div id="chart_div" style="width:100%"></div>
                <div class="card-body">
                  <div class="tab-content p-0">
                </div>
              </div>
            </div>
            <!-- /.card -->

            <!-- Photos -->
            <div class="card">
              <div class="card-header">
                <h3 class="card-title">
                  <i class="far fa-images"></i>
                  Photos
                </h3>
              </div>
              <!-- /.card-header -->
              <div class="card-body">
                <ul class="todo-list" id="photo-list" data-widget="todo-list">

              </div>
              <!-- /.card-body -->
              <div class="card-footer clearfix">
                <button type="button" class="btn btn-info float-right"><i class="fas fa-camera"></i>
      Take Photo</button>
              </div>
            </div>
            <!-- /.card -->
          </section>
          <!-- /.Left col -->
          <!-- right col (We are only adding the ID to make the widgets sortable)-->
          <section class="col-lg-5 connectedSortable">

            <!-- Map card -->
            <div class="card bg-gradient-primary">
              <div class="card-header border-0">
                <h3 class="card-title">
                  <i class="fas fa-map-marker-alt mr-1"></i>
                  Time Configuration (24h Clock)
                </h3>

              </div>
              <div class="card-body">
                <form action="/api/getTimeConfig" method="post">
                  <input type="time" id="starttime" name="starttime">
                  To
                  <input type="time" id="endtime" name="endtime">
                  <input type="submit" class="formbutton" value="OK!">
                  <!-- <input type="submit"> -->
                </form>
              </div>
              <!-- /.card-body-->
              <div class="card-footer bg-transparent">
                <div class="row">
                  <div class="col-4 text-center">
                    <div id="sparkline-1" style= "display: none"></div>
                    <div class="text-white" style= "display: none">Visitors</div>
                  </div>
                  <!-- ./col -->
                  <div class="col-4 text-center">
                    <div id="sparkline-2"style= "display: none"></div>
                    <div class="text-white"style= "display: none">Online</div>
                  </div>
                  <!-- ./col -->
                  <div class="col-4 text-center">
                    <div id="sparkline-3"style= "display: none"></div>
                    <div class="text-white"style= "display: none">Sales</div>
                  </div>
                  <!-- ./col -->
                </div>
                <!-- /.row -->
              </div>
            </div>
            <!-- /.card -->
            <!-- Calendar -->
            <div class="card bg-gradient-info">
              <div class="card-header border-0">

                <h3 class="card-title">
                  <i class="far fa-calendar-alt"></i>
                  Calendar
                </h3>
                <!-- tools card -->
                <div class="card-tools">
                  <!-- button with a dropdown -->
                  <div class="btn-group">
                    <button type="button" class="btn btn-info btn-sm dropdown-toggle" data-
      toggle="dropdown">
                      <i class="fas fa-bars"></i></button>
                    <div class="dropdown-menu float-right" role="menu">
                      <a href="#" class="dropdown-item">Add new event</a>
                      <a href="#" class="dropdown-item">Clear events</a>
                      <div class="dropdown-divider"></div>
                      <a href="#" class="dropdown-item">View calendar</a>
                    </div>
                  </div>
                  <button type="button" class="btn btn-info btn-sm" data-card-widget="collapse">
                    <i class="fas fa-minus"></i>
                  </button>
                  <button type="button" class="btn btn-info btn-sm" data-card-widget="remove">
                    <i class="fas fa-times"></i>
                  </button>
                </div>
                <!-- /. tools -->
              </div>
              <!-- /.card-header -->
              <div class="card-body pt-0">
                <!--The calendar -->
                <div id="calendar" style="width: 100%"></div>
              </div>
              <!-- /.card-body -->
```

```
          </div>

             </div>
             <!-- ./col -->
            </div>
            <!-- /.row -->
           </div>
          </div>
          <!-- /.card -->

          <!-- /.card -->
        </section>
        <!-- right col -->
      </div>
      <!-- /.row (main row) -->
     </div><!-- /.container-fluid -->
    </section>
    <!-- /.content -->
  </div>
  <!-- /.content-wrapper -->
  <footer class="main-footer">
    <strong>Copyright &copy; 2014-2019 <a href="http://adminlte.io">AdminLTE.io</a>.</strong>
    All rights reserved.
    <div class="float-right d-none d-sm-inline-block">
      <b>Version</b> 3.0.1
    </div>
  </footer>

  <!-- Control Sidebar -->
  <aside class="control-sidebar control-sidebar-dark">
    <!-- Control sidebar content goes here -->
  </aside>
  <!-- /.control-sidebar -->
</div>
<!-- ./wrapper -->


<!-- jQuery UI 1.11.4 -->
<script src="../static/plugins/jquery-ui/jquery-ui.min.js"></script>
<!-- Resolve conflict in jQuery UI tooltip with Bootstrap tooltip -->
<script>
  $.widget.bridge('uibutton', $.ui.button)
</script>
<!-- Bootstrap 4 -->
<script src="../static/plugins/bootstrap/js/bootstrap.bundle.min.js"></script>
<!-- ChartJS -->
<script src="../static/plugins/chart.js/Chart.min.js"></script>
```

```
<!-- Sparkline -->
<script src="../static/plugins/sparklines/sparkline.js"></script>
<!-- JQVMap -->
<script src="../static/plugins/jqvmap/jquery.vmap.min.js"></script>
<script src="../static/plugins/jqvmap/maps/jquery.vmap.usa.js"></script>
<!-- jQuery Knob Chart -->
<script src="../static/plugins/jquery-knob/jquery.knob.min.js"></script>
<!-- daterangepicker -->
<script src="../static/plugins/moment/moment.min.js"></script>
<script src="../static/plugins/daterangepicker/daterangepicker.js"></script>
<!-- Tempusdominus Bootstrap 4 -->
<script src="../static/plugins/tempusdominus-bootstrap-4/js/tempusdominus-bootstrap-
4.min.js"></script>
<!-- Summernote -->
<script src="../static/plugins/summernote/summernote-bs4.min.js"></script>
<!-- overlayScrollbars -->
<script src="../static/plugins/overlayScrollbars/js/jquery.overlayScrollbars.min.js"></script>
<!-- AdminLTE App -->
<script src="../static/dist/js/adminlte.js"></script>
<!-- AdminLTE dashboard demo (This is only for demo purposes) -->
<script src="../static/dist/js/pages/dashboard.js"></script>
<!-- AdminLTE for demo purposes -->
<script src="../static/dist/js/demo.js"></script>

</body>
</html>
```

charts.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta http-equiv="x-ua-compatible" content="ie=edge">

  <title>AdminLTE 3 | Dashboard 3</title>

  <!-- Font Awesome Icons -->
  <link rel="stylesheet" href="../static/plugins/fontawesome-free/css/all.min.css">
  <!-- IonIcons -->
  <link rel="stylesheet"
href="http://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
  <!-- Theme style -->
  <link rel="stylesheet" href="../static/dist/css/adminlte.min.css">
  <!-- Google Font: Source Sans Pro -->
  <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700"
rel="stylesheet">
  <!-- jQuery -->
  <script src="../static/plugins/jquery/jquery.min.js"></script>

</head>
<!--
BODY TAG OPTIONS:
=================
Apply one or more of the following classes to to the body tag
to get the desired effect
|---------------------------------------------------------|
|LAYOUT OPTIONS | sidebar-collapse                |
|               | sidebar-mini                    |
|---------------------------------------------------------|
-->
<body class="hold-transition sidebar-mini">
<div class="wrapper">
  <!-- Navbar -->
  <nav class="main-header navbar navbar-expand navbar-white navbar-light">
    <!-- Left navbar links -->
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" data-widget="pushmenu" href="#"><i class="fas fa-bars"></i></a>
      </li>
      <li class="nav-item d-none d-sm-inline-block">
        <a href="index.html" class="nav-link">Home</a>
    </li>
   </ul>

   <!-- SEARCH FORM -->
   <form class="form-inline ml-3">
    <div class="input-group input-group-sm">
     <input class="form-control form-control-navbar" type="search" placeholder="Search" aria-
label="Search">
     <div class="input-group-append">
      <button class="btn btn-navbar" type="submit">
       <i class="fas fa-search"></i>
      </button>
     </div>
    </div>
   </form>
  </nav>
  <!-- /.navbar -->

  <!-- Main Sidebar Container -->
  <aside class="main-sidebar sidebar-dark-primary elevation-4">
   <!-- Brand Logo -->
   <a href="index3.html" class="brand-link">
    <img src="../static/dist/img/AdminLTELogo.png" alt="AdminLTE Logo" class="brand-image
img-circle elevation-3"
        style="opacity: .8">
    <span class="brand-text font-weight-light">MoniCam</span>
   </a>

   <!-- Sidebar -->
   <div class="sidebar">
    <!-- Sidebar user panel (optional) -->
    <div class="user-panel mt-3 pb-3 mb-3 d-flex">
     <div class="image">
      <img src="../static/dist/img/displaypic.png" class="img-circle elevation-2" alt="User
Image">
     </div>
     <div class="info">
      <a href="#" class="d-block">nerd</a>
     </div>
    </div>

    <!-- Sidebar Menu -->
    <nav class="mt-2">
     <ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview" role="menu" data-
accordion="false">
      <li class="nav-item">
       <a href="./index.html" class="nav-link">
```

```
          <i class="nav-icon fas fa-home"></i>
          <p>
            Home
          </p>
        </a>
      </li>
      <li class="nav-item">
        <a href="./charts.html" class="nav-link active">
          <i class="nav-icon fas fa-chart-line"></i>
          <p>
            Charts
          </p>
        </a>
      </li>
      <li class="nav-item">
        <a href="forms.html" class="nav-link">
          <i class="nav-icon fas fa-th"></i>
          <p>
            Forms
          </p>
        </a>
      </li>
    </ul>
  </nav>
  <!-- /.sidebar-menu -->
 </div>
 <!-- /.sidebar -->
</aside>

<!-- Content Wrapper. Contains page content -->
<div class="content-wrapper">
 <!-- Content Header (Page header) -->
 <div class="content-header">
  <div class="container-fluid">
   <div class="row mb-2">
    <div class="col-sm-6">
     <h1 class="m-0 text-dark">Charts</h1>
    </div><!-- /.col -->
    <div class="col-sm-6">
     <ol class="breadcrumb float-sm-right">
      <li class="breadcrumb-item"><a href="#">Charts</a></li>
     </ol>
    </div><!-- /.col -->
   </div><!-- /.row -->
  </div><!-- /.container-fluid -->
 </div>
 <!-- /.content-header -->
```

```
<!-- Main content -->
<div class="content">
  <div class="container-fluid">
    <div class="row">
      <div class="col-lg-6">
        <div class="card">
          <div class="card-header border-0">
            <div class="d-flex justify-content-between">
              <h3 class="card-title">Number of motion detected on date basis</h3>
            </div>
          </div>
          <div class="card-body">
            <!-- INSERT GRAPH DETAILS HERE !!!! -->
            <div id="motion_chart_div">

            </div>
          </div>
        </div>
        <!-- /.card -->

        <div class="card">
          <div class="card-header border-0">
            <h3 class="card-title">All time motion detected table</h3>
          </div>
          <div class="card-body">
            <!-- INSERT TABLE DETAILS HERE !!!! -->
            <div id="motion_table_div" style="height: 500px;">

            </div>
          </div>
          <div class="card-body table-responsive p-0">
            <table class="table table-striped table-valign-middle">
            </table>
          </div>
        </div>
        <!-- /.card -->
      </div>
      <!-- /.col-md-6 -->
      <div class="col-lg-6">
        <div class="card">
          <div class="card-header border-0">
            <div class="d-flex justify-content-between">
              <h3 class="card-title">Number of attempts on date basis</h3>
            </div>
          </div>
          <div class="card-body">
```

```
            <!-- INSERT GRAPH DETAILS HERE !!!! -->
            <div id="attempt_chart_div">

            </div>
          </div>
        </div>

        <!-- /.card -->

        <div class="card">
          <div class="card-header border-0">
            <h3 class="card-title">All time attempts table</h3>
          </div>
          <div class="card-body">
            <!-- INSERT TABLE DETAILS HERE !!!! -->
            <div id="attempt_table_div" style="height: 500px;">

            </div>
          </div>
        </div>
      </div>
      <!-- /.col-md-6 -->
    </div>
    <!-- /.row -->
  </div>
  <!-- /.container-fluid -->
 </div>
 <!-- /.content -->
</div>
<!-- /.content-wrapper -->

<!-- Control Sidebar -->
<aside class="control-sidebar control-sidebar-dark">
  <!-- Control sidebar content goes here -->
</aside>
<!-- /.control-sidebar -->

<!-- Main Footer -->
<footer class="main-footer">
  <strong>Copyright &copy; 2014-2019 <a href="http://adminlte.io">AdminLTE.io</a>.</strong>
  All rights reserved.
  <div class="float-right d-none d-sm-inline-block">
    <b>Version</b> 3.0.1
  </div>
</footer>
</div>
<!-- ./wrapper -->
```

```
<!-- REQUIRED SCRIPTS -->

<!-- jQuery -->
<script src="../static/plugins/jquery/jquery.min.js"></script>
<!-- Bootstrap -->
<script src="../static/plugins/bootstrap/js/bootstrap.bundle.min.js"></script>
<!-- AdminLTE -->
<script src="../static/dist/js/adminlte.js"></script>

<!-- OPTIONAL SCRIPTS -->
<script src="../static/plugins/chart.js/Chart.min.js"></script>
<script src="../static/dist/js/demo.js"></script>
<script src="../static/dist/js/pages/dashboard3.js"></script>

<!-- javascripts -->
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.24.0/moment.min.js"></script>
<script type="text/javascript">
  // Load Charts, Table and the corechart package.
 google.charts.load('current', {'packages':['corechart','table', 'bar']});
 google.charts.setOnLoadCallback(drawMotionGraph);
 google.charts.setOnLoadCallback(drawMotionTable);
 google.charts.setOnLoadCallback(drawAttemptGraph);
 google.charts.setOnLoadCallback(drawAttemptTable);
 // google.charts.setOnLoadCallback(drawAttemptBar);

 function groupBy(elements, duration) {
  const formatted = elements.map(elem => {
    return { date: moment(elem.date).startOf(duration).format('YYYY-MM-DD'), count: elem.count
}
  })

  const dates = formatted.map(elem => elem.date)
  const uniqueDates = dates.filter((date, index) => dates.indexOf(date) === index)

  return uniqueDates.map(date => {
    const count = formatted.filter(elem => elem.date === date).reduce((count, elem) => count +
elem.count, 0)
    return { date, count }
  })
}

 //==================================================================//
 // start of all time motion graph
 // function to retrieve data for all time motion graph
 function drawMotionGraph(){
     jQuery.ajax({
         url: "/api/get_alltime_motion_graph_data" ,
         type: 'GET',
         success: function(ndata, textStatus, xhr){
             console.log(ndata)
             var chartdata = ndata.chart_data
             drawMotionLineChart(chartdata)
           }//end success
     });//end ajax
 }

 //draw the all time motion line chart
 function drawMotionLineChart(graphdata){
    var chart = new google.visualization.LineChart(
    document.getElementById('motion_chart_div'));
    var motion_chart = new google.visualization.DataTable();
    motion_chart.addColumn('string', 'Motion Date');
    motion_chart.addColumn('number', 'Number of Motion Sensed');
    var graphdata = JSON.parse(graphdata);
    console.log('graphdata')
    console.log(graphdata)

    for (i=0;i<graphdata.length;i++){
       date = graphdata[i].date;
       number_of_motion = graphdata[i].motion;
       motion_chart.addRows([[date,number_of_motion]]);
    }//end for
    chart.draw(motion_chart, {legend: 'none', vAxis: {baseline: 0},
       colors: ['#A0D100']});
 }
 // end of all time motion graph
 //==================================================================//

 //==================================================================//
 // start of all time motion table

 // function to retrieve data for all time motion table
 function drawMotionTable(){
     jQuery.ajax({
         url: "/api/get_alltime_motion_table_data" ,
         type: 'GET',
         success: function(ndata, textStatus, xhr){
             console.log(ndata)
             var chartdata = ndata.chart_data
             createMotionDataTable(chartdata)
```

```
         }//end success
     });//end ajax
 }

 // draw the all time motion table
 function createMotionDataTable(newdata){
   var motion_table = new google.visualization.DataTable();
   motion_table.addColumn('number', 'Light Value');
   motion_table.addColumn('string', 'Motion Datetime');
   motion_table.addColumn('string', 'Motion Image');
   var newdata = JSON.parse(newdata);

   for (i=0;i<newdata.length;i++){
      lightvalue = parseInt(newdata[i].lightvalue);
      motion_datetime = newdata[i].motion_datetime;
      motion_image = newdata[i].motion_image;
      motion_table.addRows([[lightvalue,motion_datetime,motion_image]]);
   }//end for
   //draw the motion table
   var table = new google.visualization.Table(document.getElementById('motion_table_div'));
   table.draw(motion_table, {showRowNumber: true, width: '100%', height: '100%'});
 }
 // end of all time motion table
 //=================================================================//

//###########################################################################

//=================================================================//
// start of all time attempt graph

// function to retrieve data for all time attempt graph
function drawAttemptGraph(){
    jQuery.ajax({
        url: "/api/get_alltime_attempt_graph_data" ,
        type: 'GET',
        success: function(ndata, textStatus, xhr){
            console.log(ndata);
            var chartdata = ndata.chart_data;
            drawAttemptLineChart(chartdata);
          }//end success
    });//end ajax
}

//draw the attmept line chart
function drawAttemptLineChart(newdata){

   var chart = new google.visualization.LineChart(
   document.getElementById('attempt_chart_div'));
   var attempt_chart = new google.visualization.DataTable();
   attempt_chart.addColumn('string', 'Attempt Datetime');
   attempt_chart.addColumn('number', 'Number of attempts');

   var newdata = JSON.parse(newdata);

   for (i=0;i<newdata.length;i++){
      attempt_datetime = newdata[i].date;
      number_of_attempts = newdata[i].attempt;
      attempt_chart.addRows([[attempt_datetime, number_of_attempts]]);
   }//end for

   chart.draw(attempt_chart, {legend: 'none', vAxis: {baseline: 0},
      colors: ['#A0D100']});
}

// end of all time attempt graph
//====================================================================//

//====================================================================//
//start of all time attempt table

// function to retrieve data for all time attmept table
function drawAttemptTable(){
    jQuery.ajax({
        url: "/api/get_alltime_attempt_table_data" ,
        type: 'GET',
        success: function(ndata, textStatus, xhr){
            console.log(ndata);
            $("#status").html("Data fetched! Now plotting graph!");
            var chartdata = ndata.chart_data;
            createAttemptDataTable(chartdata);
          }//end success
    });//end ajax
}

function createAttemptDataTable(newdata){
 var attempt_table = new google.visualization.DataTable();
 attempt_table.addColumn('string', 'Attempt ID');
 attempt_table.addColumn('string', 'Date and time of attempt');
 attempt_table.addColumn('string', 'RFID ID');
 attempt_table.addColumn('string', 'Username (if any)');
 attempt_table.addColumn('string', 'Attempt success');
 attempt_table.addColumn('string', 'Attempt Image');

 var newdata = JSON.parse(newdata);
```

```
  for (i=0;i<newdata.length;i++){
    id = newdata[i].id;
    datetime = newdata[i].attempt_datetime;
    rfidId = newdata[i].rfid_id;
    uname = newdata[i].username;
    yn_success = newdata[i].success;
    attempt_img = newdata[i].attempt_img_path;
    attempt_table.addRows([[id, datetime, rfidId, uname, yn_success, attempt_img]]);
  }//end for
  //draw the motion table
  var table = new google.visualization.Table(document.getElementById('attempt_table_div'));
  table.draw(attempt_table, {showRowNumber: true, width: '100%', height: '100%'});
}
</script>

</body>
</html>
```

forms.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>AdminLTE 3 | General Form Elements</title>
  <!-- Tell the browser to be responsive to screen width -->
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- Font Awesome -->
  <link rel="stylesheet" href="../static/plugins/fontawesome-free/css/all.min.css">
  <!-- Ionicons -->
  <link rel="stylesheet"
href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
  <!-- Theme style -->
  <link rel="stylesheet" href="../static/dist/css/adminlte.min.css">
  <!-- Google Font: Source Sans Pro -->
  <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700"
rel="stylesheet">
  <!-- jQuery -->
  <script src="../static/plugins/jquery/jquery.min.js"></script>
</head>
<body class="hold-transition sidebar-mini">
<div class="wrapper">
  <!-- Navbar -->
  <nav class="main-header navbar navbar-expand navbar-white navbar-light">
    <!-- Left navbar links -->
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" data-widget="pushmenu" href="#"><i class="fas fa-bars"></i></a>
      </li>
      <li class="nav-item d-none d-sm-inline-block">
        <a href="index.html" class="nav-link">Home</a>
      </li>
    </ul>

    <!-- SEARCH FORM -->
    <form class="form-inline ml-3">
      <div class="input-group input-group-sm">
        <input class="form-control form-control-navbar" type="search" placeholder="Search" aria-
label="Search">
        <div class="input-group-append">
          <button class="btn btn-navbar" type="submit">
            <i class="fas fa-search"></i>
          </button>
        </div>
      </div>
    </form>
  </nav>
  <!-- /.navbar -->

  <!-- Main Sidebar Container -->
  <aside class="main-sidebar sidebar-dark-primary elevation-4">
    <!-- Brand Logo -->
    <a href="../../index3.html" class="brand-link">
      <img src="../static/dist/img/AdminLTELogo.png"
           alt="AdminLTE Logo"
           class="brand-image img-circle elevation-3"
           style="opacity: .8">
      <span class="brand-text font-weight-light">MoniCam</span>
    </a>

    <!-- Sidebar -->
    <div class="sidebar">
      <!-- Sidebar user (optional) -->
      <div class="user-panel mt-3 pb-3 mb-3 d-flex">
        <div class="image">
          <img src="../static/dist/img/displaypic.png" class="img-circle elevation-2" alt="User
Image">
        </div>
        <div class="info">
          <a href="#" class="d-block">nerd</a>
        </div>
      </div>

      <!-- Sidebar Menu -->
      <nav class="mt-2">
        <ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview" role="menu" data-
accordion="false">
          <li class="nav-item">
            <a href="./index.html" class="nav-link">
              <i class="nav-icon fas fa-home"></i>
              <p>
                Home
              </p>
            </a>
          </li>
          <li class="nav-item">
            <a href="./charts.html" class="nav-link">
              <i class="nav-icon fas fa-chart-line"></i>
              <p>
                Charts
              </p>
            </a>
          </li>
          <li class="nav-item">
            <a href="forms.html" class="nav-link active">
              <i class="nav-icon fas fa-th"></i>
              <p>
                Forms
              </p>
            </a>
          </li>

        </ul>
      </nav>
      <!-- /.sidebar-menu -->
    </div>
    <!-- /.sidebar -->
  </aside>

  <!-- Content Wrapper. Contains page content -->
  <div class="content-wrapper">
    <!-- Content Header (Page header) -->
    <section class="content-header">
      <div class="container-fluid">
        <div class="row mb-2">
          <div class="col-sm-6">
            <h1>Forms</h1>
          </div>
          <div class="col-sm-6">
            <ol class="breadcrumb float-sm-right">
              <li class="breadcrumb-item"><a href="#">Forms</a></li>
            </ol>
          </div>
        </div>
      </div><!-- /.container-fluid -->
    </section>

    <!-- Main content -->
    <section class="content">
      <div class="container-fluid">

        <div class="row">
          <div class="col-12">
            <div class="card">
              <div class="card-header">
                <h3 class="card-title">User Table</h3>
              </div>
              <!-- /.card-header -->
              <div class="card-body">
                <!-- INSERT DATA TABLE INFO HERE -->
                <div id="user_table_div">

                </div>
              </div>
            </div>
          </div>
        </div>

        <div class="row">
          <!-- left column -->
          <div class="col-md-6">
            <!-- start of add user form -->
            <!-- add user form elements -->
            <div class="card card-success">
              <div class="card-header">
                <h3 class="card-title">Add User</h3>
              </div>
              <!-- /.card-header -->
              <!-- form start -->
              <!-- <form role="form"> -->
                <form role="form" action="/form_add_user" class="form-horizontal" method="post">
                <div class="card-body">
                  <div class="form-group">
                    <label for="exampleInputEmail1">Rfid ID</label>
                    <input type="text" class="form-control" id="add_Rfid_ID" name="add_Rfid_ID"
    placeholder="Rfid ID">
                  </div>
                  <div class="form-group">
                    <label for="exampleInputPassword1">Username</label>
                    <input type="text" class="form-control" id="add_username" name="add_username"
    placeholder="username">
                  </div>
                </div>
                <!-- /.card-body -->

                <div class="card-footer">
                  <button type="submit" class="btn btn-success">Submit</button>
                </div>
              </form>
            </div>
            <!-- /.card -->
            <!-- end of add user form -->
```

```
          <!-- start of delete user form -->
          <!-- Form Element sizes -->
          <div class="card card-danger">
            <div class="card-header">
              <h3 class="card-title">Delete User</h3>
            </div>
            <form role="form" action="/form_delete_user" class="form-horizontal" method="post">
            <div class="card-body">
              <div class="form-group">
                <label for="exampleInputEmail1">ID</label>
                <div class="form-group">
                  <select name="del_id" id="del_id" class="form-control">
                    {% for id in userid %}
                    <option value="{{ id }}">{{ id }}</option>
                    {% endfor %}
                  </select>
                </div>
              </div>
            </div>
            <!-- /.card-body -->
              <div class="card-footer">
                <button type="submit" class="btn btn-danger">Submit</button>
              </div>
            </form>
          </div>
          <!-- /.card -->



        </div>
        <!--/.col (left) -->
        <!-- right column -->
        <div class="col-md-6">
          <!-- general form elements disabled -->
          <div class="card card-warning">
            <div class="card-header">
              <h3 class="card-title">Modify User</h3>
            </div>
            <!-- /.card-header -->
            <div class="card-body">
              <form role="form" action="/form_modify_user" class="form-horizontal"
method="post">
                <div class="form-group">
                  <label for="exampleInputEmail1">ID</label>
                  <div class="form-group">
                    <select name="mod_id" id="mod_id" class="form-control">
                      {% for id in userid %}
                      <option value="{{ id }}">{{ id }}</option>
                      {% endfor %}
                    </select>
                  </div>
                </div>
                <div class="form-group">
                  <label for="exampleInputEmail1">Rfid ID</label>
                  <input type="text" class="form-control" id="mod_Rfid_ID" name="mod_Rfid_ID"
placeholder="Rfid ID">
                </div>
                <div class="form-group">
                  <label for="exampleInputPassword1">Username</label>
                  <input type="text" class="form-control" id="chg_username" name="chg_username"
placeholder="username">
                </div>
                <!-- /.card-body -->
                <div class="card-footer">
                  <button type="submit" class="btn btn-warning">Submit</button>
                </div>
              </form>
            </div>
          </div>
          <!-- /.card -->
        </div>
        <!--/.col (right) -->
       </div>
       <!-- /.row -->
      </div><!-- /.container-fluid -->
     </section>
     <!-- /.content -->
</div>
<!-- /.content-wrapper -->
<footer class="main-footer">
     <div class="float-right d-none d-sm-block">
      <b>Version</b> 3.0.1
     </div>
     <strong>Copyright &copy; 2014-2019 <a href="http://adminlte.io">AdminLTE.io</a>.</strong>
All rights
     reserved.
</footer>

<!-- Control Sidebar -->
<aside class="control-sidebar control-sidebar-dark">
     <!-- Control sidebar content goes here -->
</aside>
<!-- /.control-sidebar -->
</div>
```

```
<!-- ./wrapper -->

<!-- jQuery -->
<script src="../static/plugins/jquery/jquery.min.js"></script>
<!-- Bootstrap 4 -->
<script src="../static/plugins/bootstrap/js/bootstrap.bundle.min.js"></script>
<!-- bs-custom-file-input -->
<script src="../static/plugins/bs-custom-file-input/bs-custom-file-input.min.js"></script>
<!-- AdminLTE App -->
<script src="../static/dist/js/adminlte.min.js"></script>
<!-- AdminLTE for demo purposes -->
<script src="../static/dist/js/demo.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
  // Load Charts, Table and the corechart package.
 google.charts.load('current', {'packages':['table']});
 // Draw the pie chart for Sarah's pizza when Charts is loaded.
 google.charts.setOnLoadCallback(drawUserTable);
var user_tabledata;

function drawUserTable(){
    $("#status").html("Fetching data to plot graph...");
     jQuery.ajax({
         url: "/api/get_user_data" ,
         type: 'POST',
         success: function(ndata, textStatus, xhr){
             console.log(ndata.chart_data);
             var tabledata = ndata.chart_data;
             createUserDataTable(tabledata);
           }//end success
     });//end ajax
}

function createUserDataTable(newdata){
   user_tabledata = new google.visualization.DataTable();
   user_tabledata.addColumn('string', 'ID');
   user_tabledata.addColumn('string', 'Username');
   user_tabledata.addColumn('string', 'RFID ID');
   user_tabledata.addColumn('string', 'Modification date and time');

   var newdata = JSON.parse(newdata);

   for (i=0;i<newdata.length;i++){
     id = newdata[i].id;
     username = newdata[i].username;
     rfid_id = newdata[i].rfid_id;
    modification_datetime = newdata[i].modification_date;

    user_tabledata.addRows([[id, username, rfid_id, modification_datetime]]);
  }//end for
  //draw the motion table
  var table = new google.visualization.Table(document.getElementById('user_table_div'));
      table.draw(user_tabledata, {showRowNumber: true, width: '100%', height: '100%'});
}
</script>
<script type="text/javascript">
$(document).ready(function () {
  bsCustomFileInput.init();
});
</script>
</body>
</html>
```

takeuserpic.py

```
from picamera import PiCamera
from utils import camera
import boto3
import time
import os
from datetime import datetime
import sys
from time import sleep

if len(sys.argv) == 1:
        print("Please supply username.\n" +
        "Username has to be the same as the username in user_info.")
        sys.exit()
elif len(sys.argv) == 2:
        if sys.argv[1] == '-h' or sys.argv[1] == '--help':
                print("Supply a username that is the same as the username in user_info.")
                sys.exit()
else:
        print("Too many arguments! Supply only the username!")
        sys.exit()

# To store images to s3
def s3_imgstore():
        print("Taking image in 3 seconds!")
        for i in range(3, 0, -1):
                print(i)
                sleep(1)
        filepath = camera(str(sys.argv[1]))
        print("Picture taken.")
        # Create an S3 resource
        s3 = boto3.resource('s3', region_name='us-east-1')

        # Set the bucket name
        img_bucket = '' # <replace> with your own unique bucket name
        dir_path = "/home/pi/motion_captures/" # <replace> with your own image directory
        exists = True

        try:
          s3.meta.client.head_bucket(Bucket=img_bucket)
        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=img_bucket,CreateBucketConfiguration={
              'LocationConstraint': 'us-east-1'})

        # Upload a new file
        s3.Object(img_bucket, filepath).put(Body=open(dir_path + filepath, 'rb'))
        print("File uploaded")

if __name__ == '__main__':
        s3_imgstore()
```

Section 5
Task List

A table listing members names and the parts of the assignment they worked on

Name of member	Part of project worked on	Contribution percentage
Poh Haw Jin	• Facial Recognition • Documented connection to EC2 instance • Migrated to DynamoDB • Implemented S3 image storage • MQTT communication between hardware program and server • Telegram bot	50%
Low Shu Ting	• Arduino for motion sensor, light intensity, and Piranha LED • Implemented EC2 • Did majority of the documentation • Web Interface • Producing and editing video	50%

Section 6
References

Arvind Sanjeev, 2018. How to Connect and Interface a Raspberry Pi With an Arduino [online]. Available from: https://maker.pro/raspberry-pi/tutorial/how-to-connect-and-interface-raspberry-pi-with-arduino [Accessed 13 January 2020]

Edpresso, 2020. *How to generate a random string in Python* [online]. Available from: https://www.educative.io/edpresso/how-to-generate-a-random-string-in-python [Accessed 21 January 2020]

Hackernoon, 2018. Launching an EC2 instance [online]. Available from: https://hackernoon.com/launching-an-ec2-instance-fbfd50894aac [Accessed on 17 January 2020]

Joaquin, G., 2020. *Deploy a Flask app on AWS EC2* [online]. Available from: https://www.codementor.io/@jqn/deploy-a-flask-app-on-aws-ec2-13hp1ilqy2 [Accessed 17 January 2020]

Python, 2020. *multiprocessing — Process-based “threading” interface* [online]. Available from: https://docs.python.org/2/library/multiprocessing.html [Accessed 15 January 2020]

Stackoverflow, 2020. *Convert every dictionary value to utf-8 (dictionary comprehension?)* [online]. Available from: https://stackoverflow.com/a/55734992 [Accessed 24 January 2020]

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