Internet Of Things CA2

Step-By-Step Tutorial

Date of Submission:	20 August 2018 (8:00am)
Prepared For:	Ms Dora Chua Heok Hoon
Class:	DBIT/FT/3B/32

Submitted By:

Student ID	Name
P1502558	Lau Ze Zhun Aaron
P1612806	Nurin Qistina Bte Mohd Ali
P1613092	Yeo Boon Hao

Table of Contents

Section 1 : Overview of the Application

A. What is the application about?

This IOT system is a Home Entertainment and Security system.

1. Security
   a. Tap RFID Card and input are saved into Firebase.
   b. If authorized, you can enter peacefully and the picture is taken and uploaded to S3
   c. If unauthorized, defence section comes in and an LCD Screen will say you're not authorized.
2. Defence
   a. Press button on dashboard.
   b. Laser Turrets will attack in random burst and speed.
3. Entertainment
   a. If motion is detected, game will start.
   b. After user plays game, score is saved into Firebase.
   c. LDR values will be taken and output on the dashboard.

This application is controllable and viewable via the IBM Node Red web server. We make use of AWS and IBM Cloud Services and we used Firebase as our Database.

A tutorial of this is uploading onto Instructables and you can view it at this link:

https://www.instructables.com/id/Overview-Home-Entertainment-and-Security-System/

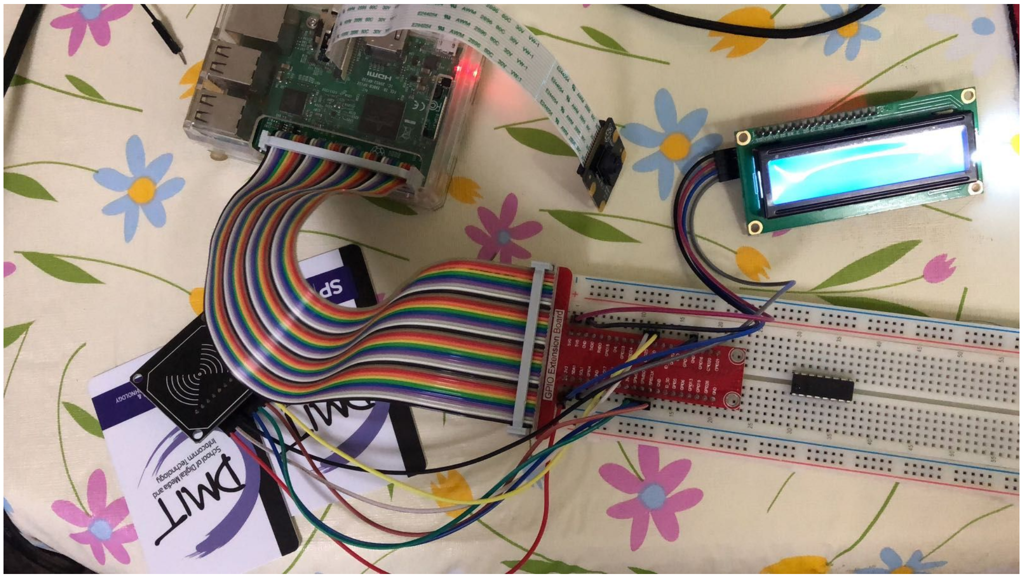
B. Summary of the steps that will be described

#	Section	Description
1	**Overview of the Application**	**A summary of what this tutorial is about.**
2	**Hardware Requirements**	**What hardware you need to create the security, defence and entertainment section.**
3	**Security**	**How to create security system**
	Hardware	How to connect the parts together
	Software	security.py is a code that will read rfid inputs and detect whether user is an intruder or not. If user is recognised, an image will be taken and uploaded to s3. The code also publishes to a topic in aws MQTT
4	**Defence**	**How to create a laser turret**
	Hardware	How to connect the parts together and most importantly, how to create the turret itself.
	Software	Laserturret.py is a code that triggers the laser turret. It shoots laser beams in random directions in random bursts and speed.
5	**Entertainment**	**How to create a simon-says game**
	Hardware	How to connect the parts together.
	Software	Entertain.py is the game code where you have to follow the pattern of the LEDS lighting up and press the corresponding buttons. It uploads scores and timestamp

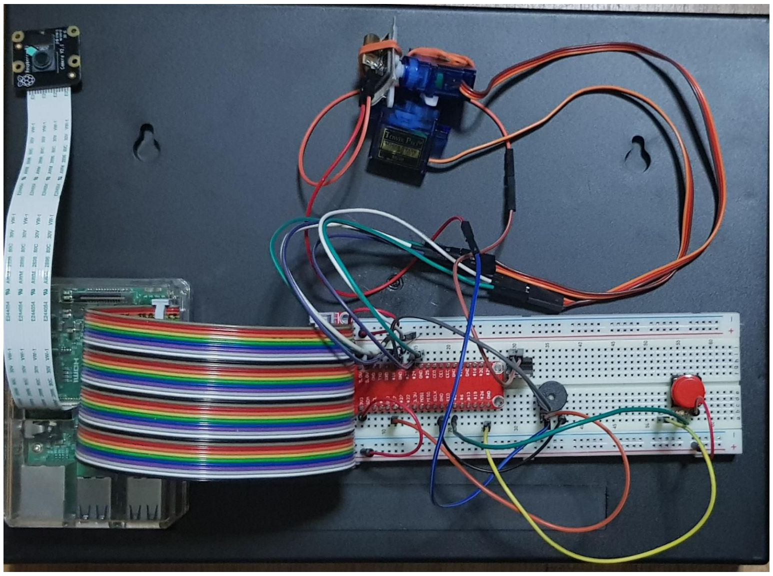
		into the firebase nosql database for further usage in the dashboards.
6	**IOT App Watson on IBM Bluemix**	**Creating a dashboard and interacting with multiple RPi from one dashboard.**
	Set up Bluemix IoT Service	Connecting different Pi to the same account
	Set up Node-Red	Downloading palettes to use in our node-red flow
	Security	
	Defence	Creating the node-red flow on RPi and on Bluemix.
	Entertainment	

C. How does the final RPI setup look like?

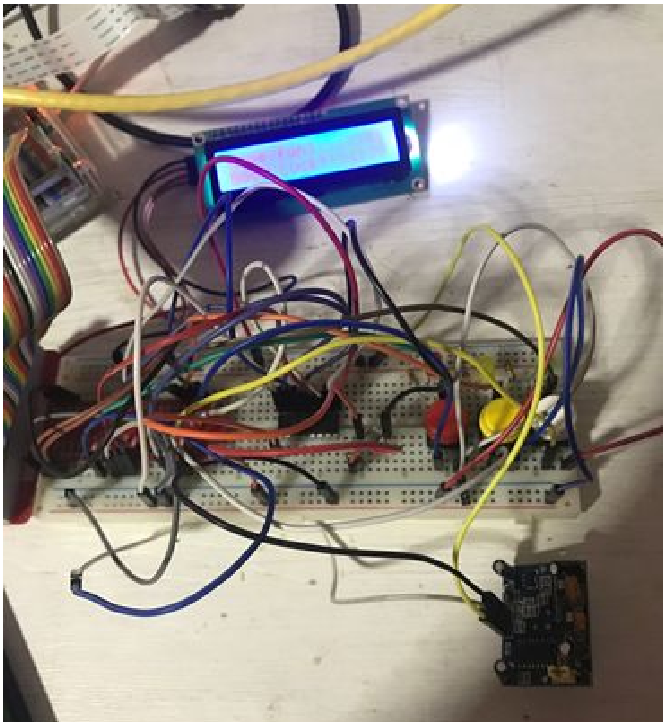
a. Security



b. Defence



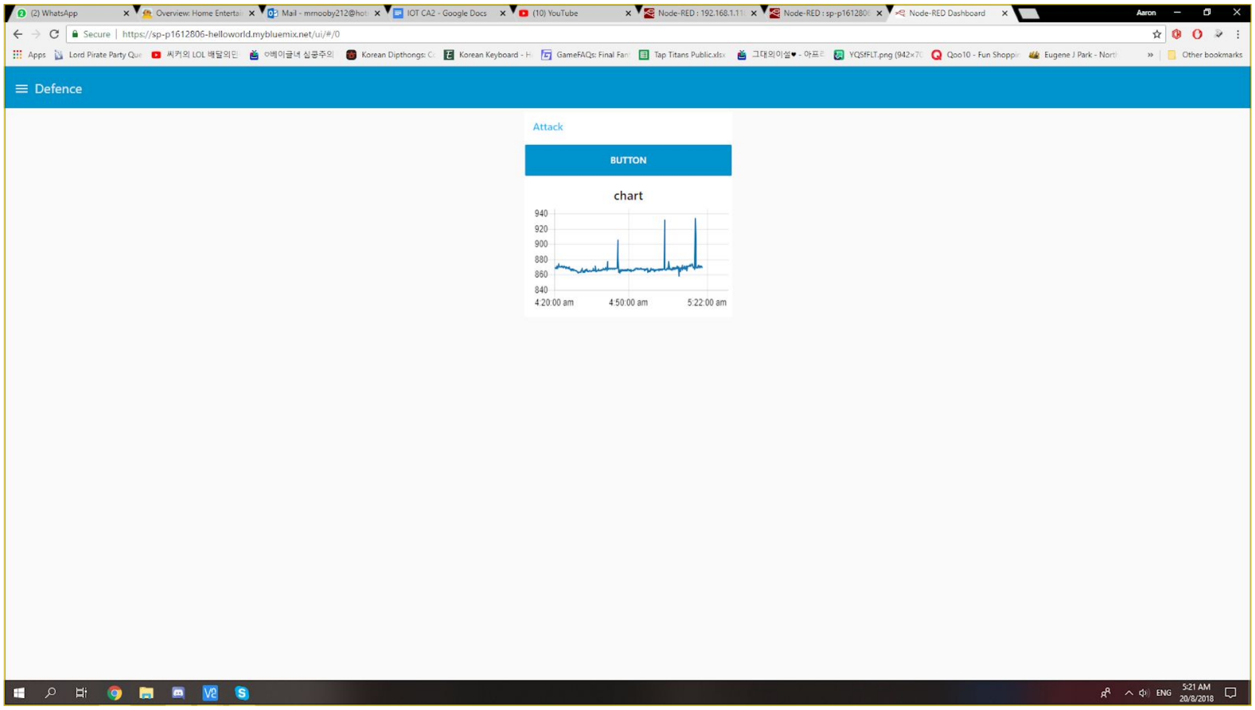
c. Entertainment



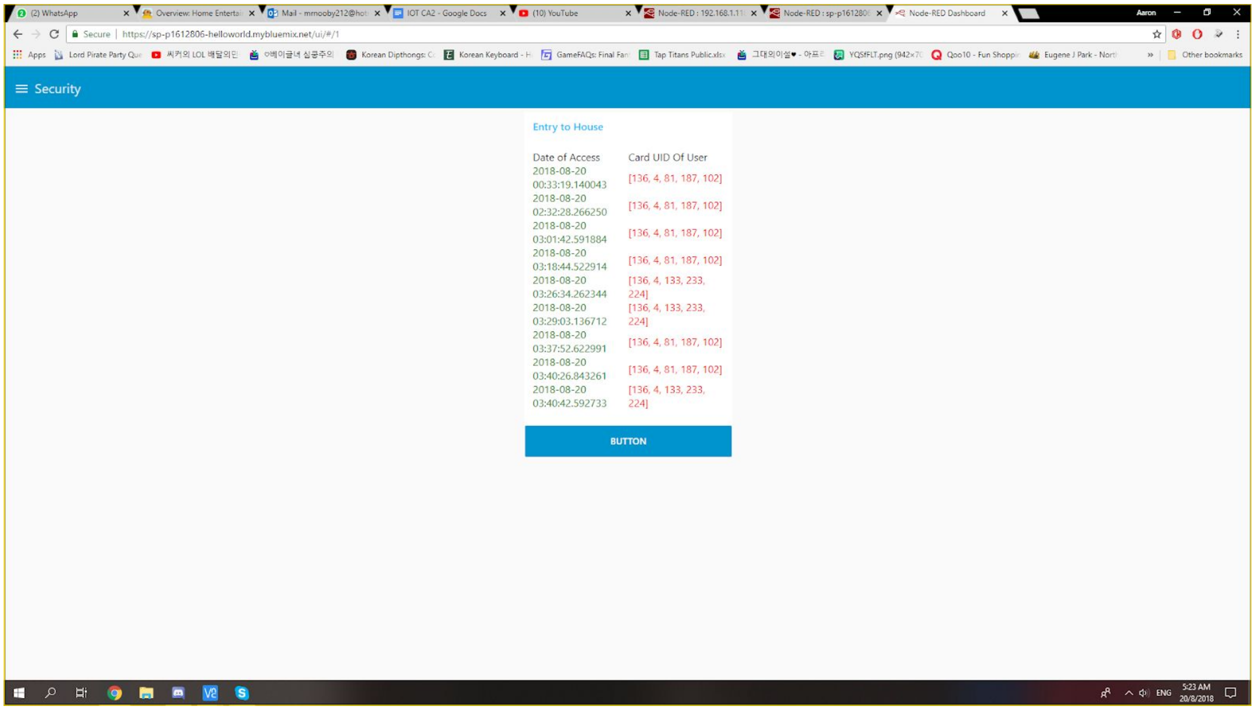
D. How does the web application look like?

As the we have created our dashboard for mobility usage on mobile phones, we decided to use a mobile responsive dashboard layout which will look better on the mobile phone. However, we still are going to show screenshots on it being opened on a web browser on a computer.

Defend + Light page



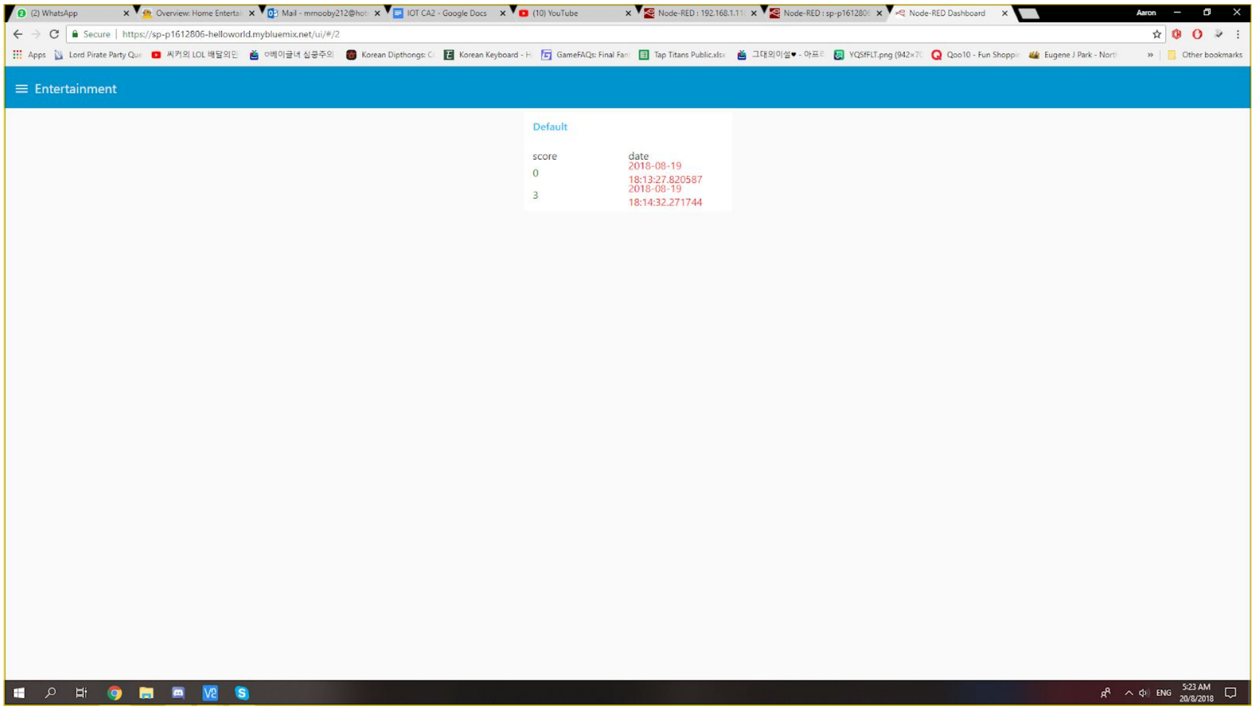
Access page



Entry to House

Date of Access	Card UID Of User
2018-08-20 00:33:19.140043	[136, 4, 81, 187, 102]
2018-08-20 02:32:28.266250	[136, 4, 81, 187, 102]
2018-08-20 03:01:42.591884	[136, 4, 81, 187, 102]
2018-08-20 03:18:44.522914	[136, 4, 81, 187, 102]
2018-08-20 03:26:34.262344	[136, 4, 133, 233, 224]
2018-08-20 03:29:03.136712	[136, 4, 133, 233, 224]
2018-08-20 03:37:52.622991	[136, 4, 81, 187, 102]
2018-08-20 03:40:26.843261	[136, 4, 81, 187, 102]
2018-08-20 03:40:42.592733	[136, 4, 133, 233, 224]

Scoreboard page



Default

score	date
0	2018-08-19 18:13:27.820587
3	2018-08-19 18:14:32.271744

Section 2 : Hardware Requirements

A. Hardware Checklist

a. Security

- 1 Raspberry Pi
- 1 LCD
- 1 RFID Reader
- 1 PiCam
- 2 RFID Cards/Buttons
- X Female -> Male jumper cables

b. Defence

You will need

- 1 Raspberry Pi

2	10 kΩ Resistor (for Buttons)
2	Micro Servo
1	650nm Laser Transmitter Module
2	Push Button
1	Buzzer
3	Small Rubber Bands/Cable Ties (for fixing)
X	Female -> Male jumper cables
X	Regular Jump Cables
1	Transistor
1	Capacitor

c. Entertainment

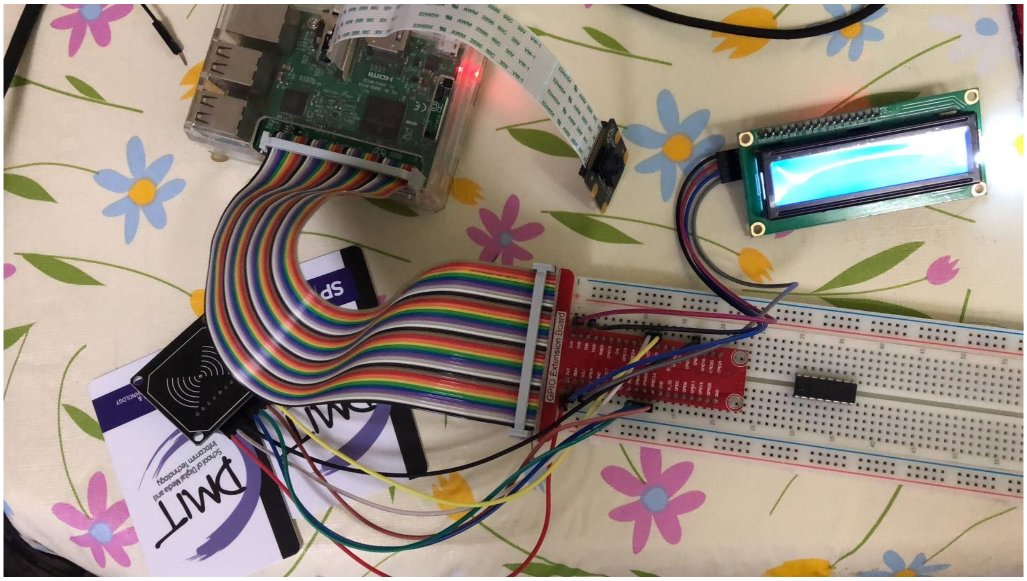
You will need

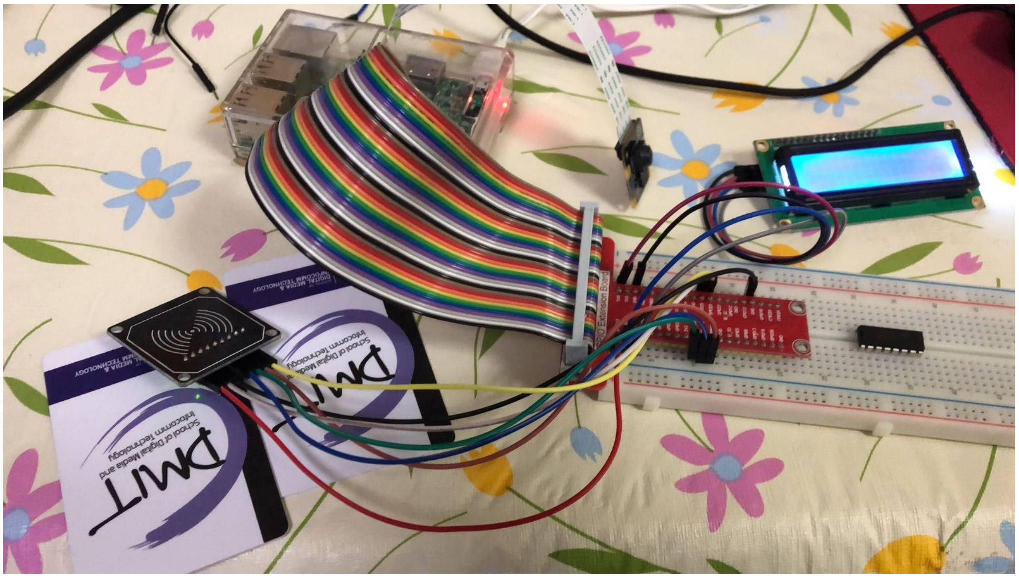
1	Raspberry Pi
3	1 kΩ Resistor (for LEDs)
1	10kΩ Resistor (for LDR)
3	LEDs (Different Colors)
3	Buttons
1	LDR
1	LCD
1	Pir Motion Sensor
X	Female -> Male jumper cables
X	Regular Jump Cables

Section 3 : Security

A. Hardware setup

A. This is what your finished connection should look like





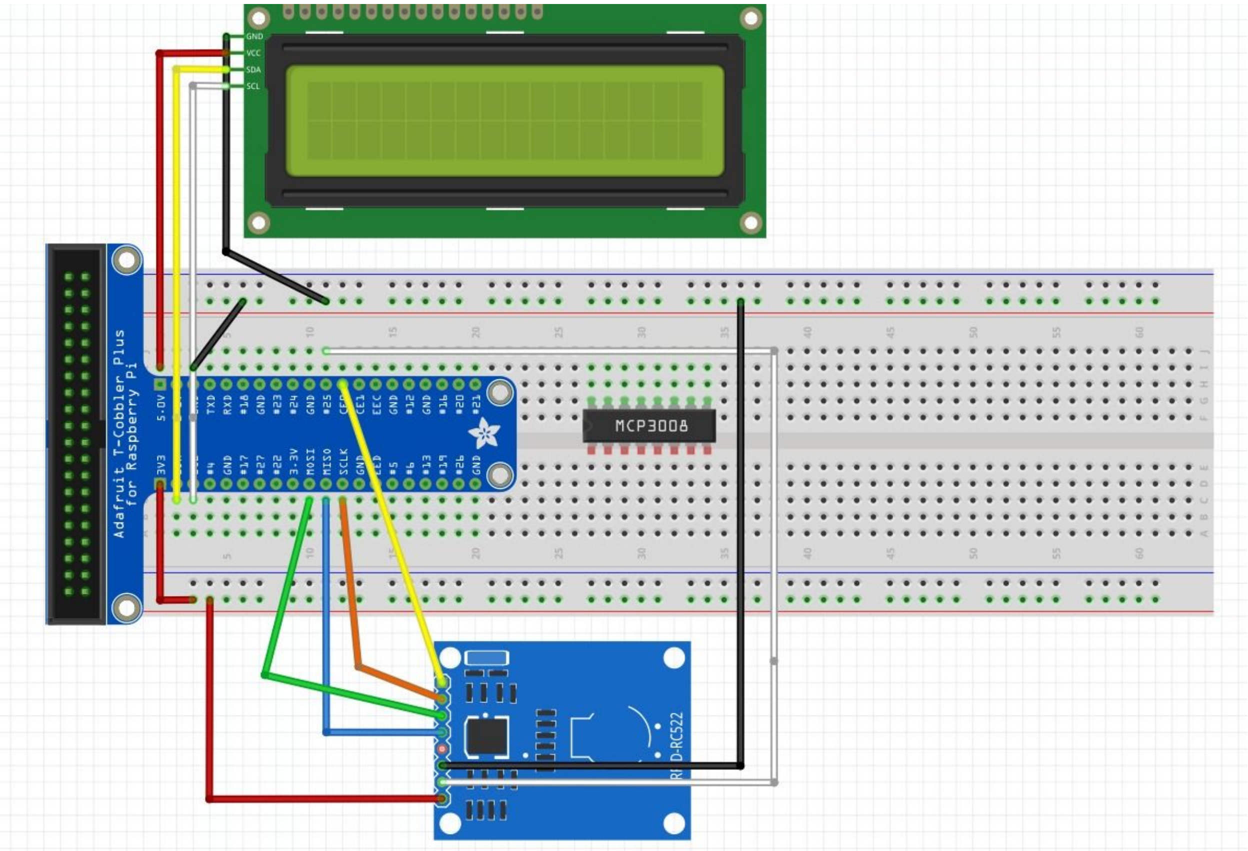
For the LCD, connect the following pins on the LCD to the RPI

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

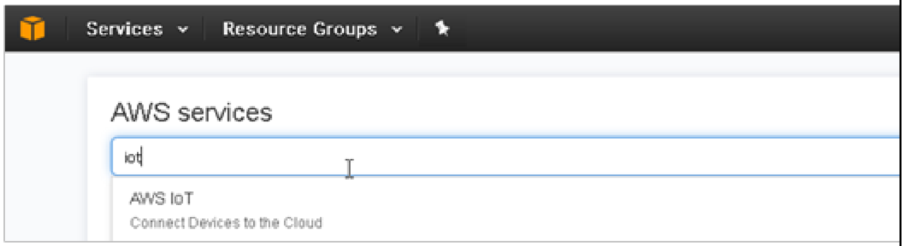
There are about 9 pins on the MFRCF522 reader given in your IoT kit. We will only be using 7 of them. Connect the pins on the MFRCF522 card reader to the RPi as indicated below.

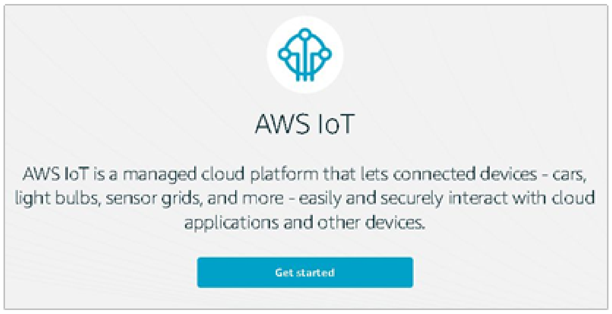
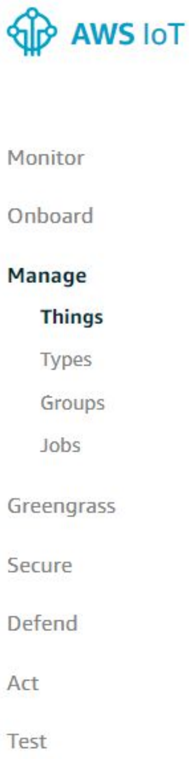
Jumper color	MFRCF522pin	RPi pin
Yellow	SDA	CE0
Orange	SCK	SCLK
Green	MOSI	MOSI
Blue	MISO	MISO
	IDR	
Black	GND	GND
White	RST	GPIO25
Red	3.3V	3.3V
	5V	

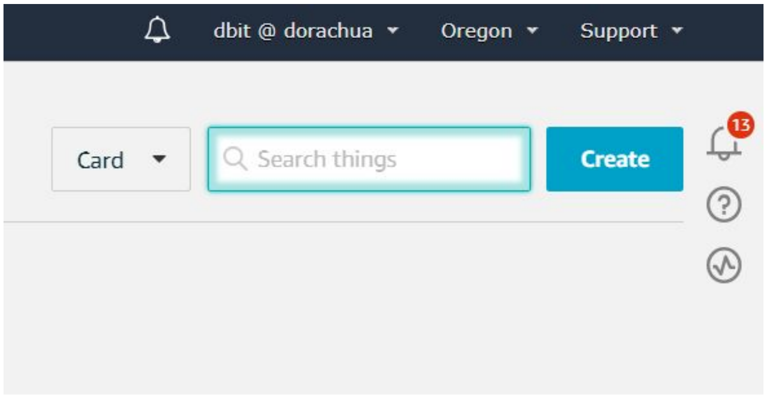
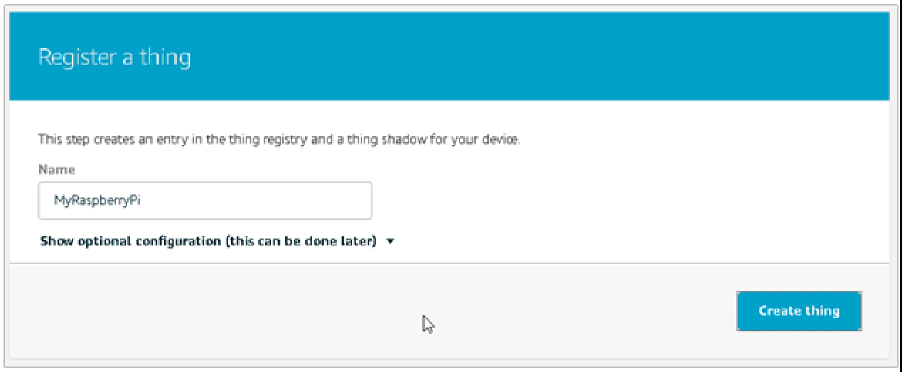
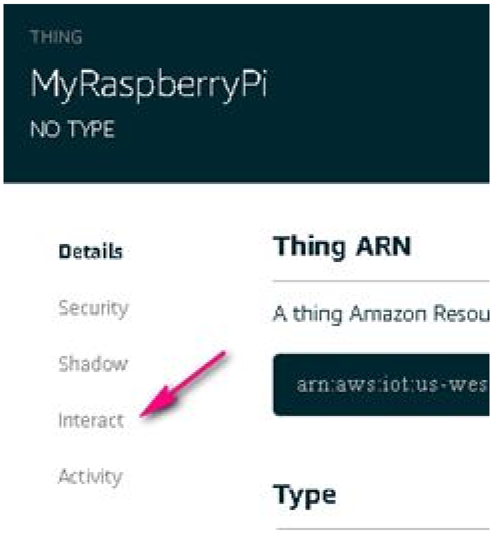
B. Fritzing Diagram

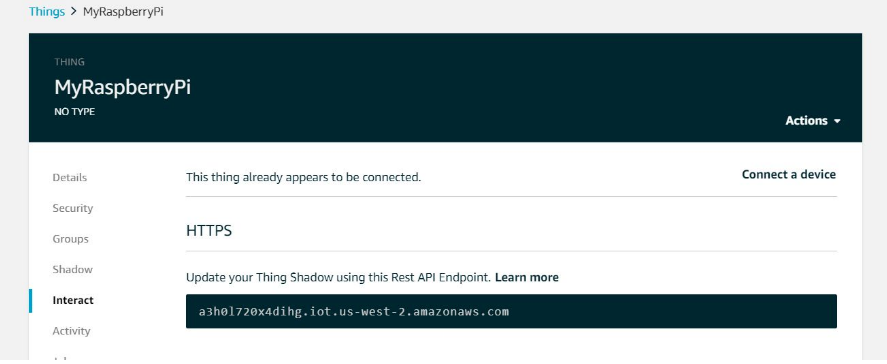
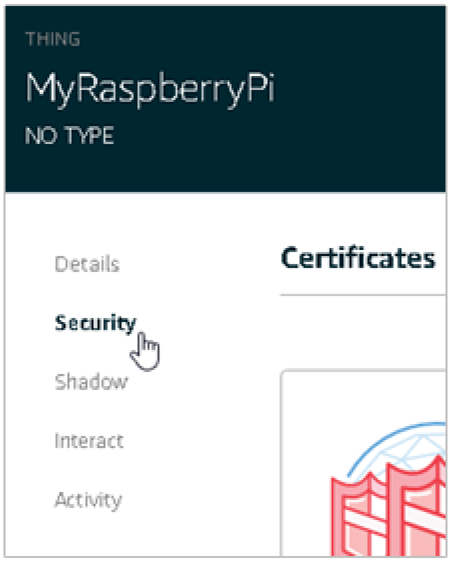
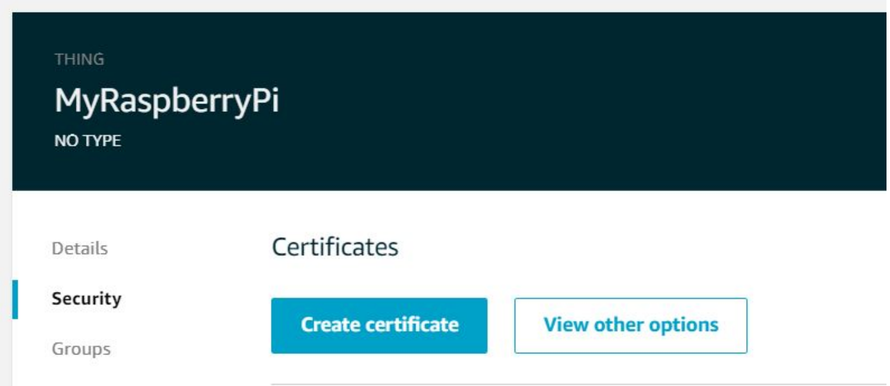


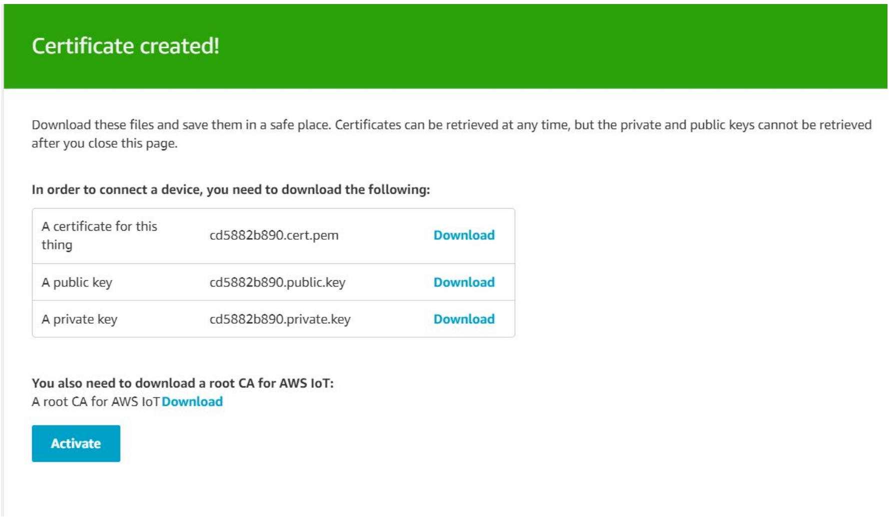
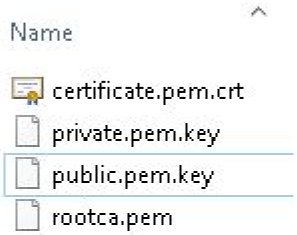
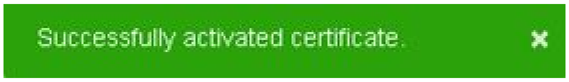
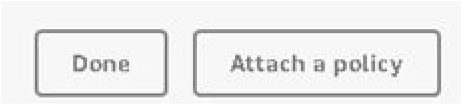
B. Configure AWS

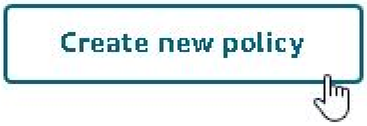
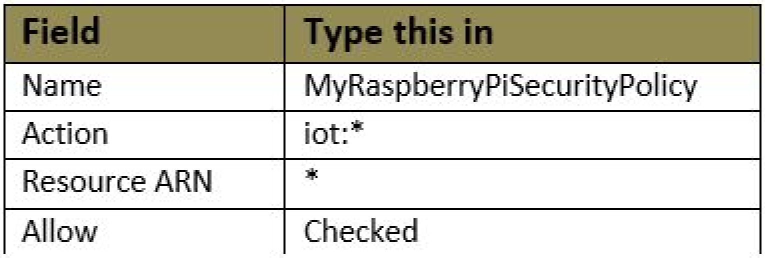
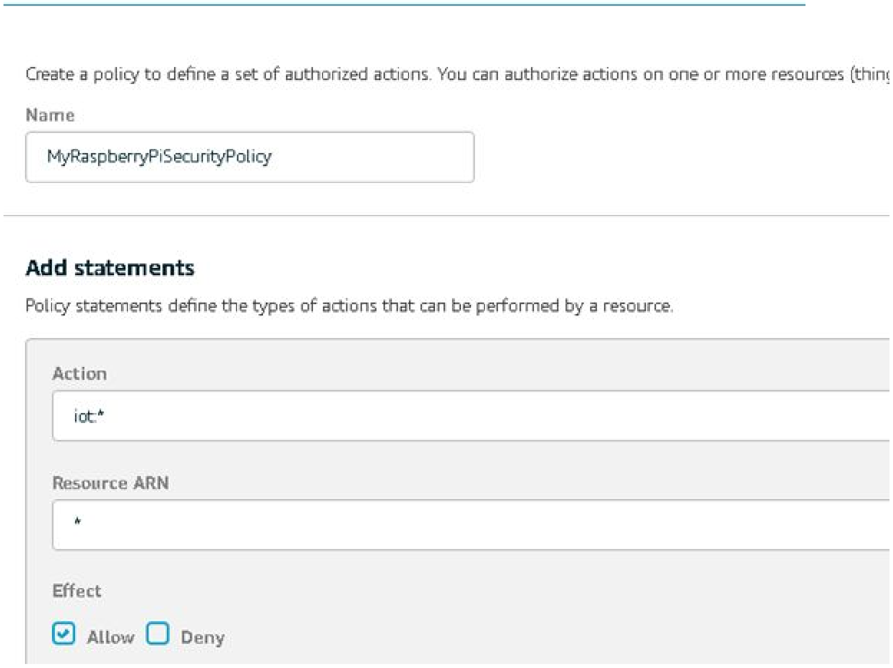
#	Description	Image
1	Turn on your Raspberry Pi and confirm you have an Internet connection. Sign in with your AWS console at https://aws.amazon.com In the AWS dashboard, type “AWS IoT” to access	 AWS services iot AWS IoT Connect Devices to the Cloud

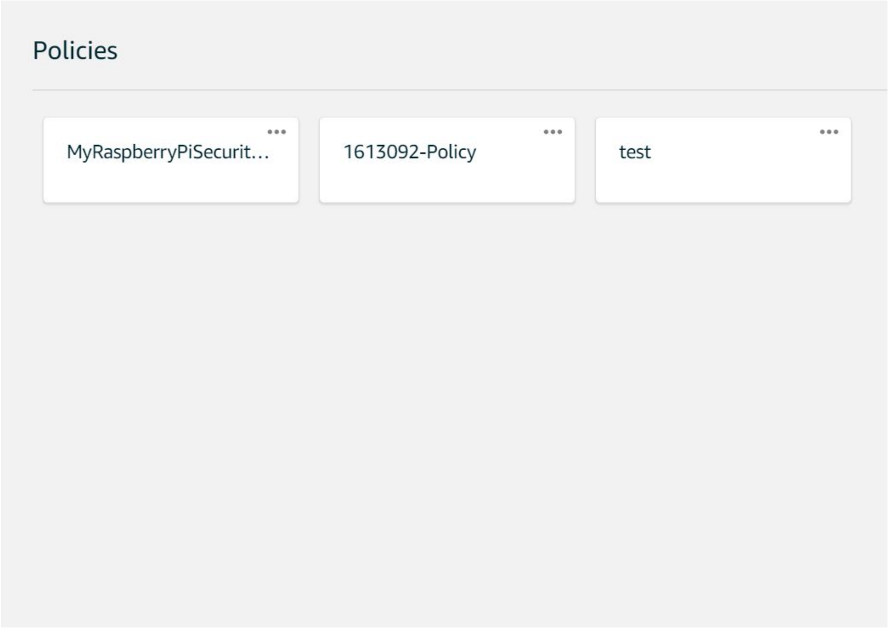
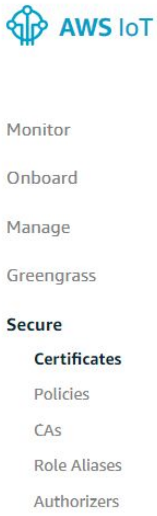
	the AWS IoT service.	
2	On the Welcome page, choose Get started	 AWS IoT AWS IoT is a managed cloud platform that lets connected devices - cars, light bulbs, sensor grids, and more - easily and securely interact with cloud applications and other devices.
3	In the left navigation pane, click “Manage” to expand it, then choose “Things”.	 AWS IoT Monitor Onboard Manage Things Types Groups Jobs Greengrass Secure Defend Act Test

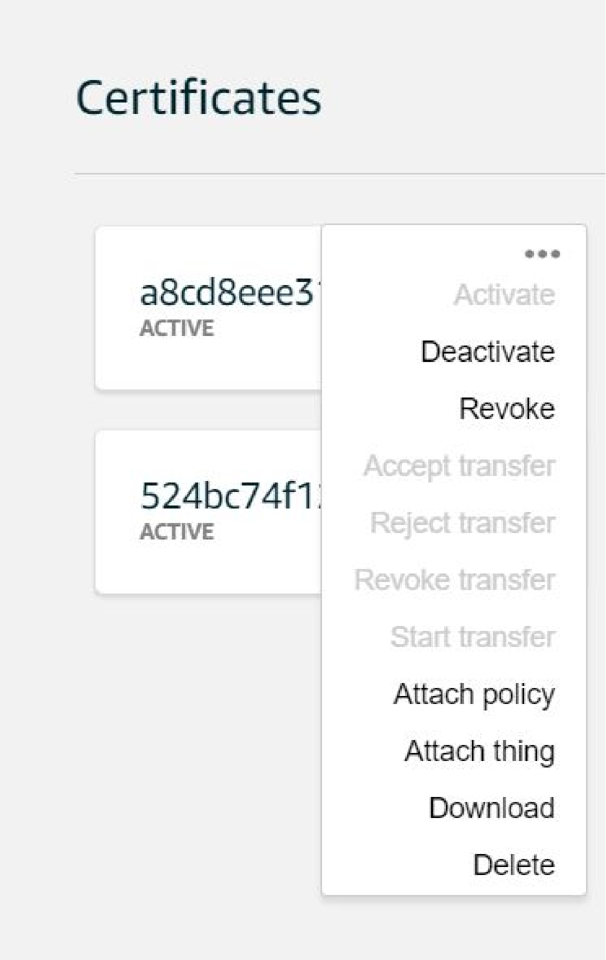
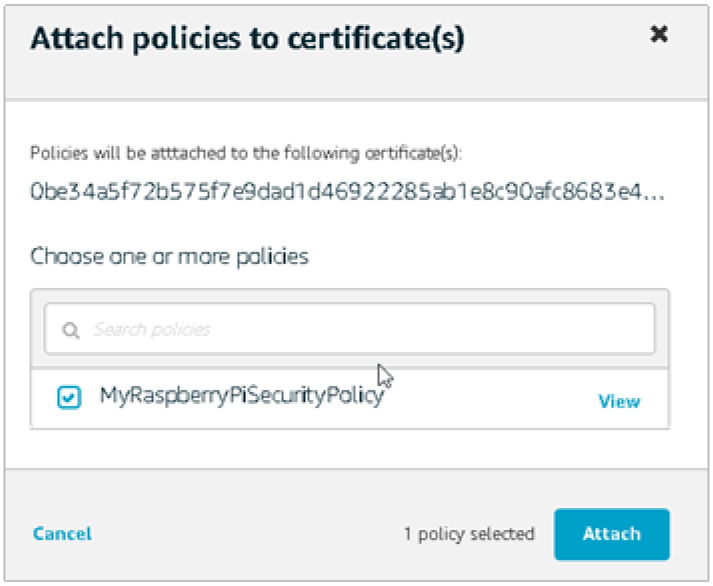
4	On the top right hand corner, you will see a button “create” Click on the button to create a thing	 dbit @ dorachua Oregon Support Card Search things Create 13
5	A thing represents a device whose status or data is stored in the AWS cloud. The Thing Shadows is the state of the device, e.g. is it “on” or “off”, is it “red” or “green” etc. Our “thing” here is our RPi, so let’s type “MyRaspberryPi” for the name. Click “Create thing”	 Register a thing This step creates an entry in the thing registry and a thing shadow for your device. Name MyRaspberryPi Show optional configuration (this can be done later)
6	On the Details page, choose **Interact**.	 THING MyRaspberryPi NO TYPE Details Security Shadow Interact Activity Thing ARN A thing Amazon Resou arn:aws:iot:us-wes Type

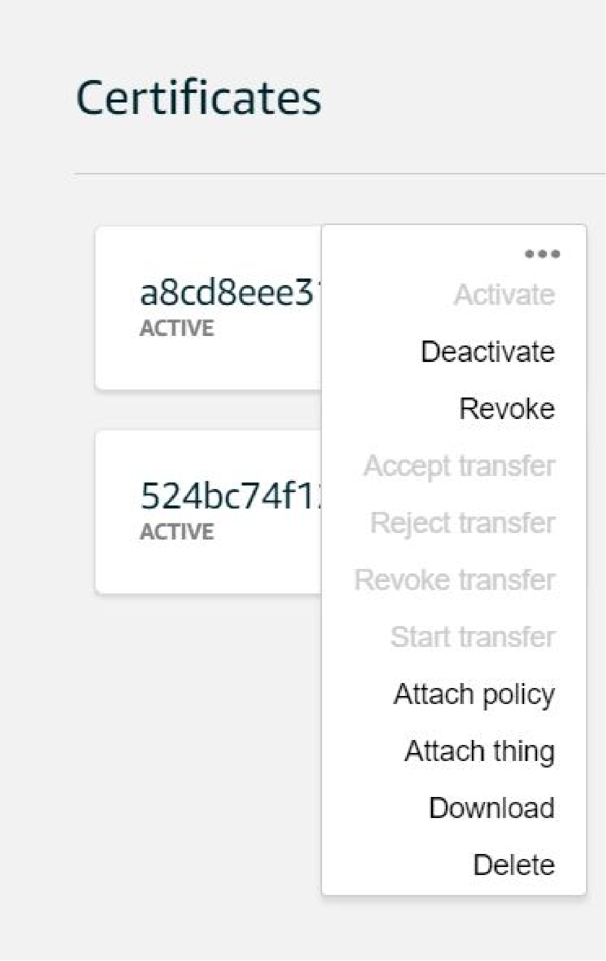
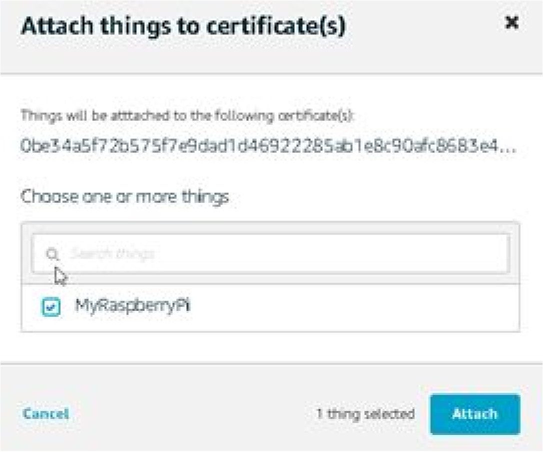
7	Copy and paste the REST API endpoint into a Notepad. You will need this value later.	 Things > MyRaspberryPi THING MyRaspberryPi NO TYPE Actions Details Security Groups Shadow Interact Activity Connect a device HTTPS Learn more This thing already appears to be connected. Update your Thing Shadow using this Rest API Endpoint. `a3h0l720x4dihg.iot.us-west-2.amazonaws.com`
8	Next, choose **Security**	 THING MyRaspberryPi NO TYPE Details Security Shadow Interact Activity Certificates
9	Choose “Create certificate” to generate an X.509 certificate and key pair.	 THING MyRaspberryPi NO TYPE Details Security Groups Certificates Create certificate View other options

10	After a while, you should see the following screen, where there are a total of **four** download links.	 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 cd5882b890.cert.pem Download A public key cd5882b890.public.key Download A private key cd5882b890.private.key Download **You also need to download a root CA for AWS IoT:** A root CA for AWS IoT Download Activate
11	Take note of the directory that you have downloaded the files into and change their filenames as shown	 Name certificate.pem.crt private.pem.key public.pem.key rootca.pem
12	Next, click the "Activate" button. Almost immediately, you should see "Successfully activated certificate" and the Activate button changes to "Deactivate"	 Successfully activated certificate. ✖
13	Next, click on the "Attach a policy" button that is near the bottom right-hand corner of the page.	

<table>
<tr>
<td>14</td>
<td>On the next page, choose “Create new policy”</td>
<td>
 </td>
</tr>
<tr>
<td>15</td>
<td>On the Create a policy page, key in the following configuration and then click “<b>Create</b>”</td>
<td>
<table>
<tr><th>Field</th><th>Type this in</th></tr>
<tr><td>Name</td><td>MyRaspberryPiSecurityPolicy</td></tr>
<tr><td>Action</td><td>iot:*</td></tr>
<tr><td>Resource ARN</td><td>*</td></tr>
<tr><td>Allow</td><td>Checked</td></tr>
</table>

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (thing
Name
MyRaspberryPiSecurityPolicy
<b>Add statements</b>
Policy statements define the types of actions that can be performed by a resource.
Action
iot:*
Resource ARN
*
Effect
☑ Allow ☐ Deny</td>
</tr>
</table>

16	You will then see a similar page like this	
17	On the left nav bar, click “Secure and then, Certificates”	 AWS IoT Monitor Onboard Manage Greengrass Secure Certificates Policies CAs Role Aliases Authorizers

18	From the certificate that you have created just now, click on the 3 dots and choose "**Attach Policy**"	 Certificates a8cd8eee3 ACTIVE 524bc74f1 ACTIVE ••• Activate Deactivate Revoke Accept transfer Reject transfer Revoke transfer Start transfer Attach policy Attach thing Download Delete
19	Check the "MyRaspberryPiSecurityPolicy" you created earlier and click "**Attach**" button.	 Attach policies to certificate(s) Policies will be atttached to the following certificate(s): 0be34a5f72b575f7e9dad1d46922285ab1e8c90afc8683e4... Choose one or more policies ☑ MyRaspberryPiSecurityPolicy 1 policy selected

20	Attach the “Thing” to the certificate	 Certificates a8cd8eee3 ACTIVE 524bc74f1 ACTIVE ••• Activate Deactivate Revoke Accept transfer Reject transfer Revoke transfer Start transfer Attach policy Attach thing Download Delete
21	In the Attach things to certificate(s) dialog box, select the check box next to the thing you created to represent your Raspberry Pi, and then choose Attach	Attach things to certificate(s) Things will be atttached to the following certificate(s): 0be34a5f72b575f7e9dad1d46922285ab1e8c90afc8683e4... Choose one or more things ☑ MyRaspberryPi

C. Install AWS Python Library

Install the AWS Python library with this command

```
sudo pip install AWSIoTPythonSDK
```

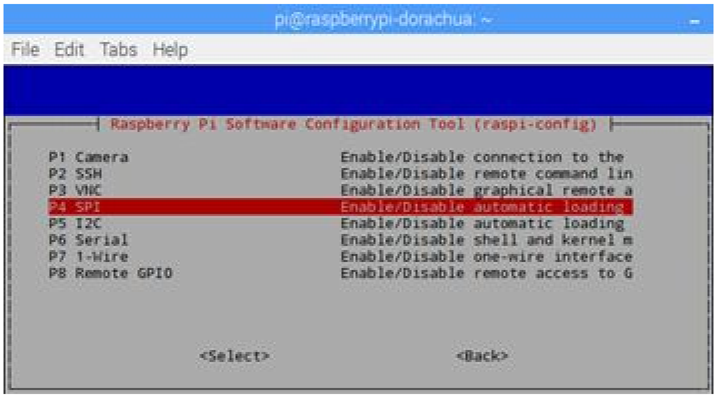
D. Install LCD Library

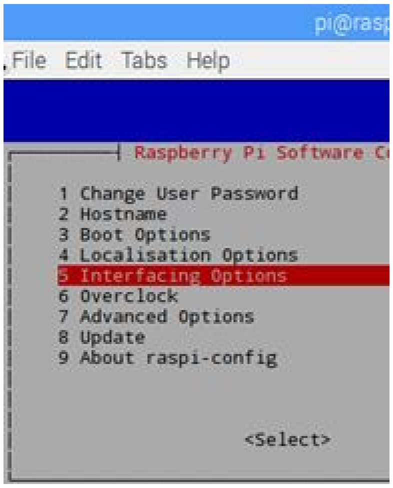
Install the LCD Library

```
sudo pip install rpi-lcd
```

E. Install RFID Library

#	Description
1	Enable SPI via raspi-config Run raspi-config, choose menu item “5 Interfacing Options” and enable SPI. `sudo raspi-config`


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>



2

Modify the /boot/config.txt to enable SPI

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

3

Ensure these lines are included in config.txt

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

4

Install the Python development libraries

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

5

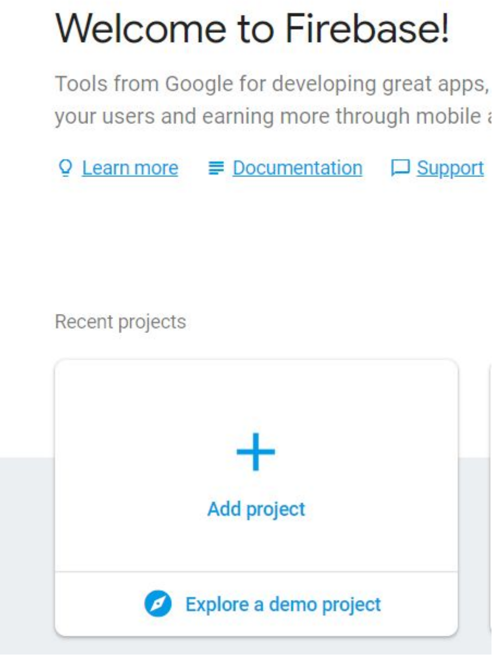
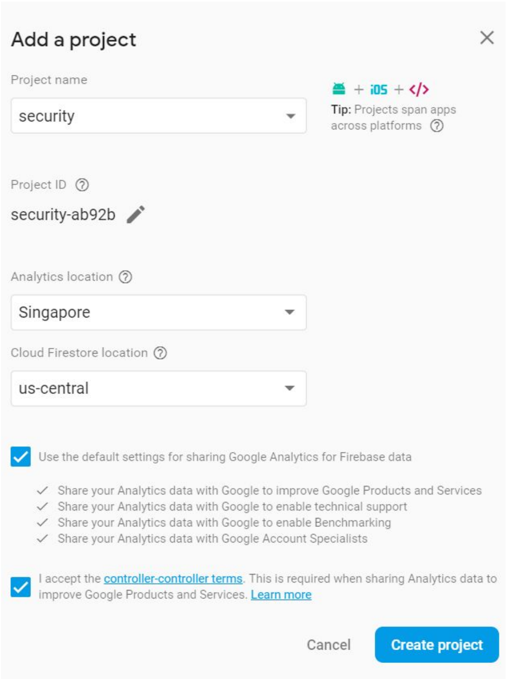
Set up the SPI Python libraries since the card reader uses the SPI interface

```
git clone https://github.com/lthiery/SPI-Py.git
cd /home/pi/SPI-Py
```

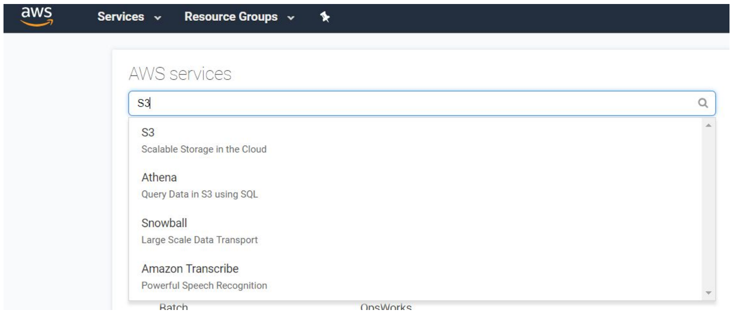
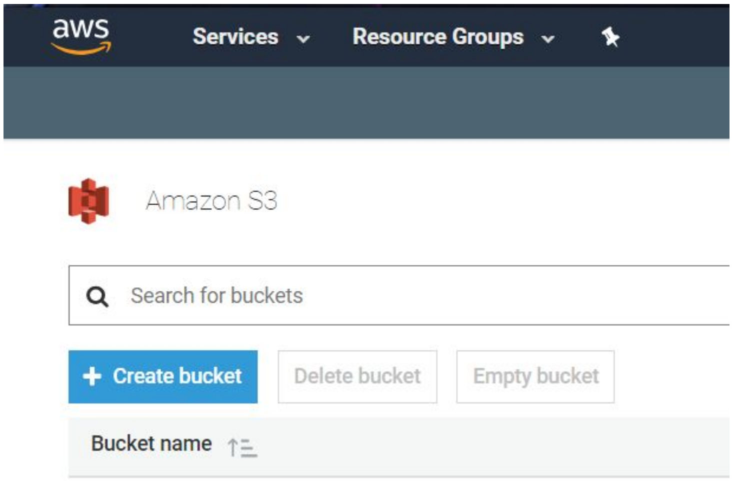
	`sudo python setup.py install`
6	Clone the MFRC522-python library and copy out MFRC522.py to your project directory `git clone https://github.com/mxgxw/MFRC522-python.git`

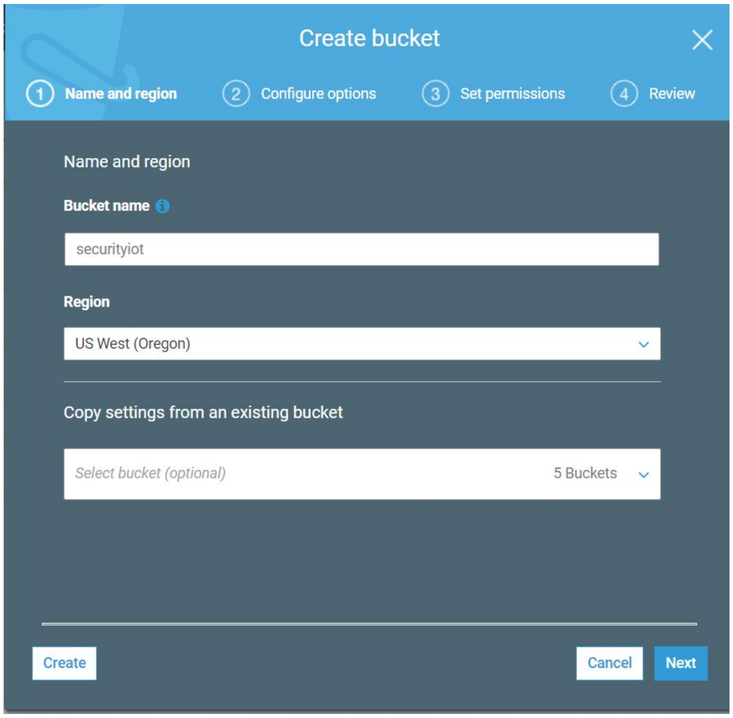
F. Setup firebase

#	Description	Image

1	To set up firebase, head to https://console.firebase.google.com After Signing in, click on add project	 Welcome to Firebase! Tools from Google for developing great apps, your users and earning more through mobile a Learn more Documentation Support Recent projects Add project Explore a demo project
2	Enter your project name whereby mine is security	 Add a project Project name: security Tip: Projects span apps across platforms Project ID: security-ab92b Analytics location: Singapore Cloud Firestore location: us-central ☑ Use the default settings for sharing Google Analytics for Firebase data ✓ Share your Analytics data with Google to improve Google Products and Services ✓ Share your Analytics data with Google to enable technical support ✓ Share your Analytics data with Google to enable Benchmarking ✓ Share your Analytics data with Google Account Specialists ☑ I accept the controller-controller terms. This is required when sharing Analytics data to improve Google Products and Services. Learn more

G. Setup S3 Storage

#	Description	Image
1	Log in your AWS console and search for S3	 aws Services Resource Groups AWS services S3 S3 Scalable Storage in the Cloud Athena Query Data in S3 using SQL Snowball Large Scale Data Transport Amazon Transcribe Powerful Speech Recognition Batch OpsWorks
2	Click Create Bucket	 aws Services Resource Groups Amazon S3 Search for buckets Create bucket Delete bucket Empty bucket Bucket name

3	Type in a unique name for your bucket and choose Region as "US West (Oregon)" which is us-west-2 Click "Create" button	 Create bucket Name and region Bucket name: securityiot Region: US West (Oregon) Copy settings from an existing bucket: Select bucket (optional) 5 Buckets ① Name and region ② Configure options ③ Set permissions ④ Review Create Cancel Next

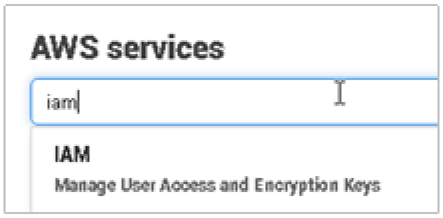
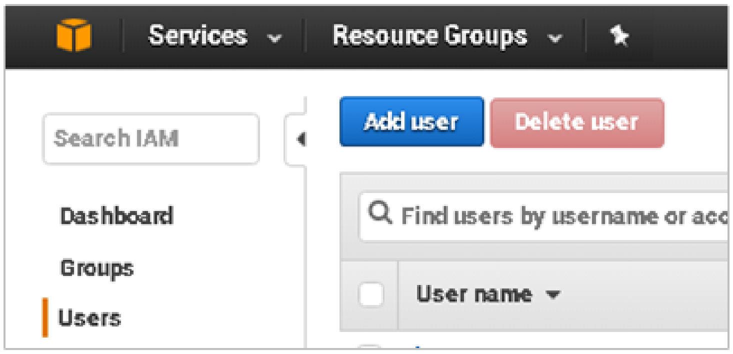
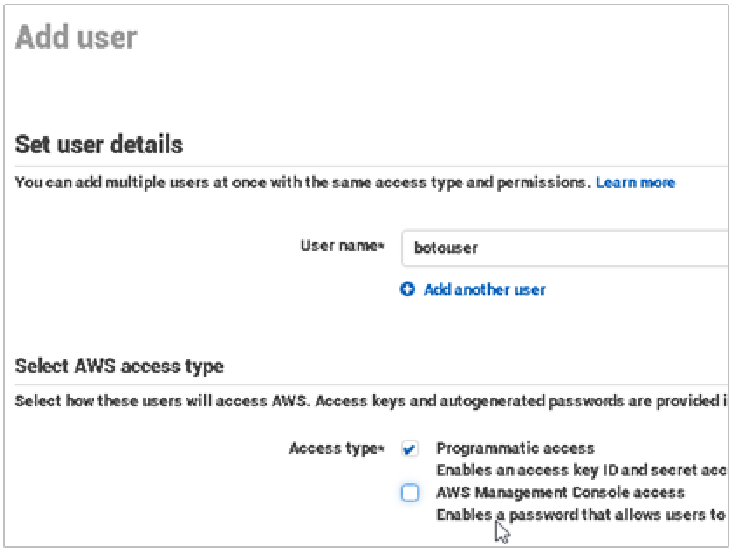
H. Install Boto on Raspberry Pi

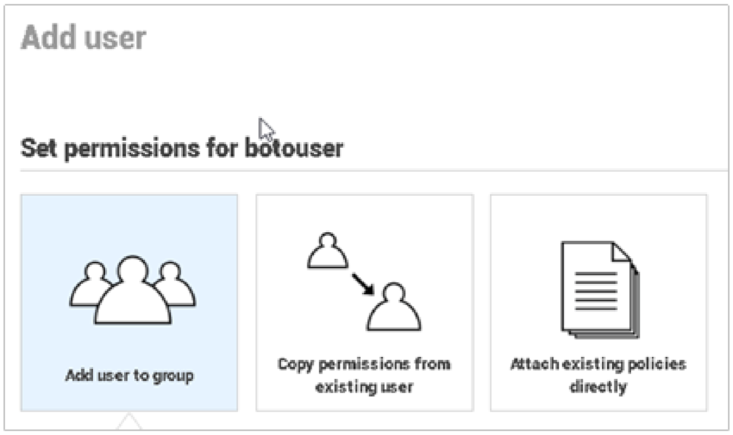
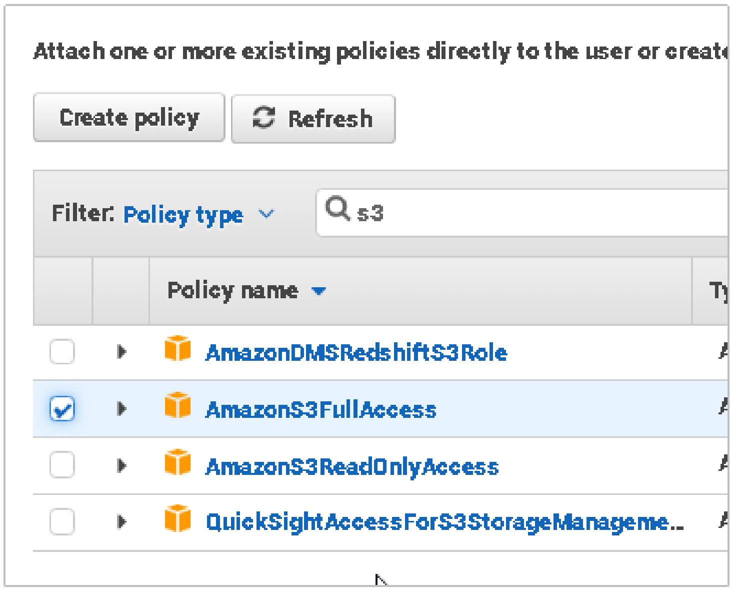
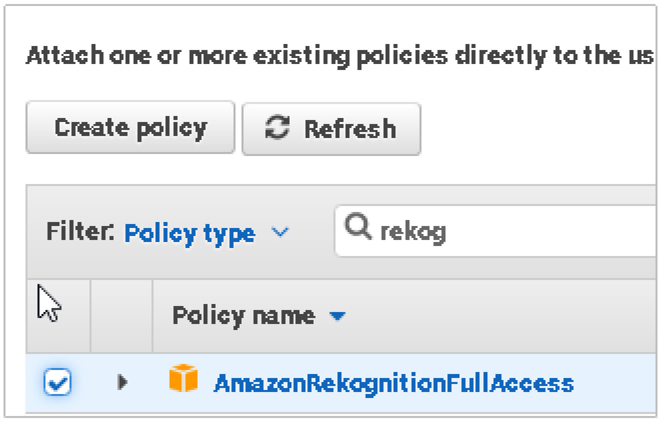
```
sudo pip install boto3
```

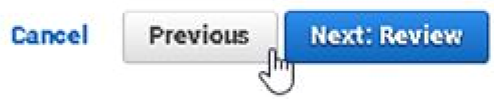
I. Install AWS CLI on Raspberry Pi

```
sudo pip install awscli
```

J. Credentials for AWS

#	Description	Image
1	Log in your AWS console and search for IAM	 AWS services iam IAM Manage User Access and Encryption Keys
2	Click on “Users” submenu, “Add user”	 Services Resource Groups Add user Delete user Search IAM Find users by username or acc Dashboard Groups Users User name
3	Create a new user botouser and enable programmatic access	 Add user Set user details You can add multiple users at once with the same access type and permissions. Learn more User name* botouser Add another user Select AWS access type Select how these users will access AWS. Access keys and autogenerated passwords are provided i Access type* ☑ Programmatic access Enables an access key ID and secret acc □ AWS Management Console access Enables a password that allows users to

4	In the next screen, click “Attach existing policies directly”	 Add user Set permissions for botouser Add user to group Copy permissions from existing user Attach existing policies directly
5	Scroll to the bottom of the page, search for “s3” and then check the “AmazonS3FullAccess” option	 Attach one or more existing policies directly to the user or create Create policy · Refresh Filter: Policy type · s3 Policy name ☐ AmazonDMSRedshiftS3Role ☑ AmazonS3FullAccess ☐ AmazonS3ReadOnlyAccess ☐ QuickSightAccessForS3StorageManageme...
6	Next, search for “Rekognition” and then check the “AmazonRekognitionFullAccess” option	 Attach one or more existing policies directly to the us Create policy · Refresh Filter: Policy type · rekog Policy name ☑ AmazonRekognitionFullAccess

7	Click the “Next: Review” button at the bottom of the page on the right, then on the next page, click “Create user”	
8	You should see the success page above with a link to download a csv file. **Make sure you download the .csv file to your laptop** You will need to refer to them later.	-

K. AWS Configure

On your Raspberry Pi, navigate to the directory where your Python code will be stored

```
cd ca2
```

Type the following command in your Raspberry Pi terminal so that you can use the AWS CLI to configure your credentials file:

```
aws configure
```

Enter the Access Key ID and Secret Access Key id you obtained from the previous section

Reminder that you would need to change a few things in the code that will be provided later on

In code which will be provided later on, you will need to change current code shown below to your own firebase application database url in the code

```
firebase = firebase.FirebaseApplication('https://iotca2-12f48.firebaseio.com', None)
```

In code which will be provided later on, you will need to change current code shown below to your own hostname of the thing created in aws which you had saved just now

```
host = "a280tk19mi5ck7.iot.us-west-2.amazonaws.com"
```

In code which will be provided later on, you will need to change current code shown below input in your credentials from the csv file downloaded from s3

```
access_key_id = 'AKIAJNPGQERG7Z5VFIFA'
secret_access_key = 'w8KxV67zAKMWtkx3RslD6kzgRqqjskuFy61ss22l'
```

In code which will be provided later on, you will need to Change the current code below to your own s3 bucket unique name

```
bucket_name = 'iotsecurity'
```

L. security.py code

Create new python file

```
sudo nano security.py
```

Input in current codes

```
 # Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from time import sleep
import RPi.GPIO as GPIO
import MFRC522
from datetime import datetime
from picamera import PiCamera
```

```
import os
import tinys3
import json
from rpi_lcd import LCD
import boto3
import botocore
from firebase import firebase


CONSUMER_KEY = 'h5Sis7TXdoUVncrpjSzGAvhBH'
CONSUMER_SECRET = 'ZfDVxc4aTd9doGmBQO3HiSKKzxSTKT4C3g0B3AGx8eETCJm2rY'
ACCESS_KEY = '988333099669901312-YDLEQN1weW2n1JP4lxJcFPppCsbvzQh'
ACCESS_SECRET = 'K2IlUPur6jx7DO5S0HhhZW29H5AQFOvkMMevSsk9ZzwLk'

firebase = firebase.FirebaseApplication('https://iotca2-12f48.firebaseio.com',
None)

host = "a280tk19mi5ck7.iot.us-west-2.amazonaws.com"
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

# photo properties
image_width = 800
image_height = 600
file_extension = '.png'

access_key_id = 'AKIAJNPGQERG7Z5VFIFA'
secret_access_key = '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'
bucket_name = 'iotsecurity'


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

my_rpi.configureOfflinePublishQueueing(-1)  # Infinite offline Publish queueing
my_rpi.configureDrainingFrequency(2)  # Draining: 2 Hz
my_rpi.configureConnectDisconnectTimeout(10)  # 10 sec
my_rpi.configureMQTTOperationTimeout(5)  # 5 sec

# camera setup
camera = PiCamera()
camera.resolution = (image_width, image_height)
camera.awb_mode = 'auto'


#LCD Setup
lcd = LCD()

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


def waitForRFIDScan():
   done = False
   while not done:
    (status,TagType) = mfrc522.MFRC522_Request(mfrc522.PICC_REQIDL)
    if status == mfrc522.MI_OK:
     # Get the UID of the card
     (status,uid) = mfrc522.MFRC522_Anticoll()
     done = True
     return uid

def uploadToS3(file_name):
   filepath = file_name + file_extension
   camera.capture(filepath)
   conn = tinys3.Connection(access_key_id, secret_access_key)
   f = open(filepath, 'rb')
   conn.upload(filepath, f, bucket_name,
              headers={
              'x-amz-meta-cache-control': 'max-age=60'
              })
   if os.path.exists(filepath):
       os.remove(filepath)


def checkRFIDNumber(rfidnumber):
   return rfidnumber == [136, 4, 133, 233, 224]

# # # Custom MQTT message callback
# def customCallback(client, userdata, message):
#     print("Received a new message: ")
#     data = json.loads(message.payload)
#     try:
#         similarity = data[1][0]['Similarity']
#         print("Received similarity: " + str(similarity))
#         if(similarity >= 90):
#             print("Access allowed, opening doors.")
#             print("Thank you!")
#     except:
#         pass
#     print("Finished processing event.")


# Custom MQTT message callback
def customCallback(client, userdata, message):
    print("Received a new message: ")
    print(message.payload)
    print("from topic: ")
    print(message.topic)
    print("--------------\n\n")

# Connect and subscribe to AWS IoT
my_rpi.connect()
my_rpi.subscribe("security/entry", 1, customCallback)
sleep(2)

# Publish to the same topic in a loop forever
while True:
   print("waiting..")
```

```
    scan = waitForRFIDScan()
    print(scan)
    if(checkRFIDNumber(scan)):
        print("RFID correct, taking photo...")
        lcd.text('Welcome', 1)
        file_name = "user"
        uploadToS3(file_name)
        my_rpi.publish("security/entry", "User entered", 1)
        datestr = str(datetime.now())
        imgUrl = "https://s3-us-west-2.amazonaws.com/iotsecurity/user.png"
        data={ 'Date': datestr, 'UID': str(scan), 'image': imgUrl}
        result = firebase.post('/entry/',data)
        sleep(5)
        lcd.clear()
    else:
        print("Bad RFID - Access Denied")
        lcd.text('Access Denied', 1)
        my_rpi.publish("security/entry", "Intruder Detected", 1)
        datestr = str(datetime.now())
        data={ 'Date': datestr, 'UID': str(scan)}
        result = firebase.post('/entry/',data)
        sleep(5)
        lcd.clear()
        # sleep(10)
```

M. Run security.py code

Run code to get security system working

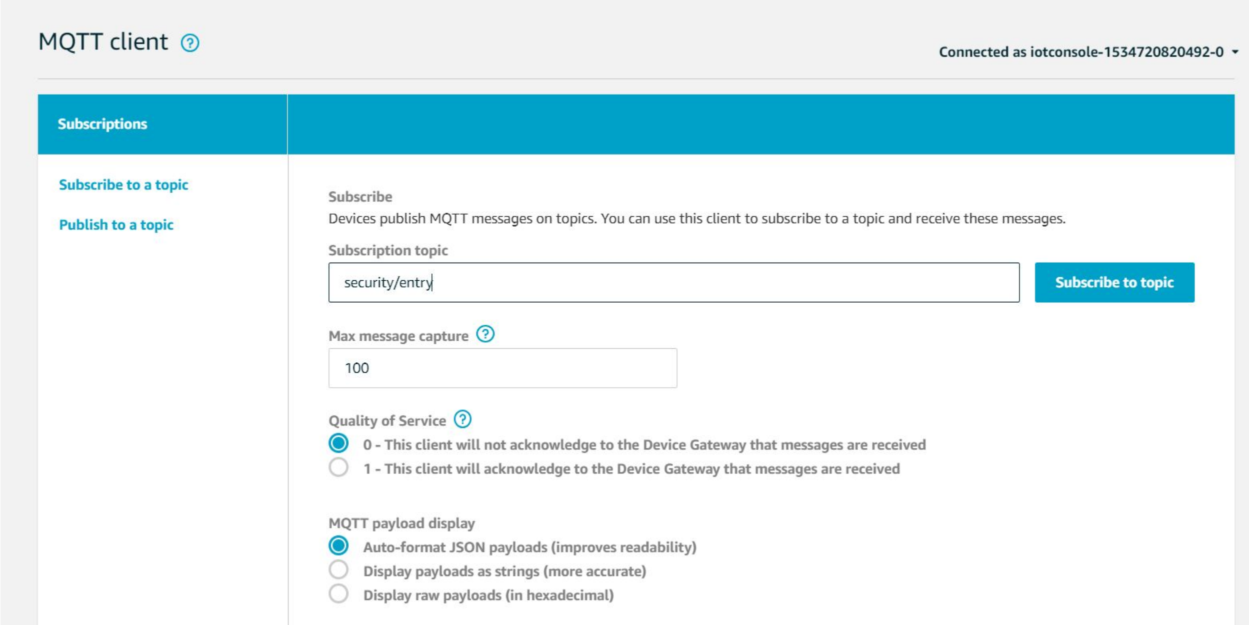
```
python security.py
```

N. View MQTT

In the AWS IoT console, in the left navigation pane, choose Test.

Type in the topic on which your thing publishes. In our case, it is “security/entry”

Click “Subscribe to topic”



Subscribe

Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

Subscription topic: security/entry

Max message capture: 100

Quality of Service

● 0 - This client will not acknowledge to the Device Gateway that messages are received

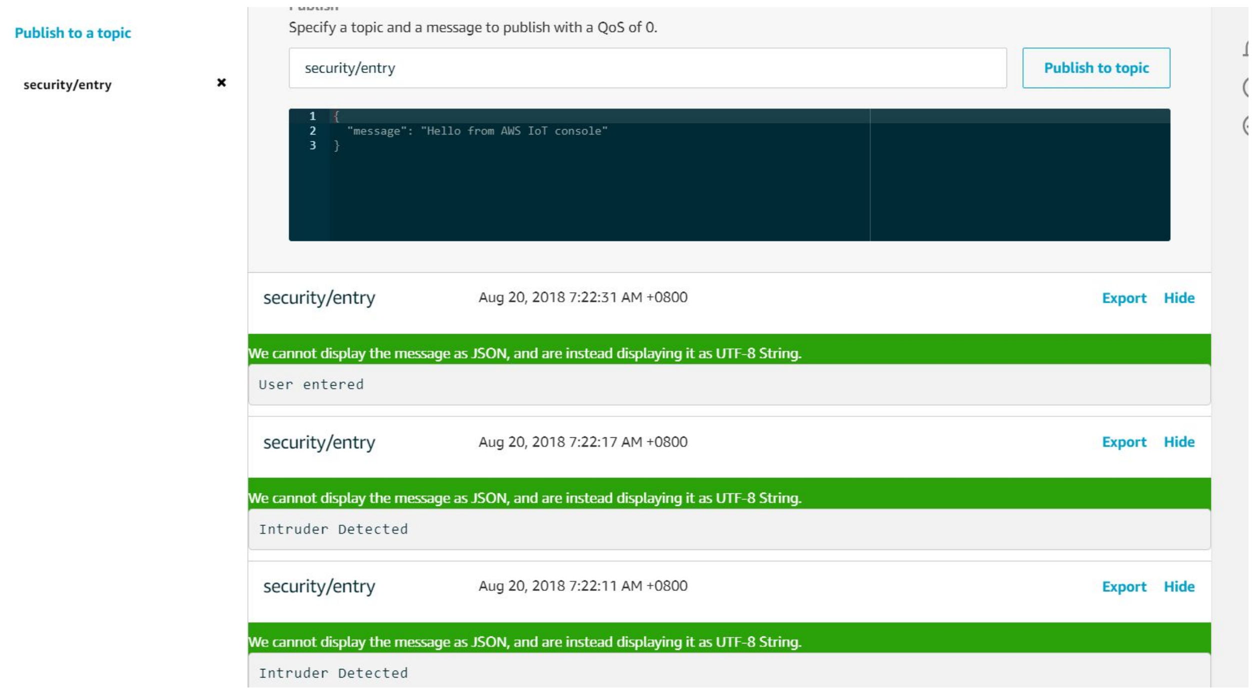
○ 1 - This client will acknowledge to the Device Gateway that messages are received

MQTT payload display

● Auto-format JSON payloads (improves readability)

○ Display payloads as strings (more accurate)

○ Display raw payloads (in hexadecimal)



Specify a topic and a message to publish with a QoS of 0.

security/entry

```
{
  "message": "Hello from AWS IoT console"
}
```

security/entry Aug 20, 2018 7:22:31 AM +0800

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

```
User entered
```

security/entry Aug 20, 2018 7:22:17 AM +0800

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

```
Intruder Detected
```

security/entry Aug 20, 2018 7:22:11 AM +0800

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

```
Intruder Detected
```

O. Image Recognition

Create a new python file

```
sudo nano imagerecognition.py
```

Input in the current code and change replace bucket name with your own

```
import boto3
import botocore
from picamera import PiCamera
from time import sleep

# Set the filename and bucket name
BUCKET = 'iotsecurity' # replace with your own unique bucket name
location = {'LocationConstraint': 'us-west-2'}
file_path = "/home/pi/Desktop"
file_name = "test.jpg"

def takePhoto(file_path,file_name):

    with PiCamera() as camera:
        #camera.resolution = (1024, 768)
        full_path = file_path + "/" + file_name
        camera.capture(full_path)
        sleep(3)

def uploadToS3(file_path,file_name, bucket_name,location):
    s3 = boto3.resource('s3') # Create an S3 resource
    exists = True

    try:
        s3.meta.client.head_bucket(Bucket=bucket_name)
    except botocore.exceptions.ClientError as e:
        error_code = int(e.response['Error']['Code'])
        if error_code == 404:
            exists = False

    if exists == False:
        s3.create_bucket(Bucket=bucket_name,CreateBucketConfiguration=location)

    # Upload the file
    full_path = file_path + "/" + file_name
    s3.Object(bucket_name, file_name).put(Body=open(full_path, 'rb'))
    print("File uploaded")


 def detect_labels(bucket, key, max_labels=10, min_confidence=90,
 region="us-west-2"):
     rekognition = boto3.client("rekognition", region)
     response = rekognition.detect_labels(
         Image={
             "S3Object": {
                 "Bucket": bucket,
                 "Name": key,
             }
         },
         MaxLabels=max_labels,
         MinConfidence=min_confidence,
     )
     return response['Labels']


 takePhoto(file_path, file_name)
 uploadToS3(file_path,file_name, BUCKET,location)
 for label in detect_labels(BUCKET, file_name):
     print("{Name} - {Confidence}%".format(**label))
```

Run code to recognize things that are outside your home!

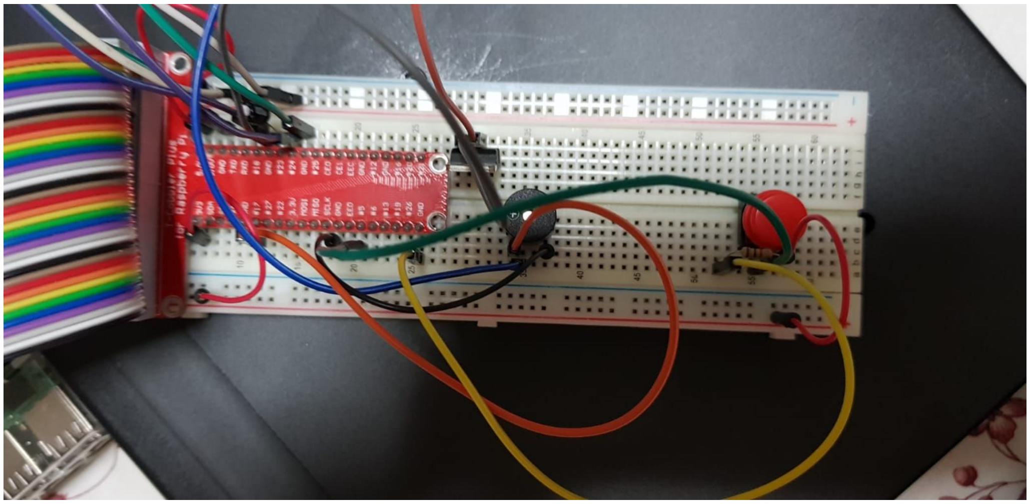
```
python imagerecognition.py
```

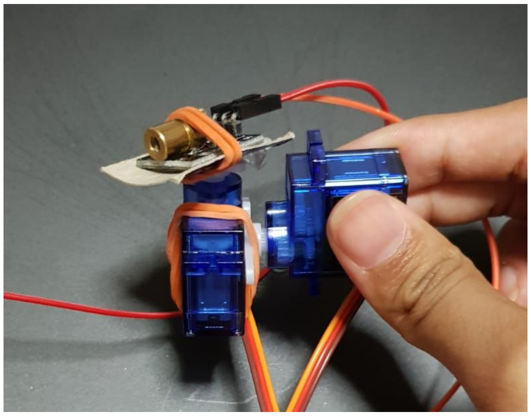
Section 4 : Defence

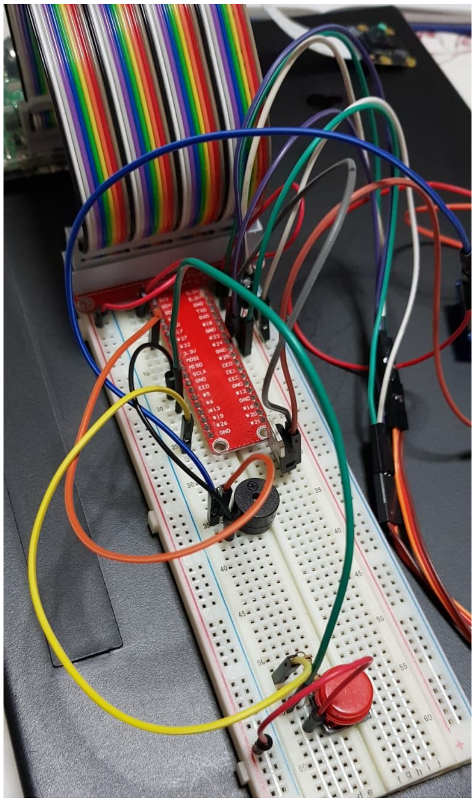
A. Hardware Setup

a. The RPi Connections

This is what your finished connection should look like.

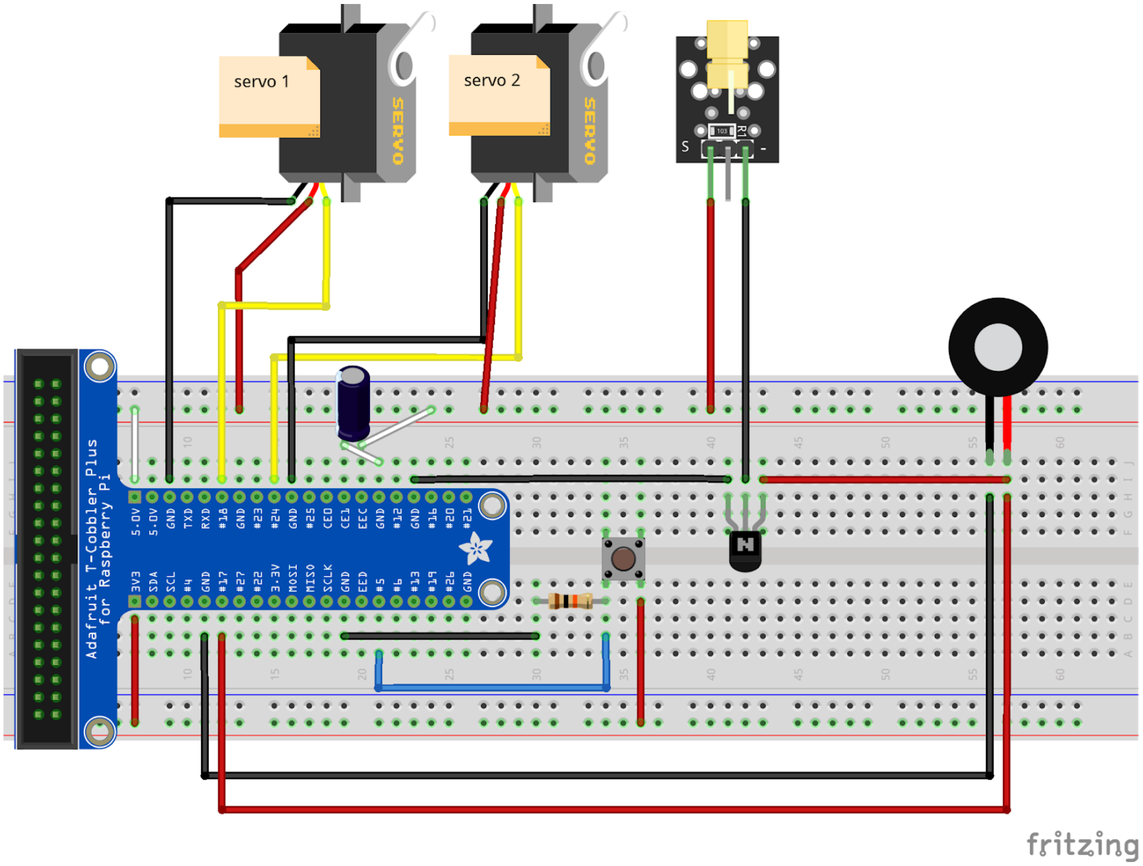






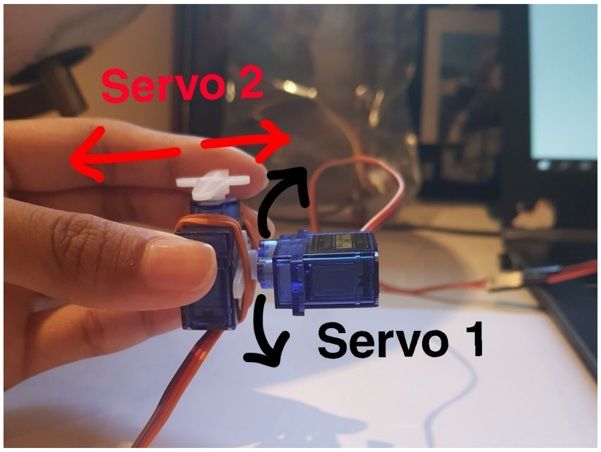
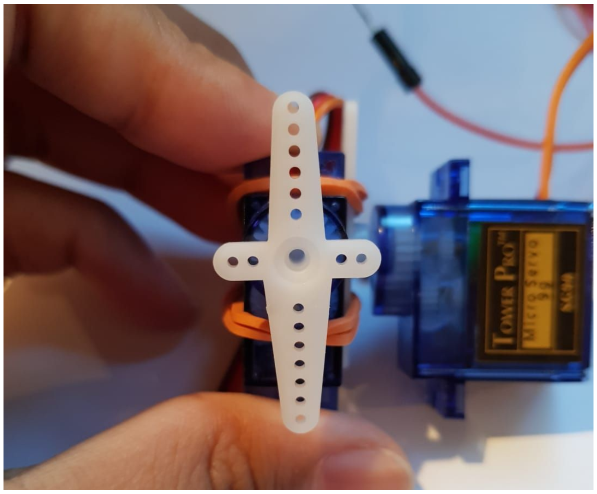
b. The Fritzing Diagram

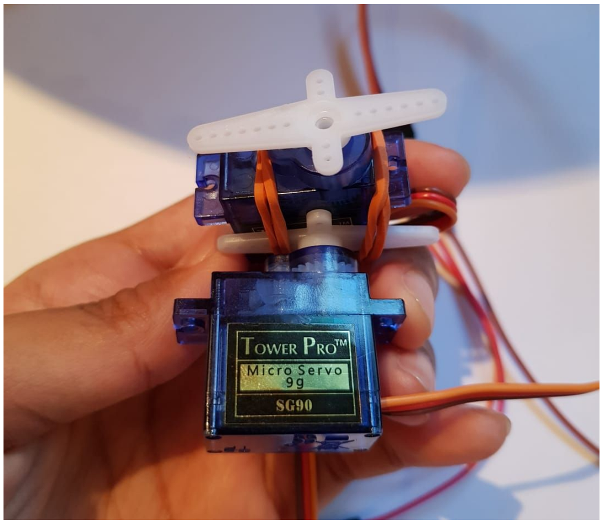
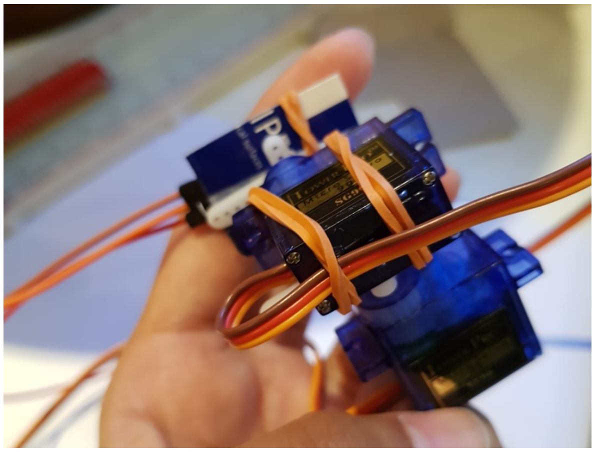
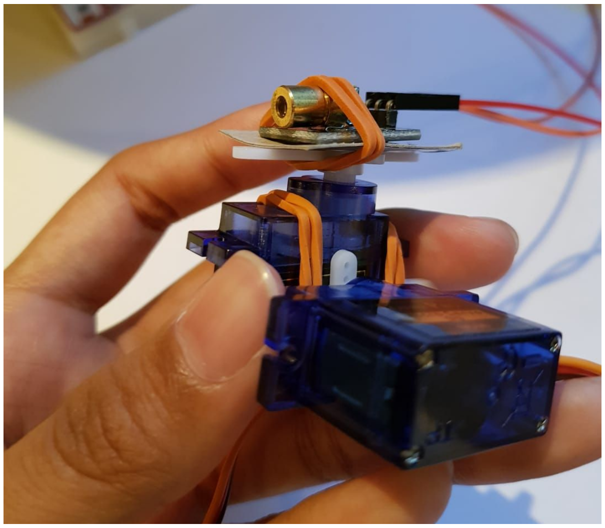
The fritzing diagram looks like this. Set up your RPi like so!

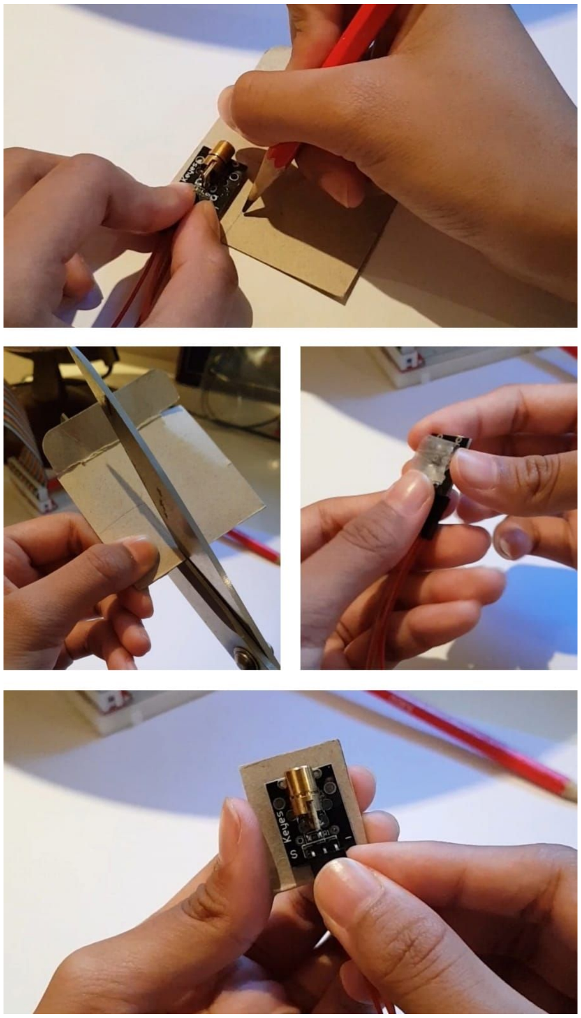
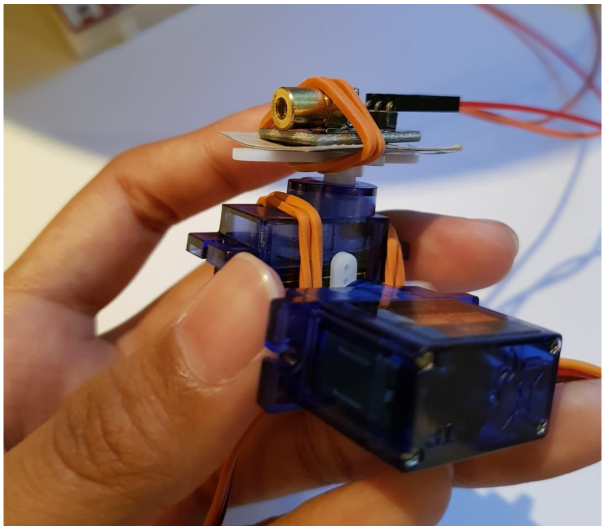

servo 1
SERVO
servo 2
SERVO
S
-
Adafruit T-Cobbler Plus for Raspberry Pi
5.0V 5.0V GND TXD RXD #18 GND #23 #24 GND #25 CE0 CE1 EEC GND #12 GND #16 #20 #21
3V3 SDA SCL #4 GND #17 #27 #22 3.3V MOSI MISO SCLK GND EED #5 #6 #13 #19 #26 GND
fritzing

c. Connect The Components

The following steps will describe how to get the laser turret setup.

#	Description	Image
1	Here’s what you want to achieve. Servo 1 allows us to aim the laser module higher or lower (black arrows). Servo 2 allows us to aim the laser module left or right (red arrows).	
2	Attach the cross head for BOTH servos.	

3	User rubber bands/zip ties to attach servo one (top) to servo 2 (bottom).	
4	Tuck in the cable of servo 2 under the rubber band as well. This is so that when the servo rotates, the wire won't be caught in the motion.	
5	Next, we'll put together the laser module.	

6	Put something hard under the laser module. This will allow us a larger and sturdier surface to attach the laser module to the servo. I used a scrap piece of cardboard and tape.	
7	Attach the laser module to the servo by using rubber band and some tape on the underside of the cardboard. Then you're done!	

The following steps will explain the main wiring. Don’t forget the resistors!

Part	**Cable Color**	**PIR Pin**	**RPi Pin**
Servo 1	Orange	PVM	#18
	Red	5V	5V Rail
	Brown	GND	GND
Servo 2	Orange	PVM	#24
	Red	5V	5V Rail
	Brown	GND	GND
Laser Transmitter Module	Orange	GND	GND / Middle Leg (Transistor)
	Red	S	5V Rail
Buzzer	Blue	VOUT	#17
	Green	GND	GND
Attack Button	Red	VOUT	3V
	Blue	GPIO	#5
	Black	GND	GND
Transistor	Black	Right Leg	GND
		Middle Leg	Laser GNT
		Left Leg	GND

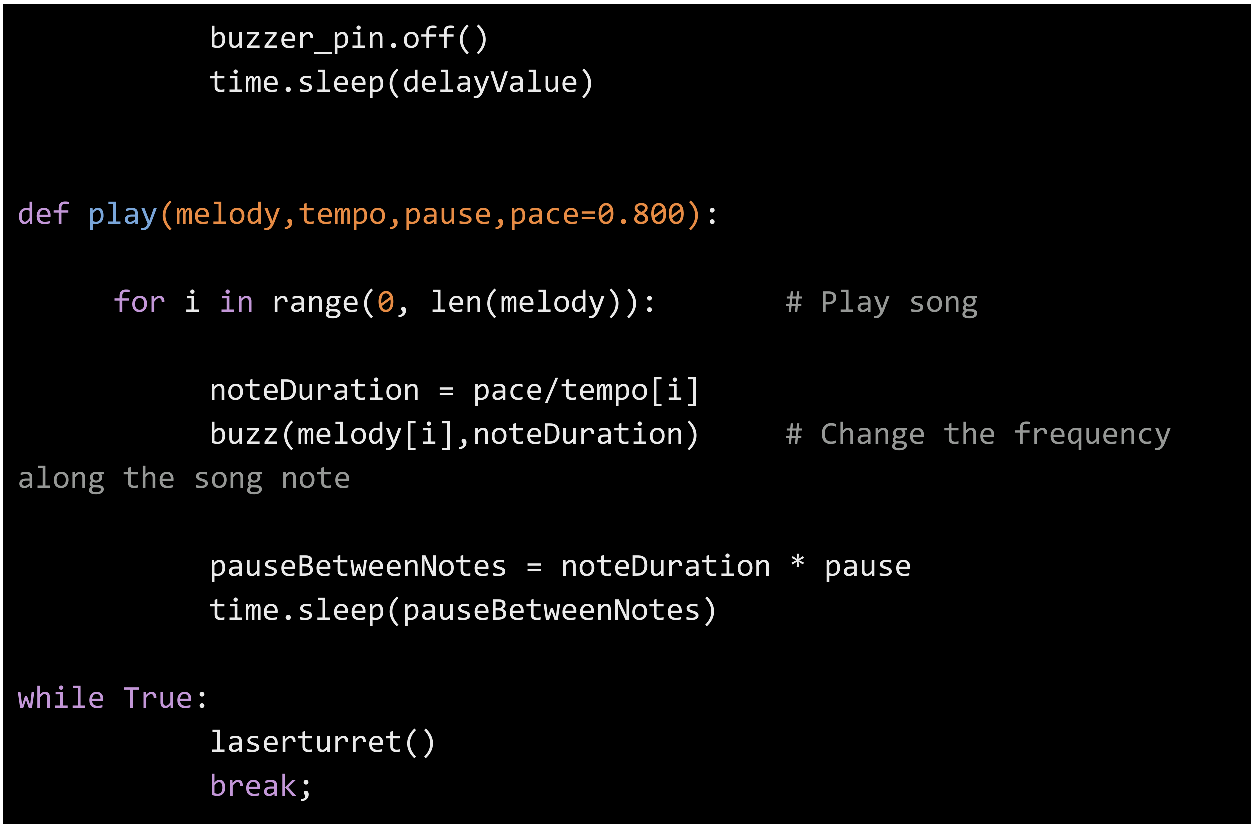
Capacitor		Long Leg	5V
		Short Leg	GND

B. Software

a. laserturret.py

The codes below, when run, will trigger the laser turret. The laser turret will shoot in random directions, in random bursts and speed.

```
from gpiozero import LED, Buzzer, Button, Servo
import time
from signal import pause
import random

#led = LED(12)
#pir = MotionSensor(19, sample_rate=5,queue_len=1)
buzzer_pin = Buzzer(17)
attack = Button(5, pull_up=False)
#reset = Button(6, pull_up=False)
servo1 = Servo(18)
servo2 = Servo(24)

def ledON():
    led.on()
    print("LED is on")

def ledOFF():
    led.off()
    print("LED is off")

def fire():
    print("weapons hot")
    buzzer_pin.on()
    time.sleep(0.1)
    buzzer_pin.off()

def laserturret():
        timeBetweenBurst = random.uniform(0.2,1)
        timeBetweenShots = random.uniform(0.05,0.2)
        servo1start = random.randrange(-1,1)
        servo1end = random.randrange(-1,1)
        servo2start = random.randrange(-1, 1)
        servo2end = random.randrange(-1,1)
        numShots = random.randrange(5,20)

        servo1change = (servo1end - servo1start)/numShots
        servo2change = (servo2end - servo2start)/numShots
        servo1.value = servo1start
        servo2.value = servo2start
        time.sleep(0.1)
        shot = 0

        detail = [timeBetweenBurst,timeBetweenShots, servo1.value, 
servo2.value, numShots]
        print(detail)

        while shot<numShots:
            shot+= 1
            servo1.value = servo1start
            servo2.value = servo2start

            servo1start = servo1change
            servo2start = servo2change

            fire()
            time.sleep(timeBetweenShots)

        time.sleep(timeBetweenBurst)

notes = {
    'B0' : 31,
    'C1' : 33, 'CS1' : 35,
    'D1' : 37, 'DS1' : 39,
    'EB1' : 39,
    'E1' : 41,
    'F1' : 44, 'FS1' : 46,
    'G1' : 49, 'GS1' : 52,
    'A1' : 55, 'AS1' : 58,
    'BB1' : 58,
    'B1' : 62,
    'C2' : 65, 'CS2' : 69,
    'D2' : 73, 'DS2' : 78,
    'EB2' : 78,
    'E2' : 82,
    'F2' : 87, 'FS2' : 93,
    'G2' : 98, 'GS2' : 104,
    'A2' : 110, 'AS2' : 117,
    'BB2' : 123,
    'B2' : 123,
    'C3' : 131, 'CS3' : 139,
    'D3' : 147, 'DS3' : 156,
    'EB3' : 156,
    'E3' : 165,
    'F3' : 175, 'FS3' : 185,
    'G3' : 196, 'GS3' : 208,
    'A3' : 220, 'AS3' : 233,
    'BB3' : 233,
    'B3' : 247,
    'C4' : 262, 'CS4' : 277,
    'D4' : 294, 'DS4' : 311,
    'EB4' : 311,
    'E4' : 330,
    'F4' : 349, 'FS4' : 370,
    'G4' : 392, 'GS4' : 415,
    'A4' : 440, 'AS4' : 466,
    'BB4' : 466,
    'B4' : 494,
    'C5' : 523, 'CS5' : 554,
    'D5' : 587, 'DS5' : 622,
    'EB5' : 622,
    'E5' : 659,
    'F5' : 698, 'FS5' : 740,
    'G5' : 784, 'GS5' : 831,
    'A5' : 880, 'AS5' : 932,
    'BB5' : 932,
    'B5' : 988,
    'C6' : 1047, 'CS6' : 1109,
    'D6' : 1175, 'DS6' : 1245,
    'EB6' : 1245,
    'E6' : 1319,
    'F6' : 1397, 'FS6' : 1480,
    'G6' : 1568, 'GS6' : 1661,
    'A6' : 1760, 'AS6' : 1865,
    'BB6' : 1865,
    'B6' : 1976,
    'C7' : 2093, 'CS7' : 2217,
    'D7' : 2349, 'DS7' : 2489,
    'EB7' : 2489,
    'E7' : 2637,
    'F7' : 2794, 'FS7' : 2960,
    'G7' : 3136, 'GS7' : 3322,
    'A7' : 3520, 'AS7' : 3729,
    'BB7' : 3729,
    'B7' : 3951,
    'C8' : 4186, 'CS8' : 4435,
    'D8' : 4699, 'DS8' : 4978
}

def buzz(frequency, length): #create the function "buzz" and feed it
the pitch and duration)

    if(frequency==0):
        time.sleep(length)
        return
    period = 1.0 / frequency          #frequency
    delayValue = period / 2           #calcuate the time for half of
the wave
    numCycles = int(length * frequency)    #num of waves = duratime
x freq

    for i in range(numCycles):       #start a loop from 0 to the
variable "cycles" calculated above
        buzzer_pin.on()
        time.sleep(delayValue)
```



```
           buzzer_pin.off()
           time.sleep(delayValue)


def play(melody,tempo,pause,pace=0.800):

     for i in range(0, len(melody)):       # Play song

           noteDuration = pace/tempo[i]
           buzz(melody[i],noteDuration)     # Change the frequency
along the song note

           pauseBetweenNotes = noteDuration * pause
           time.sleep(pauseBetweenNotes)

while True:
           laserturret()
           break;
```

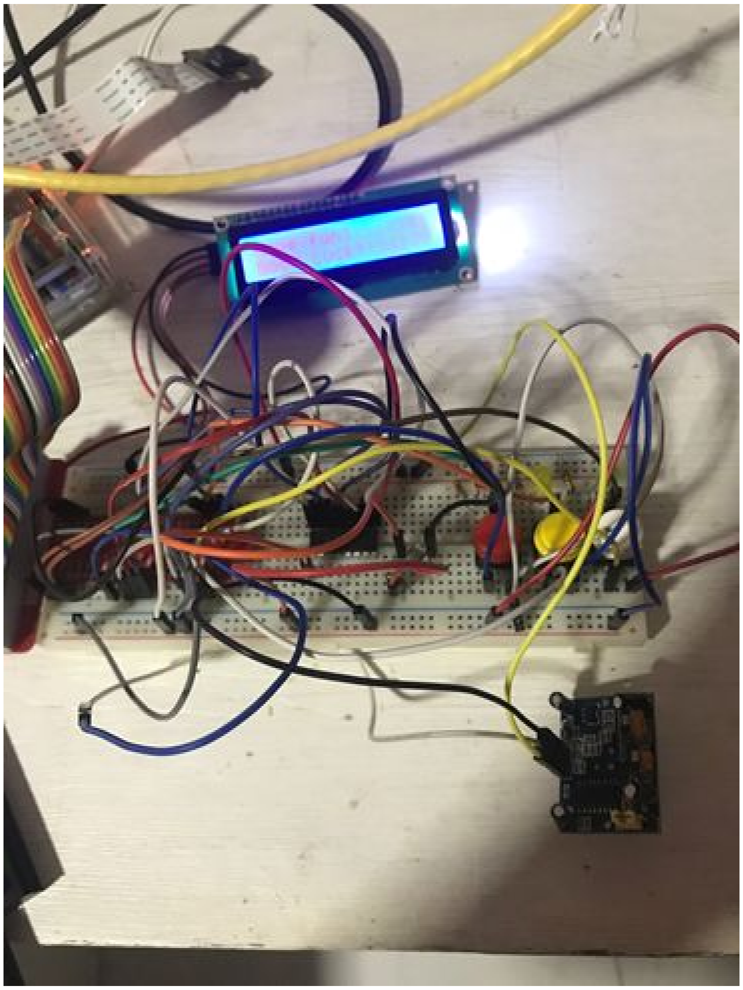
We are done with the Defence section of the Home Entertainment and Security system. We'll finish completing the other setups in the system and then proceed to creating our dashboard in Section 6.

Section 5: Entertainment

A.Hardware Setup

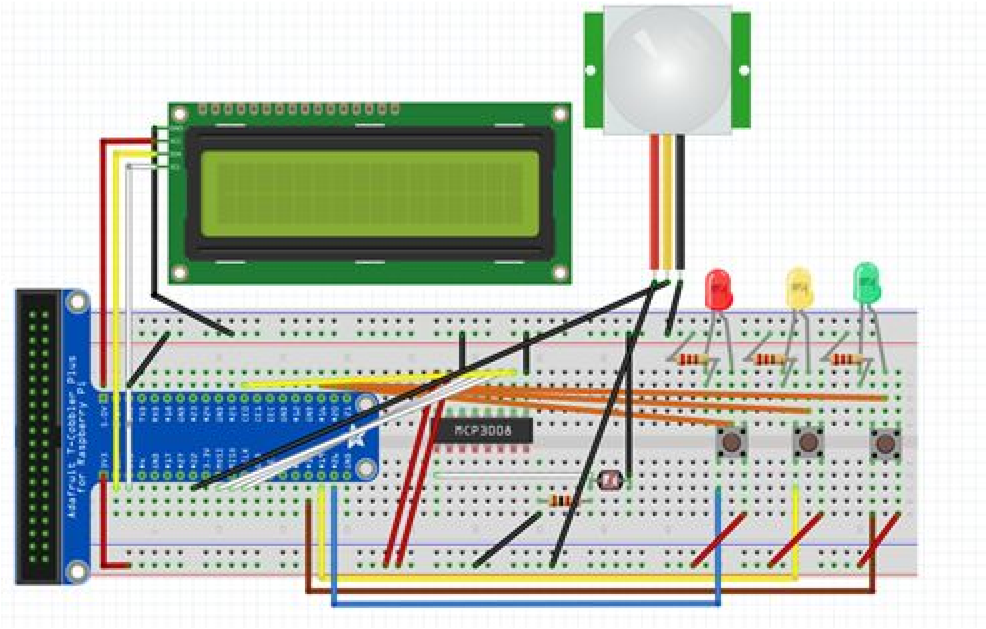
a. The RPi Connections

This is what your finished connection should look like.



b. The Fritzing Diagram

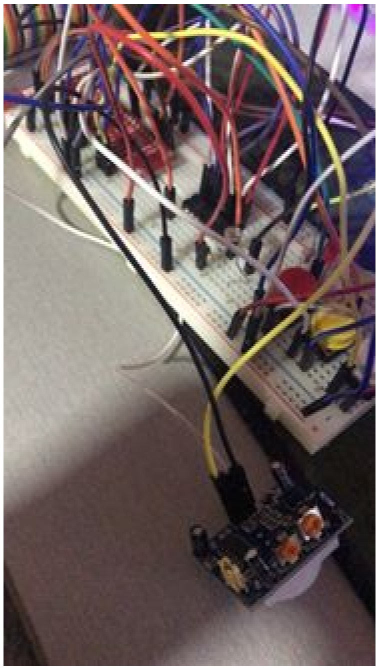
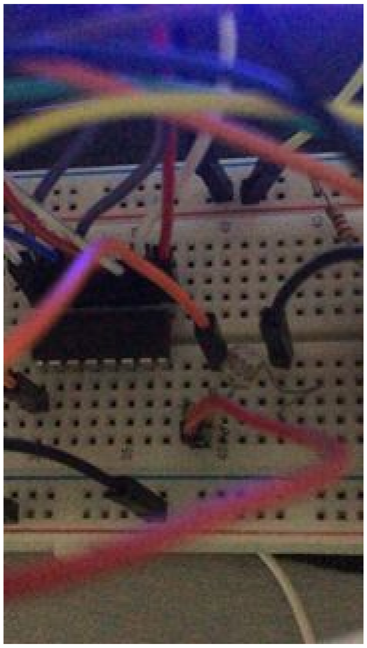
The fritzing diagram looks like this. Set up your RPi like so!

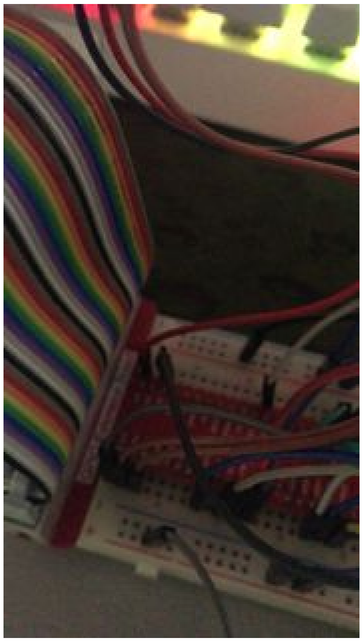


c. Connect the Components

The following steps will describe how to get the entertainment system setup.

#	Description	Image
1	Connect the three buttons and three LEDs as shown, following the fritzing diagram for a clearer view.	

2	Connect the PIR Motion Sensor to detect motion and start game.	
3	Connect the LDR alongside the MCP3008 Required connections	

4	Connect the LCD to the raspberry pi to view the status of the game, and you're done!	

The following steps will explain the main wiring. Don't forget the resistors!

Part	Cable Color	PIR Pin	RPi Pin
LCD	Orange	SCL	SCL
	Red	SDA	SDA
	Brown	GND	GND
	Black	VCC	5V
PIR Motion Sensor	Red	VCC	3V Rail
	Brown	VOUT	#22

	Black	GND	GND
LDR			
	Orange	A0	Pin 1 MCP3008
LED (R, Y, G)	Red	GPIO	#16
	Yellow	GPIO	#20
	Green	GPIO	#21
Buttons (R, Y, G)	Red	GPIO	#13
	Yellow	GPIO	#19
	Green	GPIO	#26

B.Software

A. Entertain.py

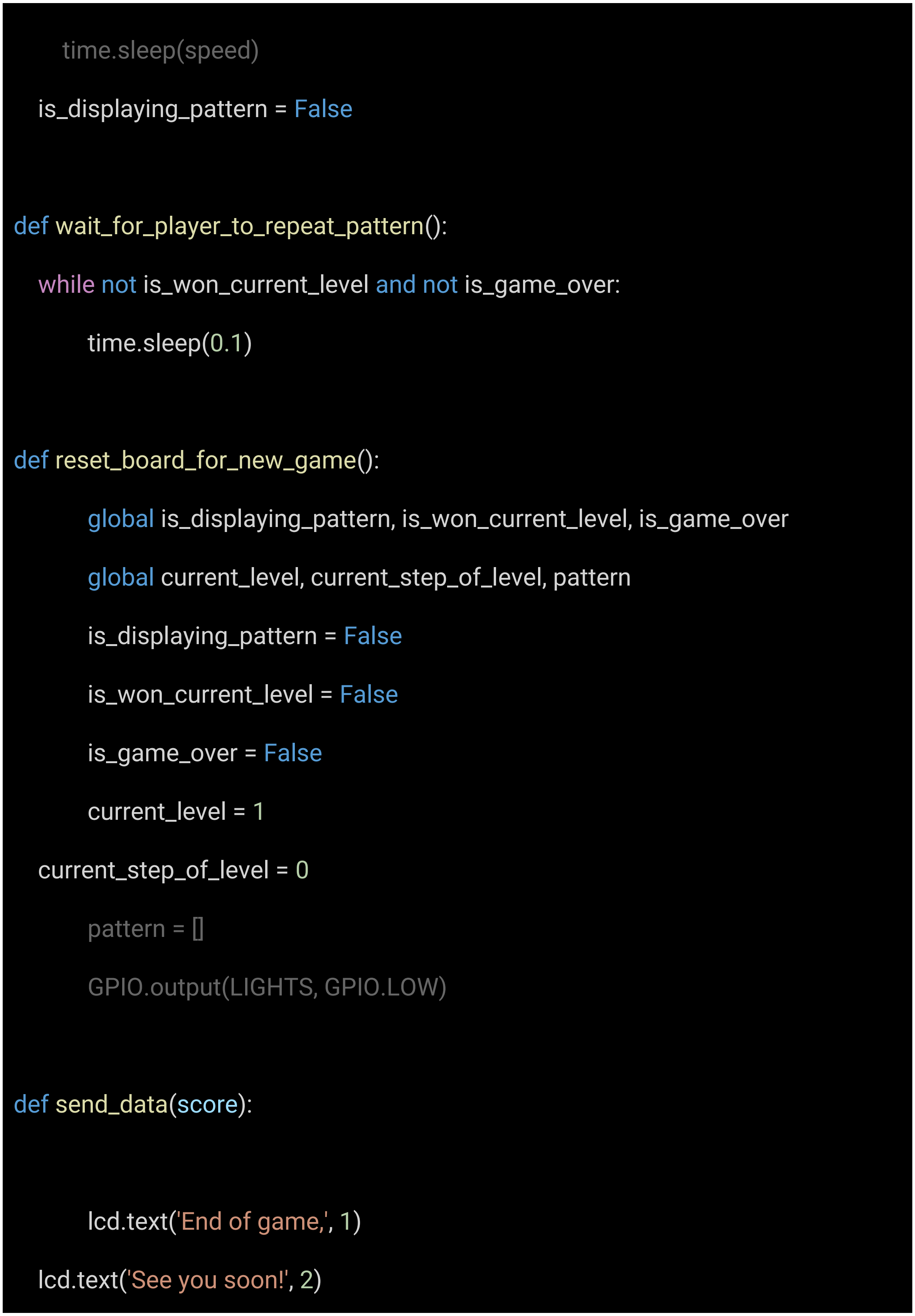
The codes below, when run, will run the simon says game, which u have to follow the pattern of the LEDS lighting up and press the corresponding buttons. It uploads scores and timestamp into the firebase nosql database for further usage in the dashboards.

```
import RPi.GPIO as GPIO

import threading

import time

import random

import os

import tweepy

from rpi_lcd import LCD

from subprocess import call

from time import sleep

from datetime import datetime

from firebase import firebase


CONSUMER_KEY = 'h5Sis7TXdoUVncrpjSzGAvhBH'

CONSUMER_SECRET =
'ZfDVxc4aTd9doGmBQO3HiSKKzxSTKT4C3g0B3AGx8eETCJm2rY'

ACCESS_KEY = '988333099669901312-YDLEQN1weW2n1JP4lxJcFPppCsbvzQh'

ACCESS_SECRET = 'K2IlUPur6jx7DO5S0HhhZW29H5AQFOvkMMevSsk9ZzwLk'


auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)

auth.secure = True

auth.set_access_token(ACCESS_KEY, ACCESS_SECRET)

api = tweepy.API(auth)
```

```
firebase = firebase.FirebaseApplication('https://iotca2-12f48.firebaseio.com', None)
lcd=LCD()
lcd.text('Have fun!', 1)
lcd.text('Good Luck!', 2)
sleep(1)

# Red, Yellow, Green
LIGHTS = [40, 38, 36]
BUTTONS = [37, 33, 35]
NOTES = ["E3", "A4", "E4"]

# values you can change that affect game play
speed = 0.5

# flags used to signal game status
is_displaying_pattern = False
is_won_current_level = False
is_game_over = False

# game state
current_level = 1
current_step_of_level = 0
```

```
pattern = []


def initialize_gpio():
    GPIO.setmode(GPIO.BOARD)
            GPIO.setup(LIGHTS, GPIO.OUT, initial=GPIO.LOW)
            GPIO.setup(BUTTONS, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
            for i in range(3):
        GPIO.add_event_detect(BUTTONS[i], GPIO.FALLING,
                              verify_player_selection)


def verify_player_selection(channel):
            global current_step_of_level, current_level, is_won_current_level,
is_game_over
            if not is_displaying_pattern and not is_won_current_level and not
is_game_over:
        flash_led_for_button(channel)
            if channel == BUTTONS[pattern[current_step_of_level]]:
            current_step_of_level += 1
            if current_step_of_level >= current_level:
                current_level += 1
                is_won_current_level = True
            else:
            is_game_over = True
```

```
def flash_led_for_button(button_channel):
        led = LIGHTS[BUTTONS.index(button_channel)]
        GPIO.output(led, GPIO.HIGH)
        time.sleep(0.4)
        GPIO.output(led, GPIO.LOW)

def add_new_color_to_pattern():
        global is_won_current_level, current_step_of_level
        is_won_current_level = False
   current_step_of_level = 0
        next_color = random.randint(0, 2)
   pattern.append(next_color)

def display_pattern_to_player():
        global is_displaying_pattern
   is_displaying_pattern = True
        GPIO.output(LIGHTS, GPIO.LOW)
        for i in range(current_level):
      GPIO.output(LIGHTS[pattern[i]], GPIO.HIGH)
      time.sleep(speed)
      GPIO.output(LIGHTS[pattern[i]], GPIO.LOW)
```



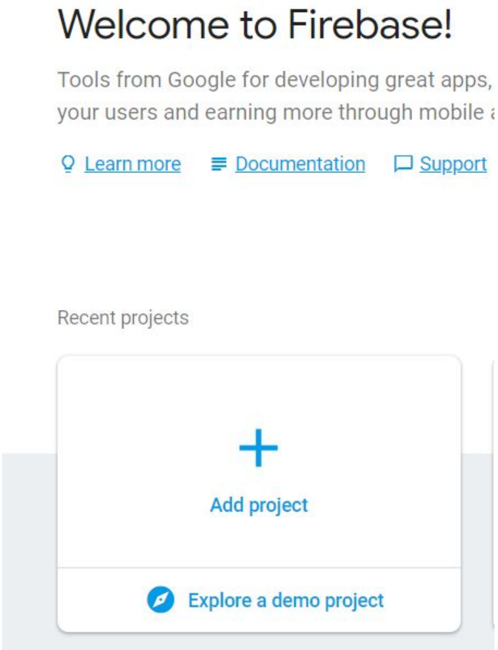
```
    time.sleep(speed)
  is_displaying_pattern = False


def wait_for_player_to_repeat_pattern():
  while not is_won_current_level and not is_game_over:
      time.sleep(0.1)


def reset_board_for_new_game():
      global is_displaying_pattern, is_won_current_level, is_game_over
      global current_level, current_step_of_level, pattern
      is_displaying_pattern = False
      is_won_current_level = False
      is_game_over = False
      current_level = 1
  current_step_of_level = 0
      pattern = []
      GPIO.output(LIGHTS, GPIO.LOW)


def send_data(score):


      lcd.text('End of game,', 1)
  lcd.text('See you soon!', 2)
```

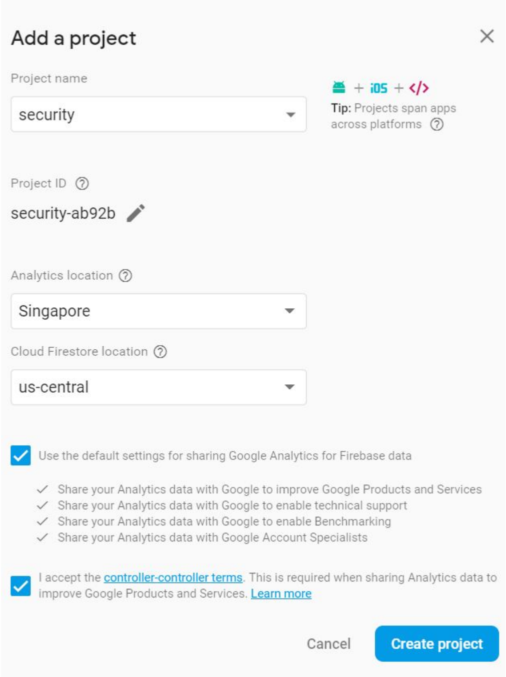
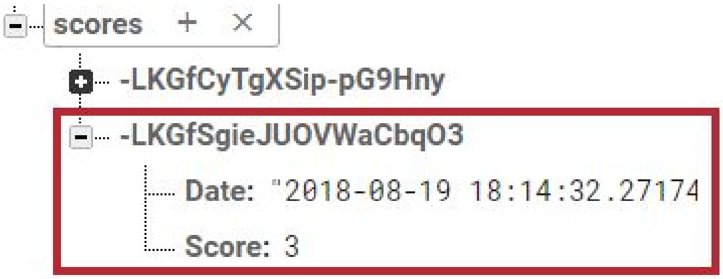
```
         datestr = str(datetime.now())

         while True:

   print(datestr)

   print(score)

   data={ 'Date': datestr,

        'Score': score

      }

   result = firebase.post('/scores/',data)

   print(result)

         if score > 2:

      status='Someone has scored '+ (str(score)) +' on '+datestr+'!'

      api.update_status (status = status)

         break


def start_game():

         while True:

      add_new_color_to_pattern()

      display_pattern_to_player()

         wait_for_player_to_repeat_pattern()

         if is_game_over:

         send_data(current_level - 1)

         print("Game Over! score is {} colors!\n".format(current_level - 1))
         sleep(2)
      print("Thanks for playing!\n")
      lcd.text('',1)
      lcd.text('',2)
         break
         time.sleep(2)

def start_game_monitor():
         t = threading.Thread(target=start_game)
         t.daemon = True
         t.start()
         t.join()

def main():
         try:
         os.system('cls' if os.name == 'nt' else 'clear')
         print("Begin new round!\n")
      initialize_gpio()
      start_game_monitor()
         finally:
         GPIO.cleanup()
```

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

We have now come to the end of the entertainment system setup guide!

B. Setup Firebase database

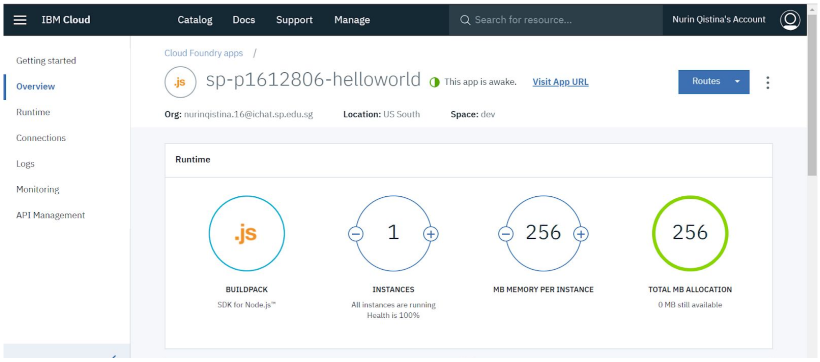
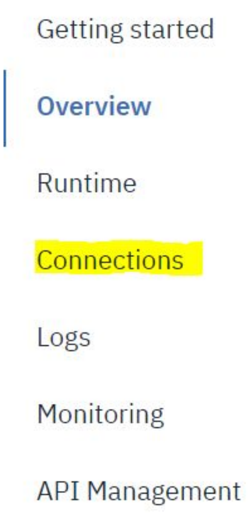
#	Description	Image
1	To set up firebase, head to https://console.firebase.google.com After Signing in, click on add project	 Welcome to Firebase! Tools from Google for developing great apps, your users and earning more through mobile a Learn more Documentation Support Recent projects Add project Explore a demo project

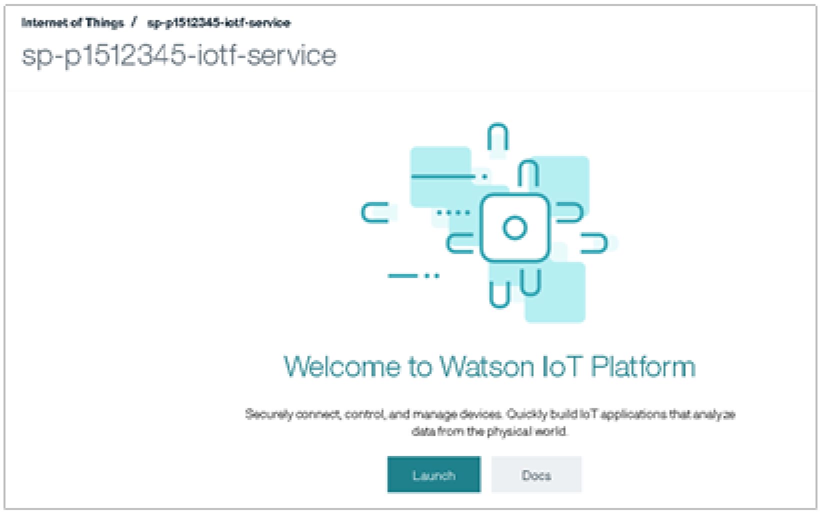
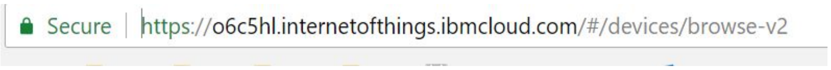
2	Enter your project name whereby mine is security	 Add a project Project name: security Tip: Projects span apps across platforms Project ID: security-ab92b Analytics location: Singapore Cloud Firestore location: us-central ☑ Use the default settings for sharing Google Analytics for Firebase data ✓ Share your Analytics data with Google to improve Google Products and Services ✓ Share your Analytics data with Google to enable technical support ✓ Share your Analytics data with Google to enable Benchmarking ✓ Share your Analytics data with Google Account Specialists ☑ I accept the controller-controller terms. This is required when sharing Analytics data to improve Google Products and Services. Learn more Cancel Create project
3	Now, when u run the entertainment system, it will log into the firebase realtime database as such, whereby date and score will be stored per game.	 scores -LKGfCyTgXSip-pG9Hny -LKGfSgieJUOVWaCbqO3 Date: "2018-08-19 18:14:32.27174 Score: 3

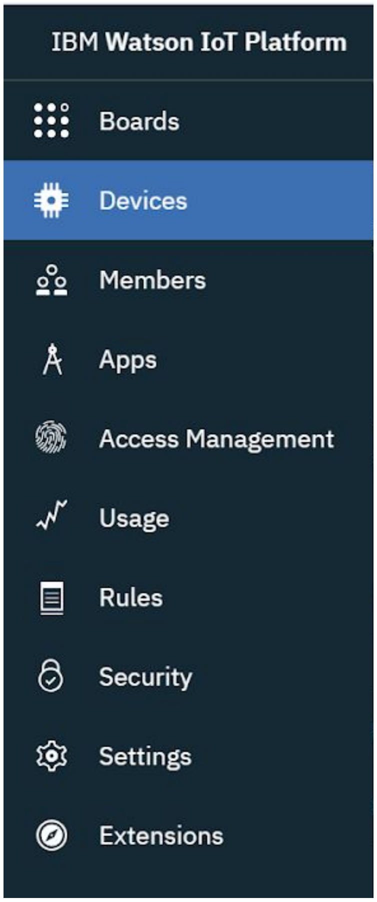
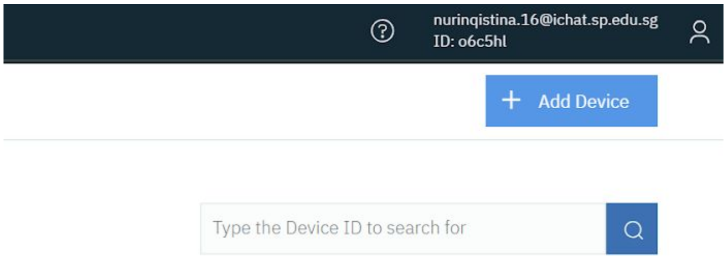
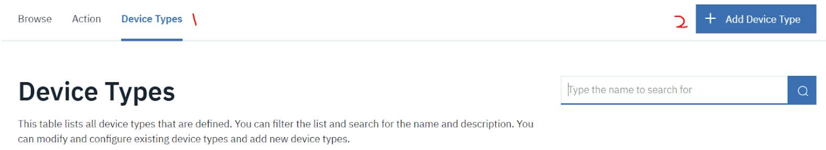
Section 6 : IOT App Watson on IBM Bluemix

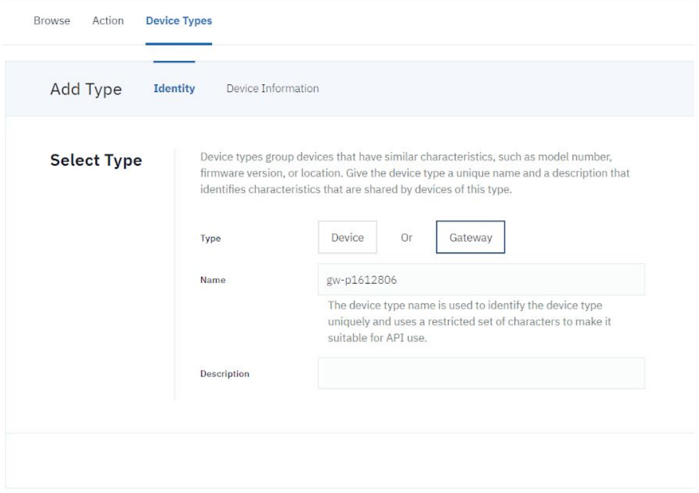
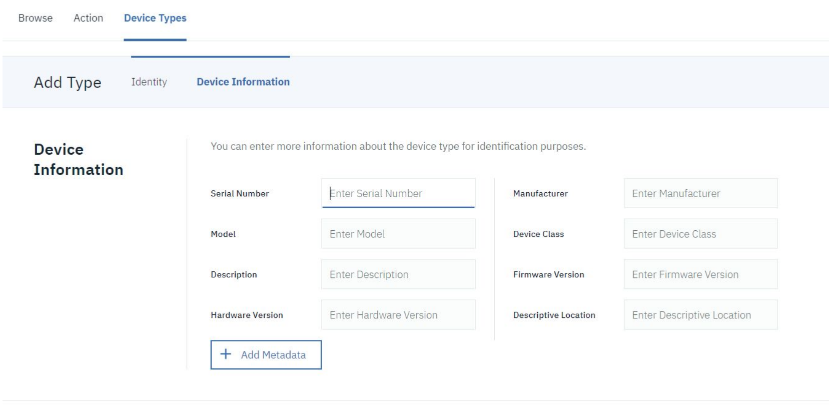
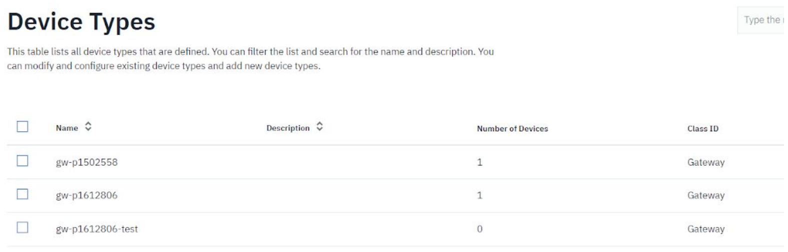
Set up Bluemix IoT Service

a. Set up a Gateway Device Type

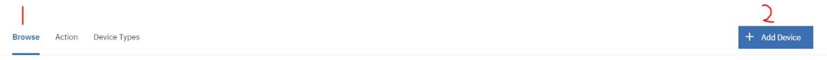
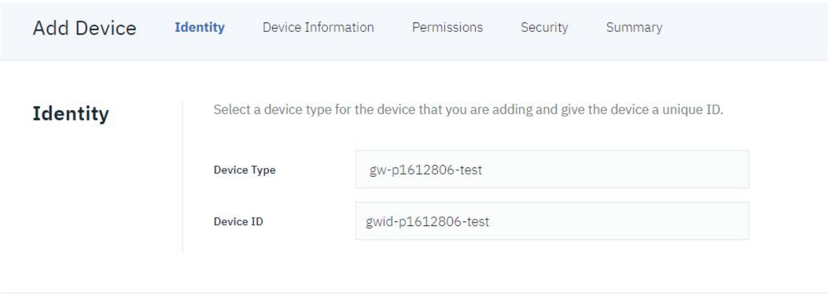
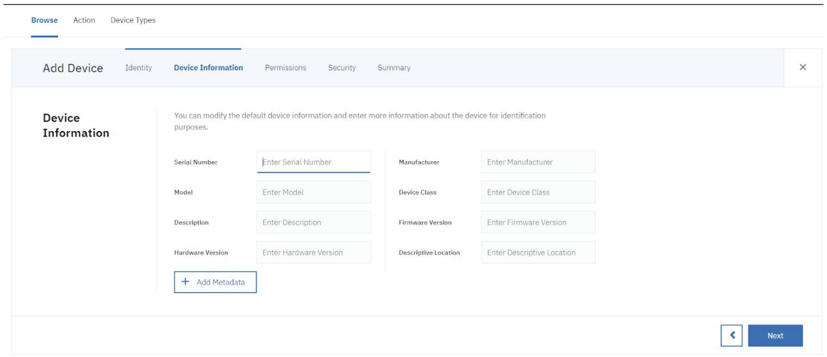
#	Description	Image
1	Open your cloud foundry app and make sure it is running.	 IBM Cloud Catalog Docs Support Manage Search for resource... Nurin Qistina's Account Getting started Overview Runtime Connections Logs Monitoring API Management Cloud Foundry apps / .js sp-p1612806-helloworld This app is awake. Visit App URL Routes Runtime BUILDPACK SDK for Node.js™ 1 INSTANCES All instances are running Health is 100% 256 MB MEMORY PER INSTANCE 256 TOTAL MB ALLOCATION 0 MB still available Org: nurinqistina.16@ichat.sp.edu.sg Location: US South Space: dev
2	Click on the connections tab.	 Getting started Overview Runtime Connections Logs Monitoring API Management

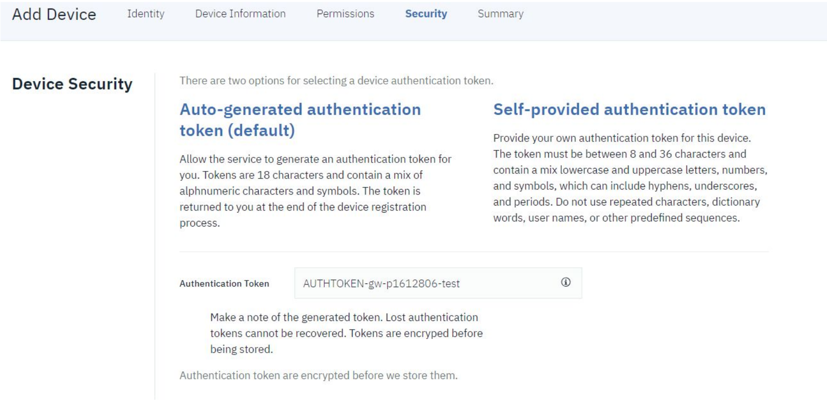
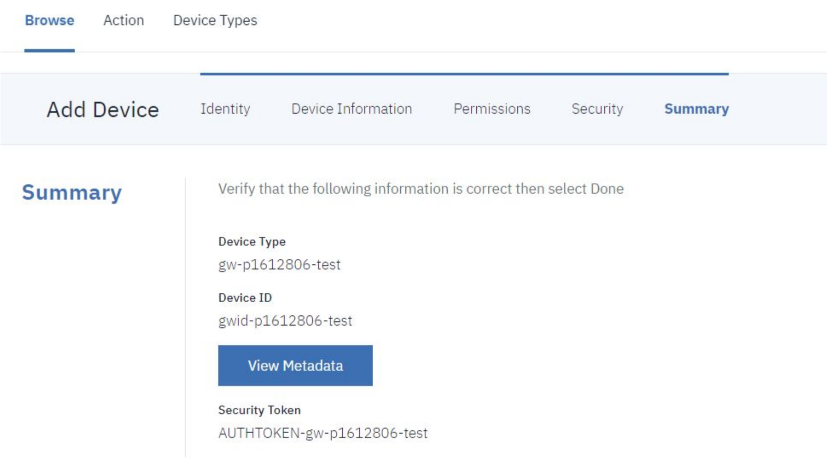
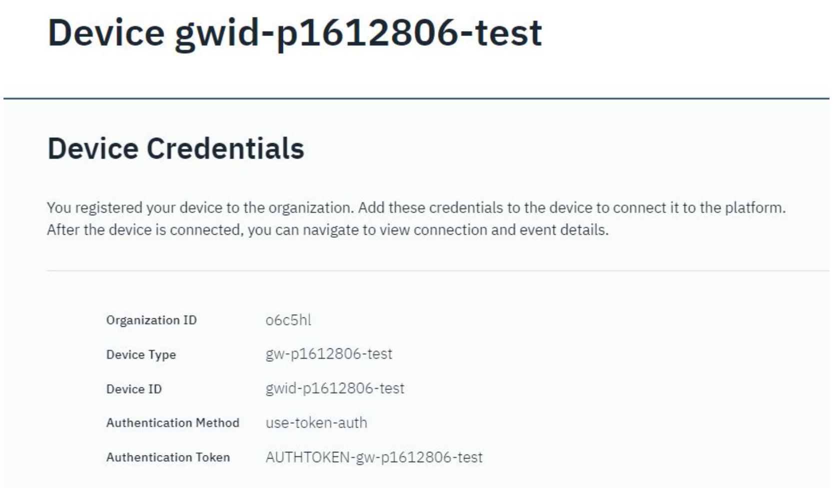
3	You will be brought to the IoT service configuration as shown. Click on the “Launch” button.	 Internet of Things / sp-p1512345-iotf-service sp-p1512345-iotf-service Welcome to Watson IoT Platform Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.
4	Note the URL you’ve been redirected to. In our case, it is: *https://o6c5hl.internetofthings.ibmcloud.com*	 Secure | https://o6c5hl.internetofthings.ibmcloud.com/#/devices/browse-v2

5	Hover over the side bar and click Devices	 IBM Watson IoT Platform Boards Devices Members Apps Access Management Usage Rules Security Settings Extensions
6	Click on add devices on the top right.	 nurinqistina.16@ichat.sp.edu.sg ID: o6c5hl
7	On the top bar, you'll see as shown. Click "Device Types" then click "Add Device Type"	 Browse Action Device Types 1 2 Add Device Type Type the name to search for Device Types This table lists all device types that are defined. You can filter the list and search for the name and description. You can modify and configure existing device types and add new device types.

8	Select Gateway and create a recognizable name for the Pi you are using. Eg. gw-p1612806	 Browse Action Device Types Add Type Identity Device Information **Select Type** Device types group devices that have similar characteristics, such as model number, firmware version, or location. Give the device type a unique name and a description that identifies characteristics that are shared by devices of this type. Type: Device Or [✓ Gateway] Name: gw-p1612806 The device type name is used to identify the device type uniquely and uses a restricted set of characters to make it suitable for API use. Description:
9	Skip this and click done.	 Browse Action Device Types Add Type Identity Device Information **Device Information** You can enter more information about the device type for identification purposes. Serial Number: Enter Serial Number Manufacturer: Enter Manufacturer Model: Enter Model Device Class: Enter Device Class Description: Enter Description Firmware Version: Enter Firmware Version Hardware Version: Enter Hardware Version Descriptive Location: Enter Descriptive Location
10	If successful, you'll be able to see the gateway you've created. Create a gateway for each Pi that you'll be using. For example, we have 2 pi currently for our system. Note: gw-1612806-test is for demo.	 **Device Types** This table lists all device types that are defined. You can filter the list and search for the name and description. You can modify and configure existing device types and add new device types. □ Name / Description / Number of Devices / Class ID □ gw-p1502558 / / 1 / Gateway □ gw-p1612806 / / 1 / Gateway □ gw-p1612806-test / / 0 / Gateway

b. Set up the Pi as a Gateway

#	Description	Image
1	Continuing from the previous section, we'll add a new device.	
2	Create an identity for the device. Device type will be the type we created previously (gw-p1612806-test). We name our Device ID as "gwid-p1612806-test" Click next.	Add Device · Identity · Device Information · Permissions · Security · Summary **Identity** Select a device type for the device that you are adding and give the device a unique ID. Device Type: gw-p1612806-test Device ID: gwid-p1612806-test
3	Skip Device Information and Permissions. Click Security.	Browse · Action · Device Types Add Device · Identity · Device Information · Permissions · Security · Summary **Device Information** You can modify the default device information and enter more information about the device for identification purposes. Serial Number: Enter Serial Number Manufacturer: Enter Manufacturer Model: Enter Model Device Class: Enter Device Class Description: Enter Description Firmware Version: Enter Firmware Version Hardware Version: Enter Hardware Version Descriptive Location: Enter Descriptive Location

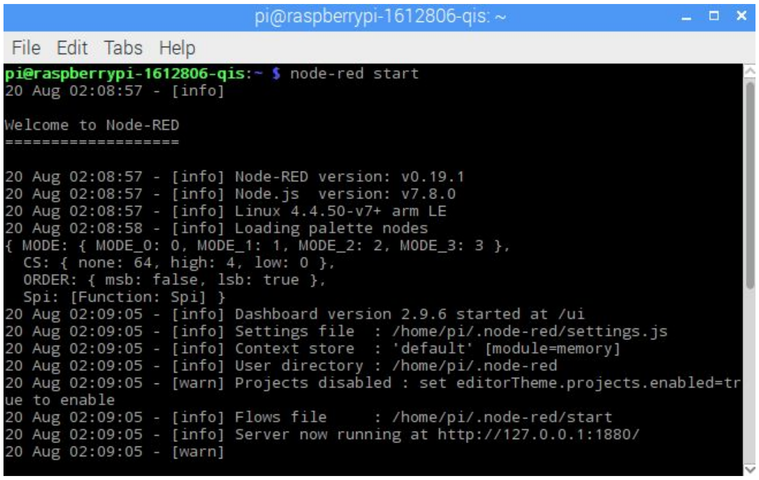
4	On the Security tab, you'll see this. We create our Auth Token and name it "AUTHTOKEN-gw-p1612806-test" Once you're done, Click next.	 Add Device Identity Device Information Permissions Security Summary **Device Security** There are two options for selecting a device authentication token. **Auto-generated authentication token (default)** Allow the service to generate an authentication token for you. Tokens are 18 characters and contain a mix of alphnumeric characters and symbols. The token is returned to you at the end of the device registration process. **Self-provided authentication token** Provide your own authentication token for this device. The token must be between 8 and 36 characters and contain a mix lowercase and uppercase letters, numbers, and symbols, which can include hyphens, underscores, and periods. Do not use repeated characters, dictionary words, user names, or other predefined sequences. Authentication Token: AUTHTOKEN-gw-p1612806-test Make a note of the generated token. Lost authentication tokens cannot be recovered. Tokens are encryped before being stored. Authentication token are encrypted before we store them.
5	Ensure you've types everything correctly and click done.	 Browse Action Device Types Add Device Identity Device Information Permissions Security Summary **Summary** Verify that the following information is correct then select Done Device Type gw-p1612806-test Device ID gwid-p1612806-test Security Token AUTHTOKEN-gw-p1612806-test
6	Wait a few minutes for this page to load. This is important! Take down this information and save it.	 **Device gwid-p1612806-test** **Device Credentials** You registered your device to the organization. Add these credentials to the device to connect it to the platform. After the device is connected, you can navigate to view connection and event details. Organization ID: o6c5hl Device Type: gw-p1612806-test Device ID: gwid-p1612806-test Authentication Method: use-token-auth Authentication Token: AUTHTOKEN-gw-p1612806-test

Set up Node-Red

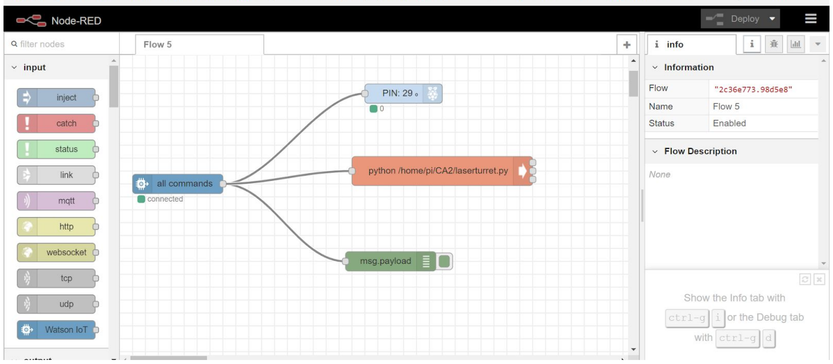
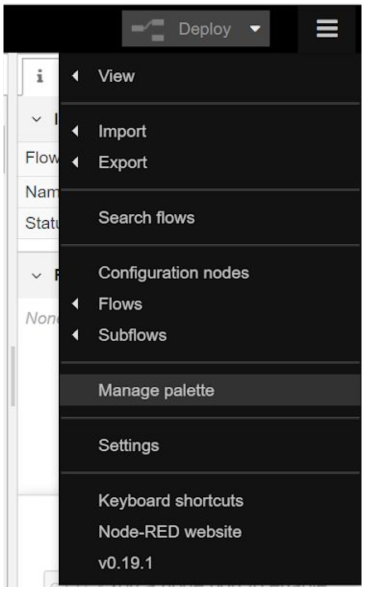
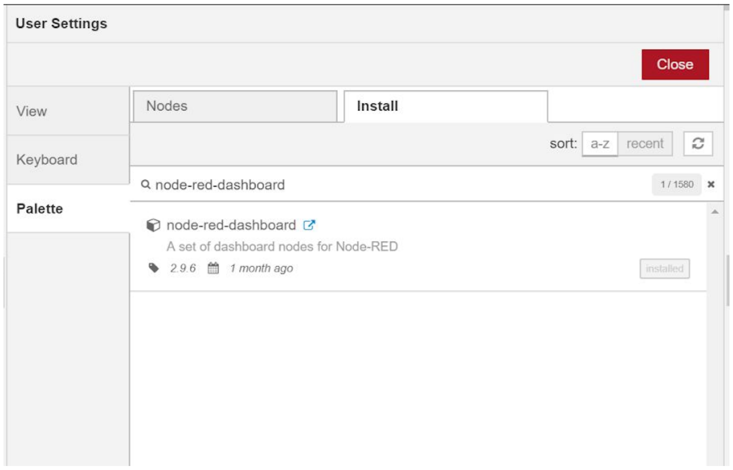
c. Install IBM Watson Node-RED nodes

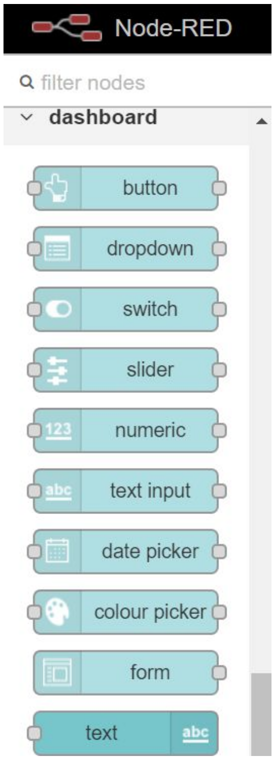
#	Description	Image
1	Open a terminal window and install it in on your Rpi.	sudo npm i -g node-red-contrib-ibm-watson-iot
2	Once succesful, reboot your RPi.	Sudo reboot now

d. Install Node-Red-Dashboard Nodes

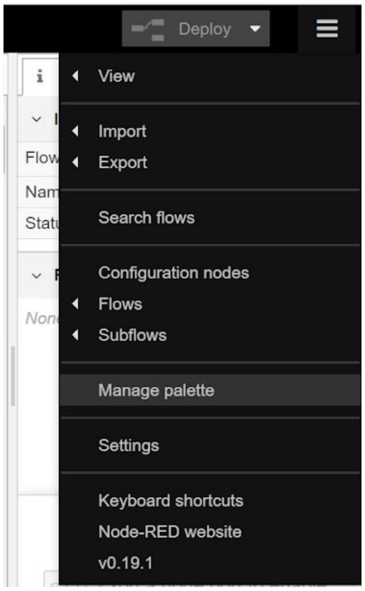
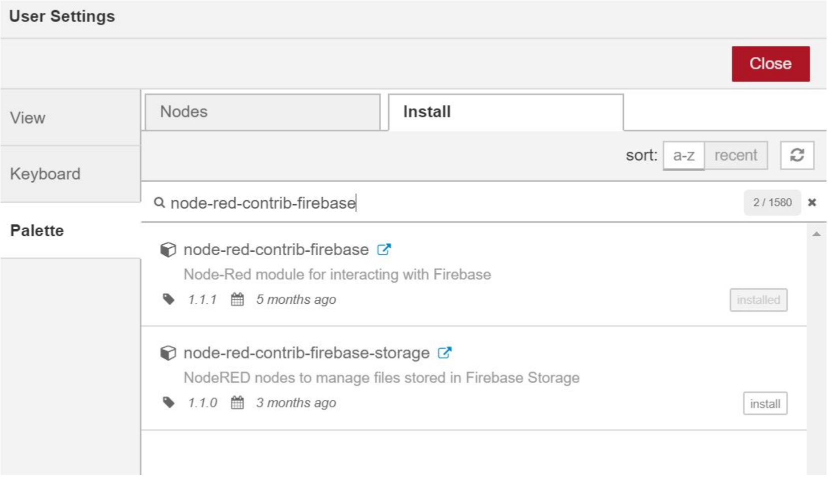
#	Description	Image
1	Open a terminal window and run Node-red.	

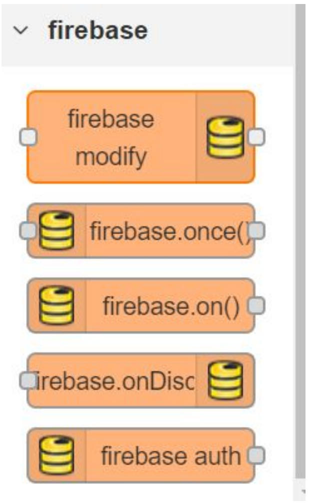
```
pi@raspberrypi-1612806-qis:~ $ node-red start
20 Aug 02:08:57 - [info]

Welcome to Node-RED
===================

20 Aug 02:08:57 - [info] Node-RED version: v0.19.1
20 Aug 02:08:57 - [info] Node.js  version: v7.8.0
20 Aug 02:08:57 - [info] Linux 4.4.50-v7+ arm LE
20 Aug 02:08:58 - [info] Loading palette nodes
{ MODE: { MODE_0: 0, MODE_1: 1, MODE_2: 2, MODE_3: 3 },
  CS: { none: 64, high: 4, low: 0 },
  ORDER: { msb: false, lsb: true },
  Spi: [Function: Spi] }
20 Aug 02:09:05 - [info] Dashboard version 2.9.6 started at /ui
20 Aug 02:09:05 - [info] Settings file  : /home/pi/.node-red/settings.js
20 Aug 02:09:05 - [info] Context store  : 'default' [module=memory]
20 Aug 02:09:05 - [info] User directory : /home/pi/.node-red
20 Aug 02:09:05 - [warn] Projects disabled : set editorTheme.projects.enabled=tr
ue to enable
20 Aug 02:09:05 - [info] Flows file     : /home/pi/.node-red/start
20 Aug 02:09:05 - [info] Server now running at http://127.0.0.1:1880/
20 Aug 02:09:05 - [warn]
```

2	Open the node-red flow editor by going to your RPi ip address followed by the :1880 extension For example, mine is http://192.168.0.117:1880	 Node-RED Deploy filter nodes input inject catch status link mqtt http websocket tcp udp Watson IoT Flow 5 PIN: 29 0 all commands connected python /home/pi/CA2/laserturret.py msg.payload info Information Flow "2c36e773.98d5e8" Name Flow 5 Status Enabled Flow Description None Show the Info tab with ctrl-g i or the Debug tab with ctrl-g d
3	Click the hamburger icon on the top right followed by Manage Palette	 Deploy View Import Export Search flows Configuration nodes Flows Subflows Manage palette Settings Keyboard shortcuts Node-RED website v0.19.1
4	Click the install tab and search for “node-red-dashboard” If it’s not installed yet, go ahead and install it. As you can see, I have already installed it.	 User Settings Close View Keyboard Palette Nodes Install sort: a-z recent node-red-dashboard 1 / 1580 node-red-dashboard A set of dashboard nodes for Node-RED 2.9.6 1 month ago installed

5	Once installed, you should see it on the side in the list of the available nodes.	 Node-RED filter nodes dashboard button dropdown switch slider numeric text input date picker colour picker form text

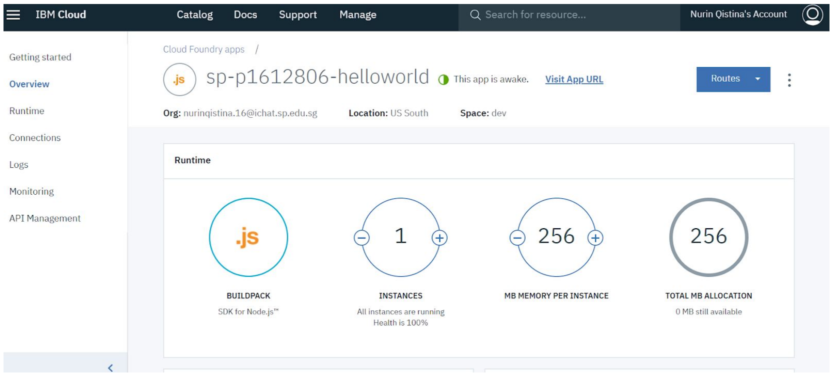
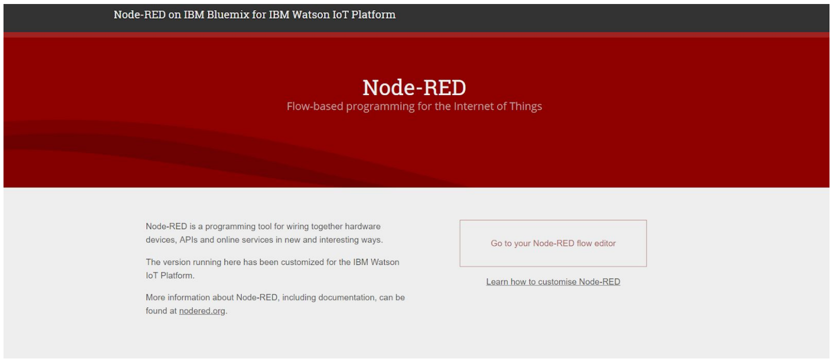
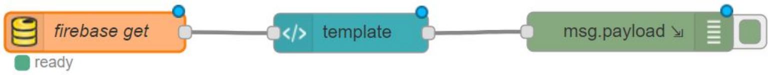
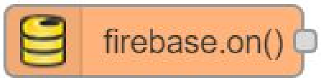
e. Install Firebase Nodes

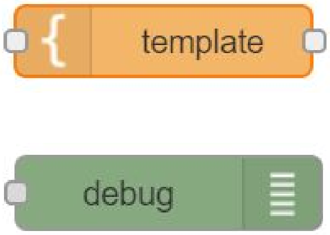
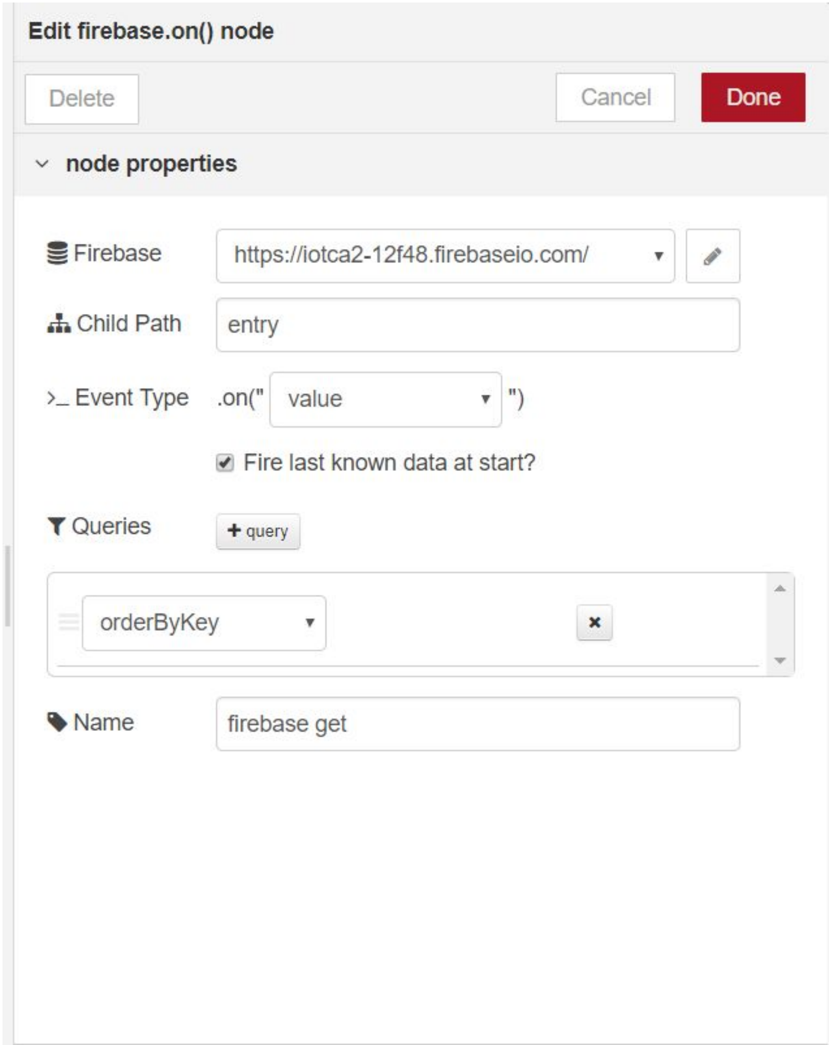
#	Description	Image
1	Click the hamburger icon on the top right followed by Manage Palette	 Deploy View Import Export Search flows Configuration nodes Flows Subflows Manage palette Settings Keyboard shortcuts Node-RED website v0.19.1
2	Click the install tab and search for “node-red-contrib-firebase” If it’s not installed yet, go ahead and install it. As you can see, I have already installed it.	 User Settings Close View Keyboard Palette Nodes Install sort: a-z recent 2 / 1580 installed install node-red-contrib-firebase node-red-contrib-firebase Node-Red module for interacting with Firebase 1.1.1 5 months ago node-red-contrib-firebase-storage NodeRED nodes to manage files stored in Firebase Storage 1.1.0 3 months ago

3	Once installed, you should see it on the side in the list of the available nodes.	 firebase firebase modify firebase.once() firebase.on() irebase.onDisc firebase auth

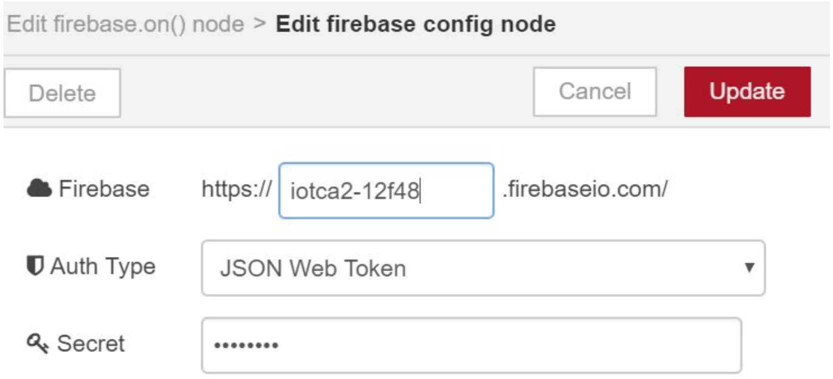
Security

A. Create Scoreboard Node-RED Flow on Bluemix

#	Description	Image
1	On the IBM Bluemix Console. On our App, you'll see this page. Click "Visit App URL"	 IBM Cloud Catalog Docs Support Manage Search for resource... Nurin Qistina's Account Getting started Overview Runtime Connections Logs Monitoring API Management Cloud Foundry apps / Visit App URL Routes Runtime .js 1 256 256 BUILDPACK INSTANCES MB MEMORY PER INSTANCE TOTAL MB ALLOCATION sp-p1612806-helloworld This app is awake. **Org:** nurinqistina.16@ichat.sp.edu.sg **Location:** US South **Space:** dev SDK for Node.js™ All instances are running Health is 100% 0 MB still available
2	You'll see this page. Click go to your Node-RED Flow Editor.	 Node-RED on IBM Bluemix for IBM Watson IoT Platform Node-RED Flow-based programming for the Internet of Things Go to your Node-RED flow editor Learn how to customise Node-RED Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. The version running here has been customized for the IBM Watson IoT Platform. More information about Node-RED, including documentation, can be found at nodered.org.
3	This is the flow that we will be creating show who has attempted to enter the house at different timestamps	
4	You will need the following nodes. 1. Firebase.on 2. Template	

	3. Debug	
5	Double click the firebase.on node. Change the child path to whatever you have set in the code, mine for example is entry	 Edit firebase.on() node node properties Firebase https://iotca2-12f48.firebaseio.com/ Child Path entry Event Type .on(" value ") ☑ Fire last known data at start? Queries orderByKey Name firebase get

6 You will be prompted to config the firebase auth, enter the secret key which can be retrieved from the firebase console



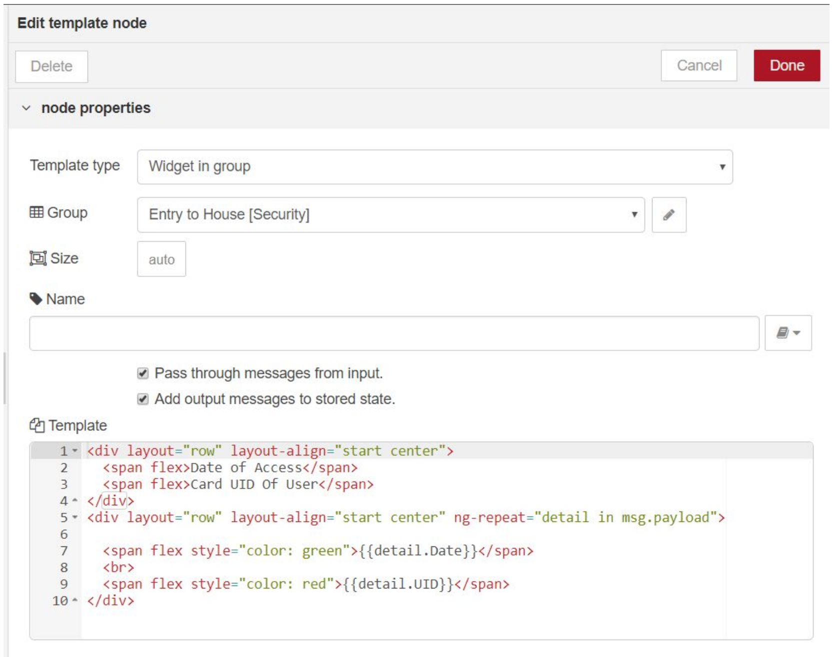
Firebase https:// iotca2-12f48 .firebaseio.com/

Auth Type: JSON Web Token

Secret: ••••••••

7 Double click the template node.

Configure it as such.



Template type: Widget in group

Group: Entry to House [Security]

Size: auto

Name:

☑ Pass through messages from input.

☑ Add output messages to stored state.

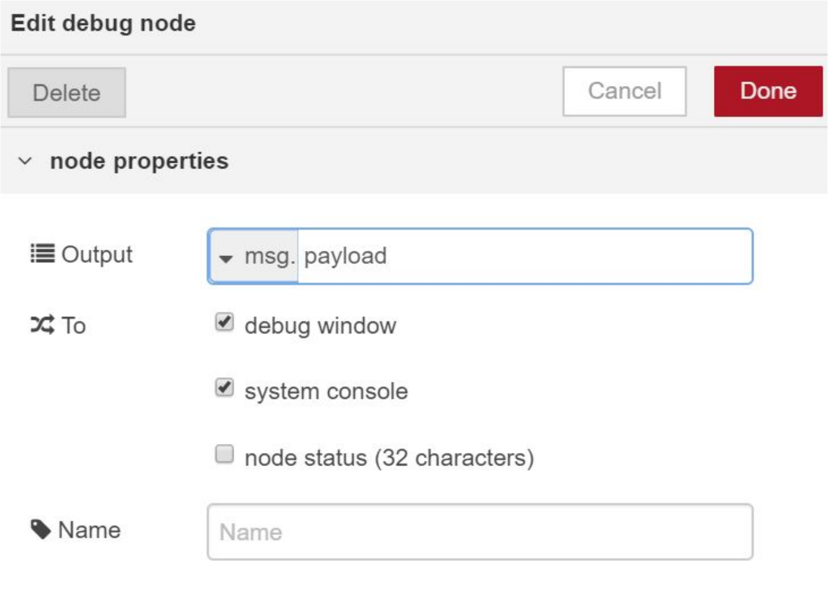
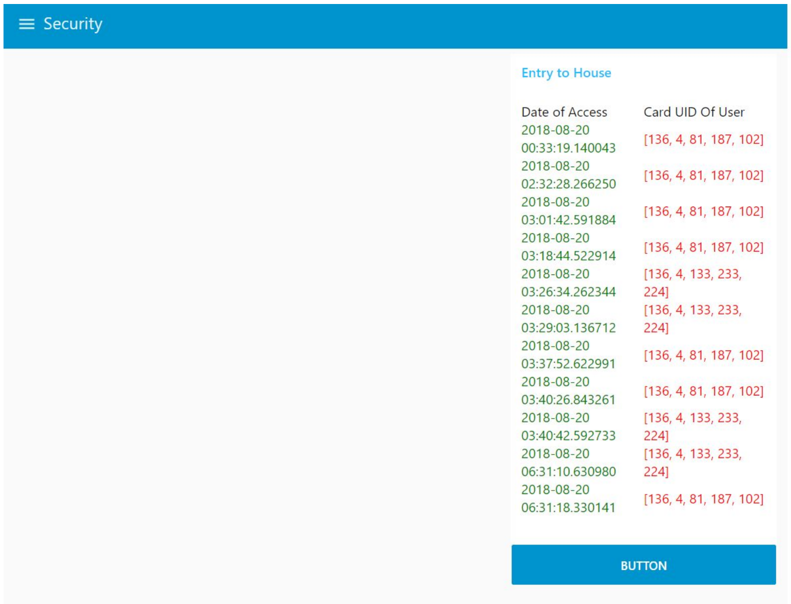
Template

```
<div layout="row" layout-align="start center">
  <span flex>Date of Access</span>
  <span flex>Card UID Of User</span>
</div>
<div layout="row" layout-align="start center" ng-repeat="detail in msg.payload">

  <span flex style="color: green">{{detail.Date}}</span>
  <br>
  <span flex style="color: red">{{detail.UID}}</span>
</div>
```

8 In the code section, we'll paste the following codes;

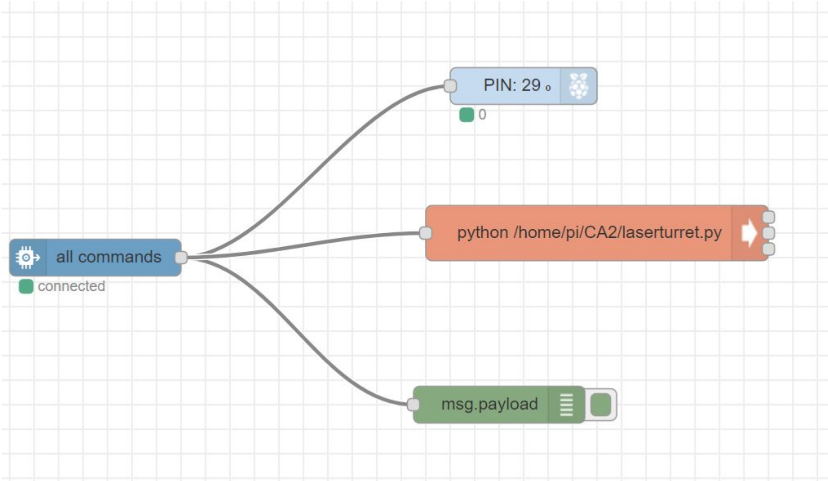
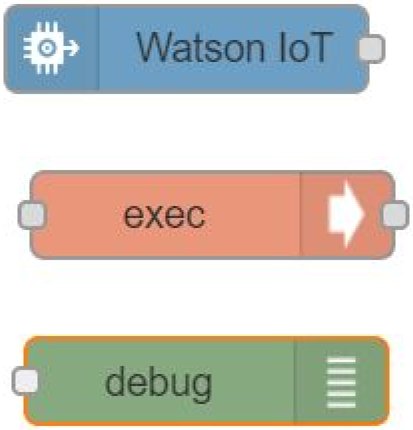
```
<div layout="row" layout-align="start center">
  <span flex>Date of Access</span>
  <span flex>Card UID Of User</span>
</div>
<div layout="row" layout-align="start center"
ng-repeat="detail in msg.payload">

  <span flex style="color:
green">{{detail.Date}}</span>
  <br>
  <span flex style="color:
```

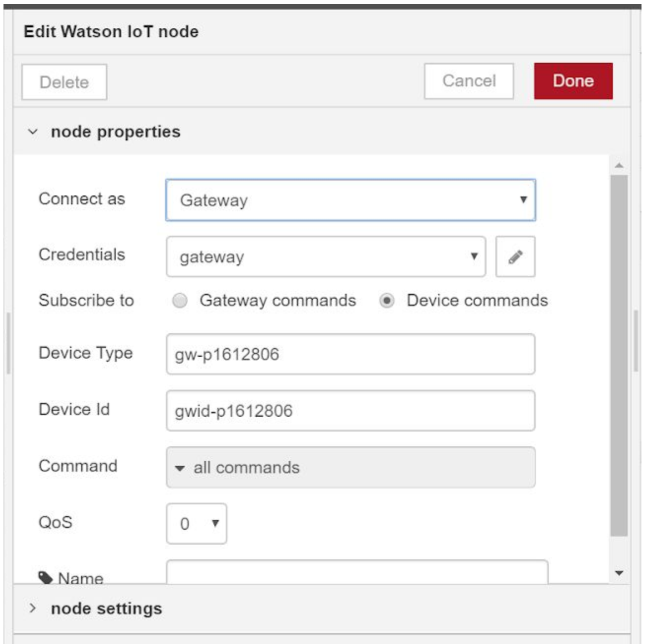
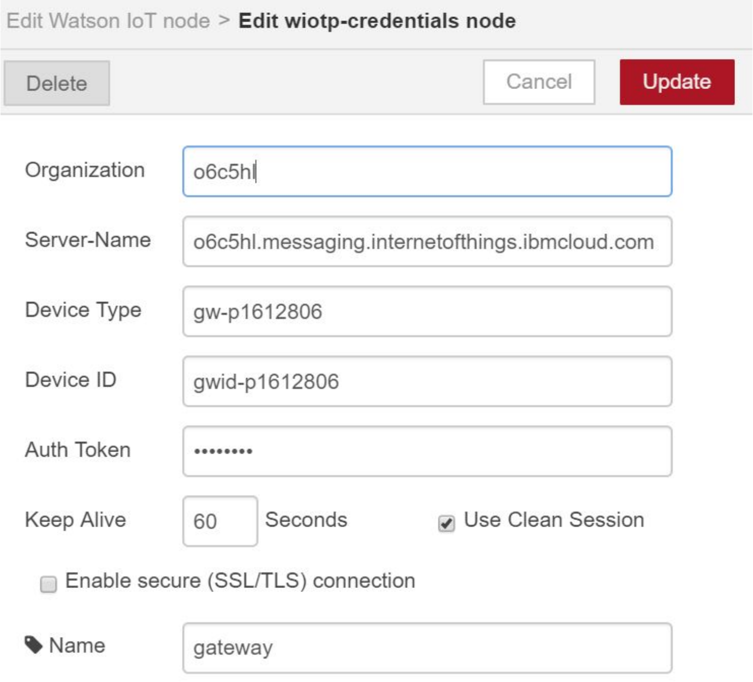
<table>
<tr>
<td></td>
<td></td>
<td>red">{{detail.UID}}</span>
</div></td>
</tr>
<tr>
<td>9</td>
<td>In the debug code, configure it as such.

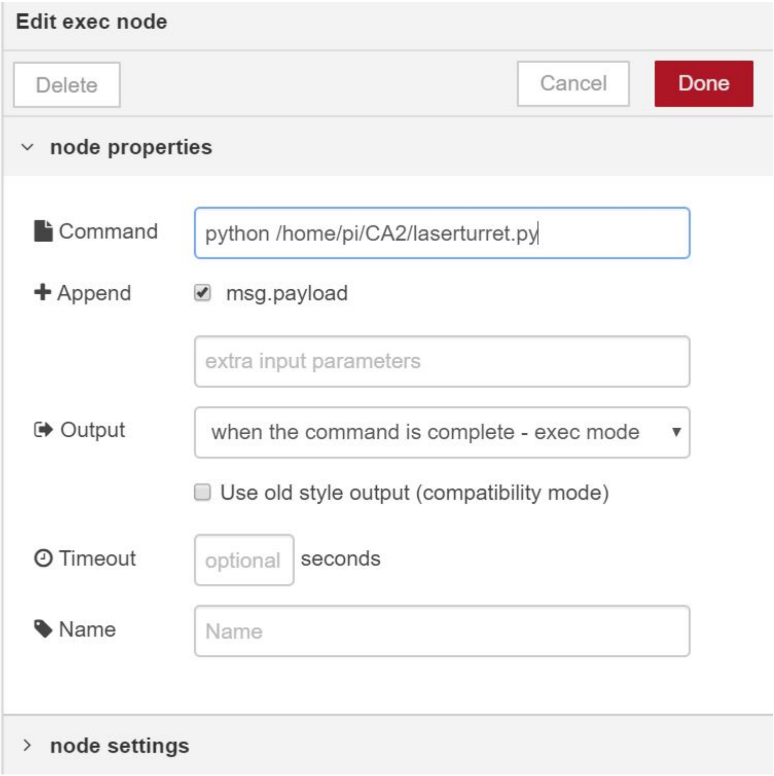
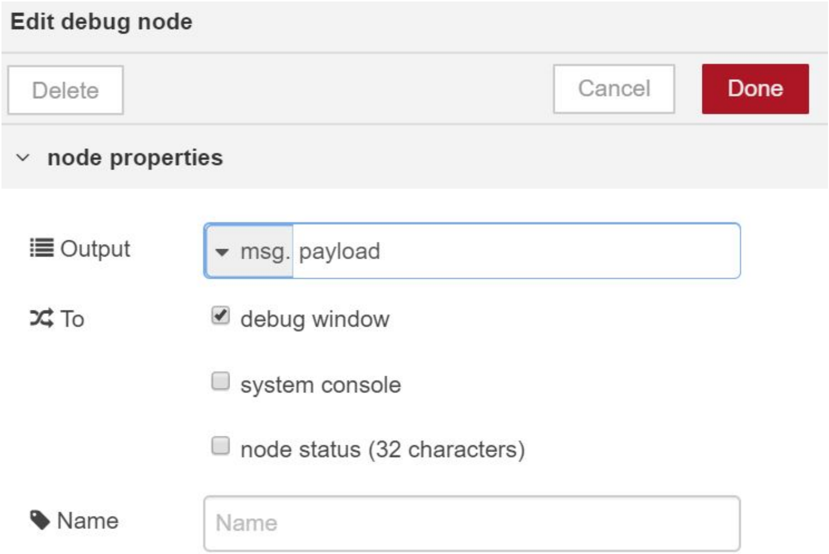
This is just for our reference in case we need it later on.</td>
<td>
<b>Edit debug node</b>
Delete Cancel Done
node properties
Output: msg. payload
To: ☑ debug window
☑ system console
□ node status (32 characters)
Name: Name</td>
</tr>
<tr>
<td>10</td>
<td>You can then view who has entered/attempeted to enter your house at what time on the dashboard</td>
<td>
Security
Entry to House
<table>
<tr><th>Date of Access</th><th>Card UID Of User</th></tr>
<tr><td>2018-08-20 00:33:19.140043</td><td>[136, 4, 81, 187, 102]</td></tr>
<tr><td>2018-08-20 02:32:28.266250</td><td>[136, 4, 81, 187, 102]</td></tr>
<tr><td>2018-08-20 03:01:42.591884</td><td>[136, 4, 81, 187, 102]</td></tr>
<tr><td>2018-08-20 03:18:44.522914</td><td>[136, 4, 81, 187, 102]</td></tr>
<tr><td>2018-08-20 03:26:34.262344</td><td>[136, 4, 133, 233, 224]</td></tr>
<tr><td>2018-08-20 03:29:03.136712</td><td>[136, 4, 133, 233, 224]</td></tr>
<tr><td>2018-08-20 03:37:52.622991</td><td>[136, 4, 81, 187, 102]</td></tr>
<tr><td>2018-08-20 03:40:26.843261</td><td>[136, 4, 81, 187, 102]</td></tr>
<tr><td>2018-08-20 03:40:42.592733</td><td>[136, 4, 133, 233, 224]</td></tr>
<tr><td>2018-08-20 06:31:10.630980</td><td>[136, 4, 133, 233, 224]</td></tr>
<tr><td>2018-08-20 06:31:18.330141</td><td>[136, 4, 81, 187, 102]</td></tr>
</table>
BUTTON</td>
</tr>
</table>

Defence

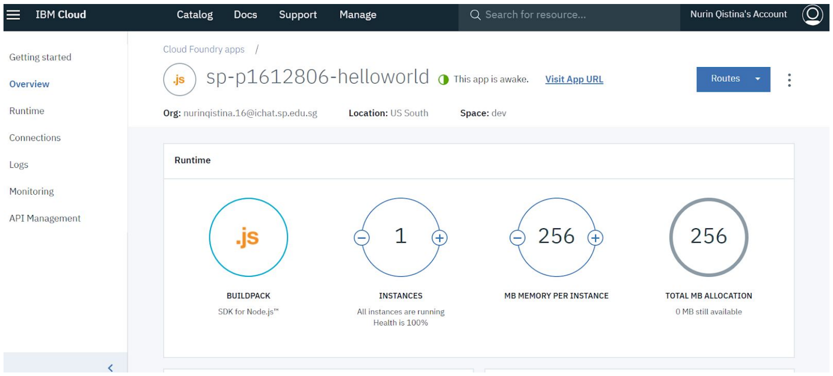
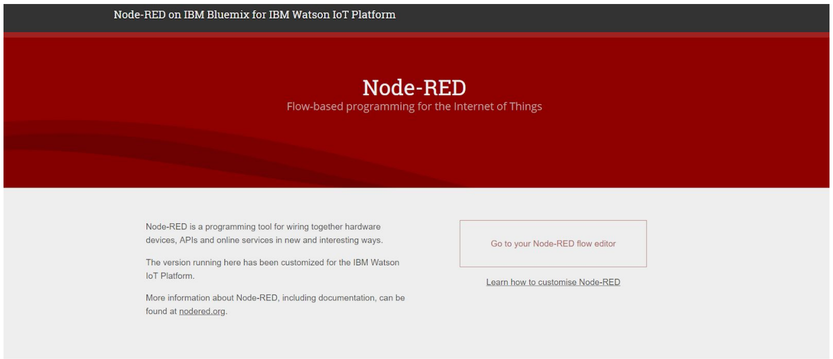
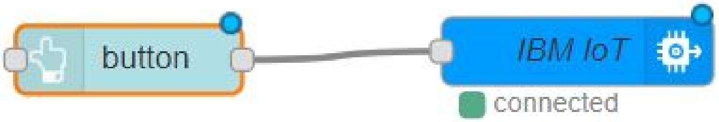
A. Create Node-RED Flow on RPi

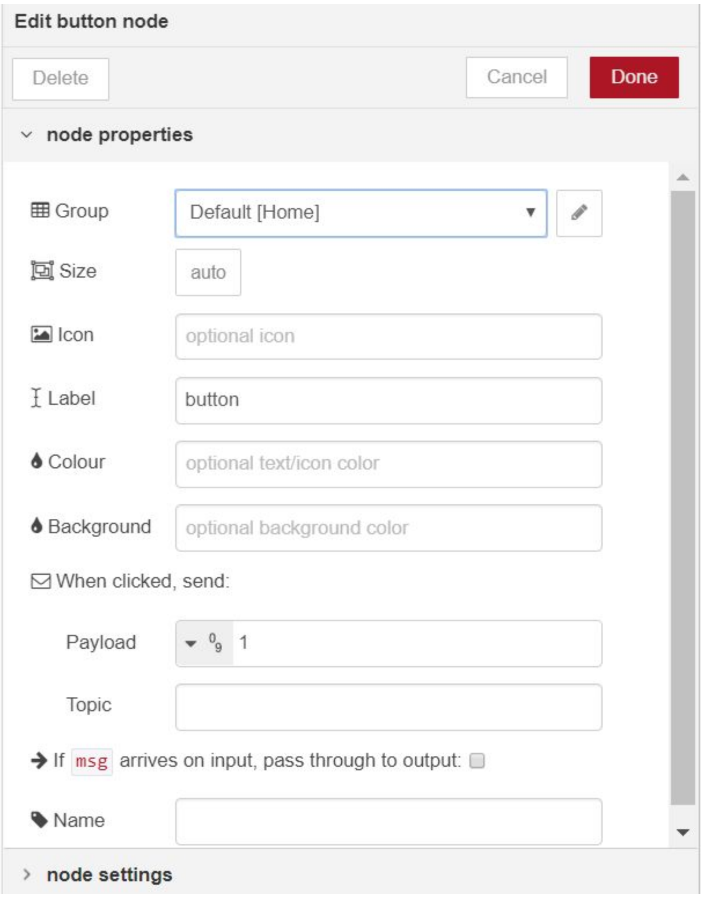
#	Description	Image
1	We are going to create a flow on your Defence RPi. This is a look on the completed flow. How this works is: When there is an input from the watson bluemix IOT end, it will run laserturret.py. In the later sections, we'll create a dashboard where you can interact with the laser turret.	 PIN: 29 0 python /home/pi/CA2/laserturret.py all commands connected msg.payload
2	Add the following nodes to your flow. 1. Watson IOT Input 2. Exec 3. Debug	

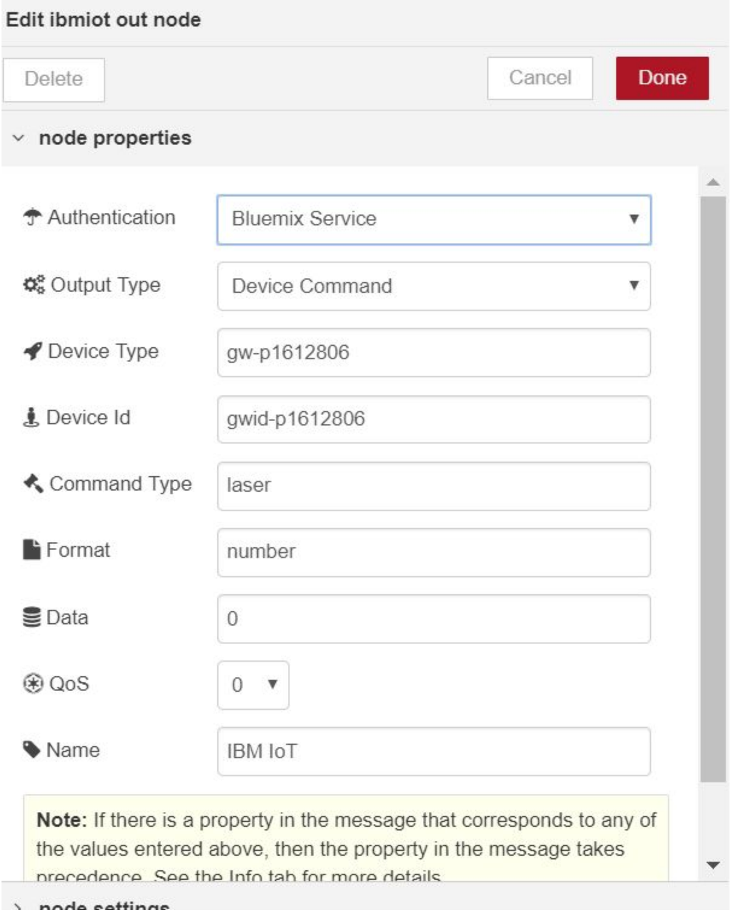
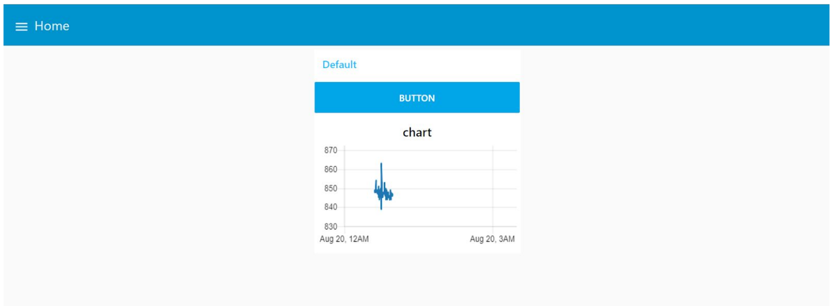
3	Double click your Watson IOT node and set the configurations like so :	 Edit Watson IoT node Delete Cancel Done node properties Connect as: Gateway Credentials: gateway Subscribe to: ○ Gateway commands ● Device commands Device Type: gw-p1612806 Device Id: gwid-p1612806 Command: all commands QoS: 0 Name: node settings
4	You'll be prompted to add new wiotp credentials. Remember previously in section 6 part 1, we added new device gateway and type. We were also given credentials that we saved. This is where you put those credentials.	 Edit Watson IoT node > Edit wiotp-credentials node Delete Cancel Update Organization: o6c5hl Server-Name: o6c5hl.messaging.internetofthings.ibmcloud.com Device Type: gw-p1612806 Device ID: gwid-p1612806 Auth Token: •••••••• Keep Alive: 60 Seconds ☑ Use Clean Session □ Enable secure (SSL/TLS) connection Name: gateway

5	Double click your exec node. In Section 4, we created a python code called laserturret.py where we store inside our RPi. Now, we'll call it.	 Edit exec node Delete Cancel Done node properties node settings Command: python /home/pi/CA2/laserturret.py Append: ☑ msg.payload extra input parameters Output: when the command is complete - exec mode ▾ ☐ Use old style output (compatibility mode) Timeout: optional seconds Name: Name
6	Double click the debug node. This is just for our reference later in case we need it.	 Edit debug node Output: ▾ msg. payload To: ☑ debug window ☐ system console ☐ node status (32 characters) Name: Name
7	Run node-red on the defence RPi and continue to the next section.	

B. Create Node-RED Flow on Bluemix

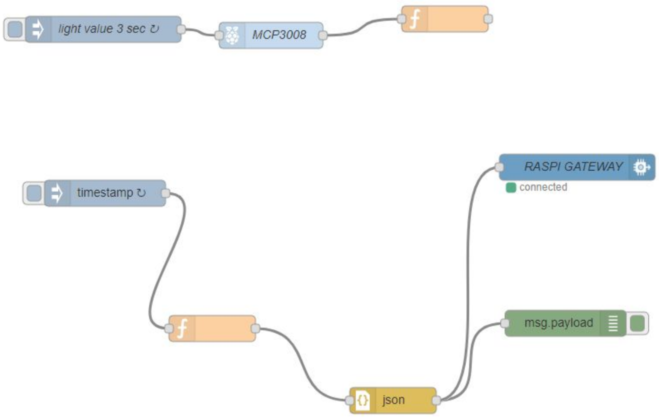
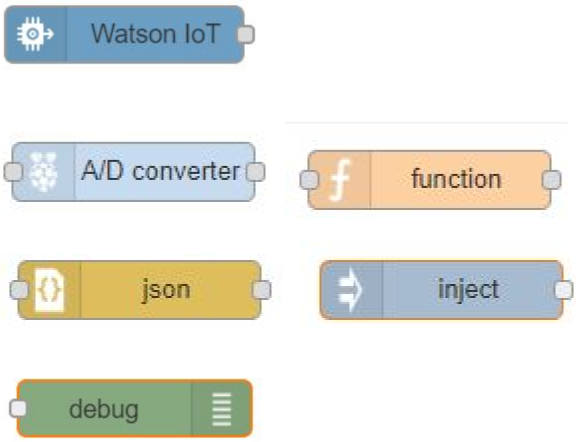
#	Description	Image
1	Back on the IBM Bluemix Console. On our App, you'll see this page. Click "Visit App URL"	 IBM Cloud Catalog Docs Support Manage Search for resource... Nurin Qistina's Account Getting started Overview Runtime Connections Logs Monitoring API Management Cloud Foundry apps / Visit App URL Routes Runtime .js 1 256 256 sp-p1612806-helloworld This app is awake. Org: nurinqistina.16@ichat.sp.edu.sg Location: US South Space: dev BUILDPACK SDK for Node.js™ INSTANCES All instances are running Health is 100% MB MEMORY PER INSTANCE 256 TOTAL MB ALLOCATION 256 0 MB still available
2	You'll see this page. Click go to your Node-RED Flow Editor.	 Node-RED on IBM Bluemix for IBM Watson IoT Platform Node-RED Flow-based programming for the Internet of Things Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. The version running here has been customized for the IBM Watson IoT Platform. More information about Node-RED, including documentation, can be found at nodered.org. Go to your Node-RED flow editor Learn how to customise Node-RED
3	This is the very simple flow that we will create. You will need a button node and a IBM IoT output node.	

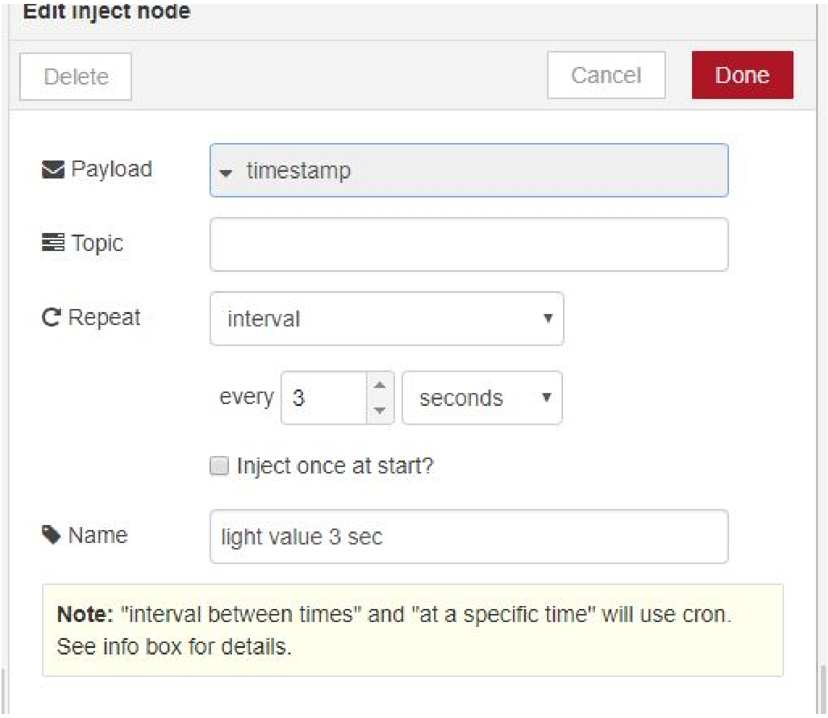
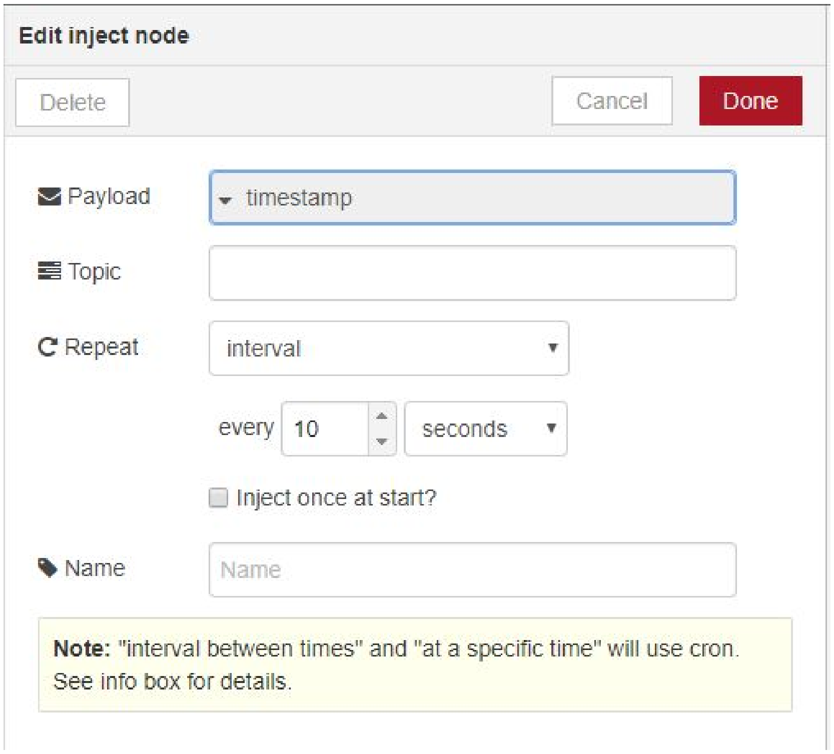
4	Configure the button node as such. This button will stimulate a button press. So whenever this button is pressed, it will send a signal to our Defence RPi and run laserturret.py	 Edit button node Delete Cancel Done node properties Group: Default [Home] Size: auto Icon: optional icon Label: button Colour: optional text/icon color Background: optional background color When clicked, send: Payload: 1 Topic: If msg arrives on input, pass through to output: □ Name:

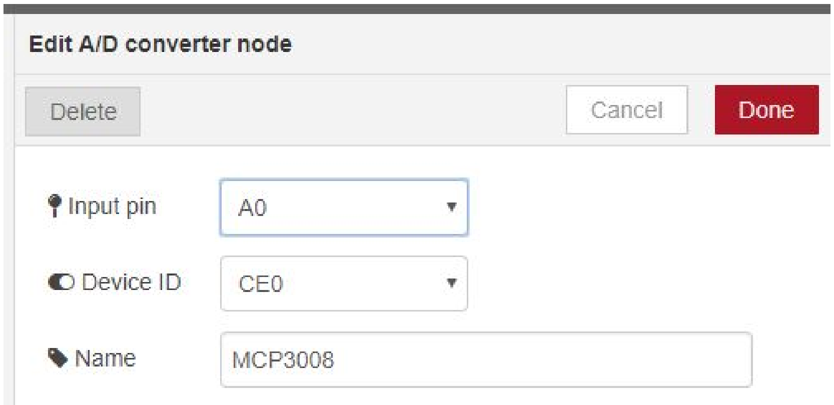
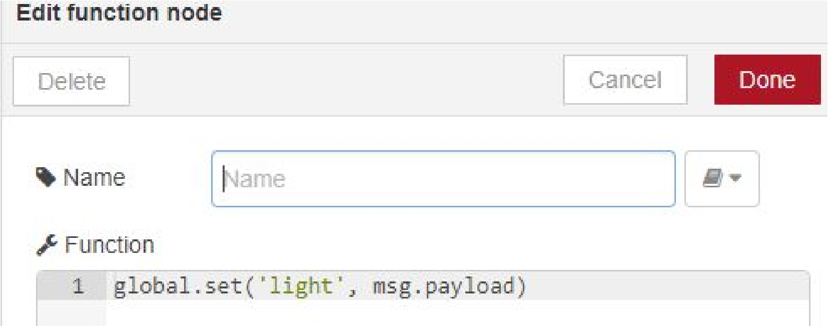
5	Configure the IBM IOT node as such.	 Edit ibmiot out node Delete Cancel Done node properties Authentication: Bluemix Service Output Type: Device Command Device Type: gw-p1612806 Device Id: gwid-p1612806 Command Type: laser Format: number Data: 0 QoS: 0 Name: IBM IoT **Note:** If there is a property in the message that corresponds to any of the values entered above, then the property in the message takes precedence. See the Info tab for more details. node settings
6	Preview on the dashboard. This is what the button will look like on the dashboard! Note that the chart is part of another section in this tutorial.	 Home Default BUTTON chart 870 860 850 840 830 Aug 20, 12AM Aug 20, 3AM
7	Now, when you press the button, your laser turret will start attacking any intruders!! Rmb to run node-red flow for the RPi.	

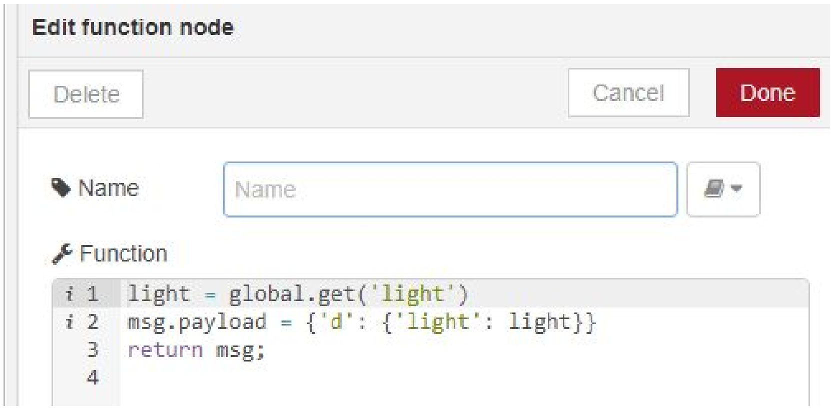
Entertainment

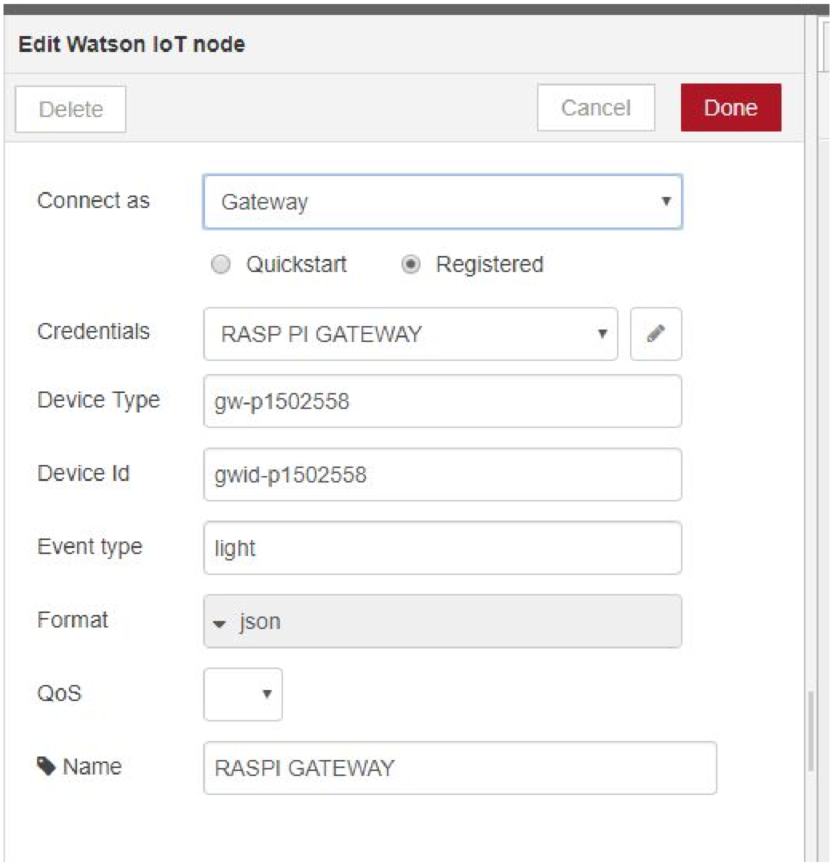
A. Create LDR Node-RED Flow on RPi

#	Description	Image
1	We are going to create a flow on your Entertainment RPi. This is a look on the completed flow. How this works is: It'll keepsending light data from LDR every 10 seconds In the later sections, we'll create a dashboard where you can interact and view charts of given values	 light value 3 sec ↻ MCP3008 timestamp ↻ RASPI GATEWAY connected msg.payload json
2	Add the following nodes to your flow. 4. Watson IOT Input 5. A/D Converter 6. Debug 7. Function 8. Json 9. inject	 Watson IoT A/D converter function json inject debug

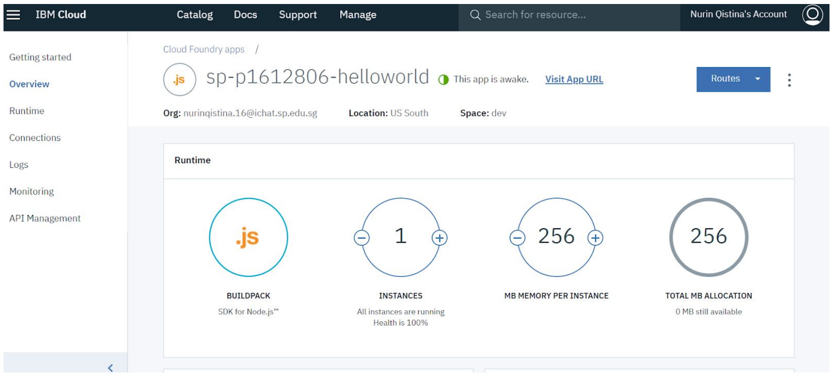
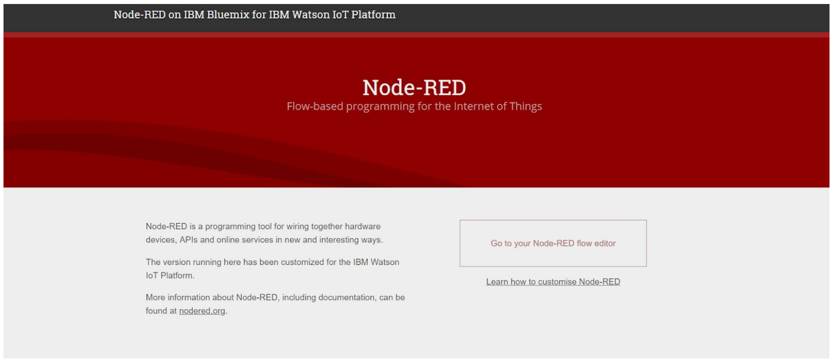
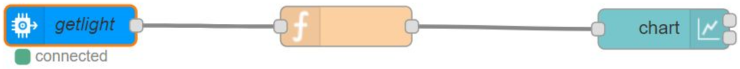
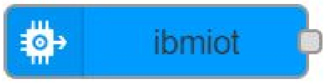
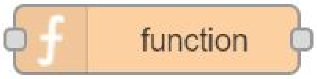
3	Create an injector with interval to store light values and post	 Edit inject node Delete Cancel Done Payload: timestamp Topic: Repeat: interval every 3 seconds □ Inject once at start? Name: light value 3 sec **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.
4	Create an injector to send light values to IOT Watson Gateway	 Edit inject node Delete Cancel Done Payload: timestamp Topic: Repeat: interval every 10 seconds □ Inject once at start? Name: Name **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.

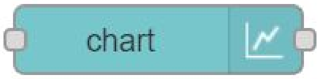
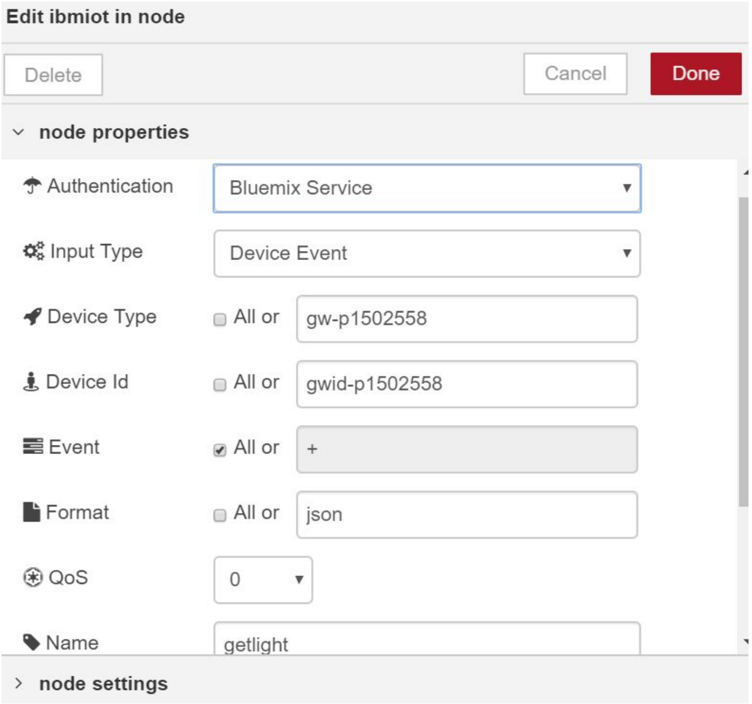
5	Write inputs for MCP3008 node	 Edit A/D converter node Delete Cancel Done Input pin: A0 Device ID: CE0 Name: MCP3008
6	Parse payload from MCP3008 node to set globally	 Edit function node Delete Cancel Done Name: Name Function `global.set('light', msg.payload)`

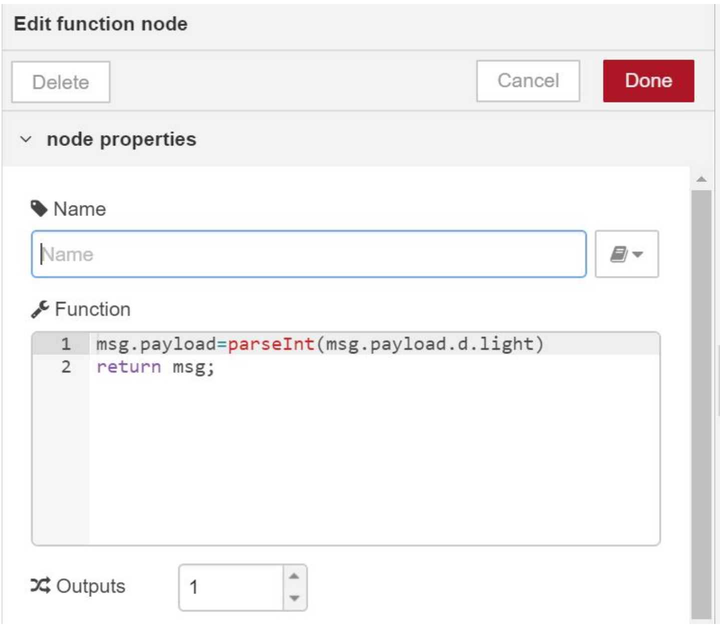
7	Read globally set light vaules and convert to json using json node	
		Edit function node
		Delete Cancel Done
		Name: Name
		Function
		`light = global.get('light')`
		`msg.payload = {'d': {'light': light}}`
		`return msg;`

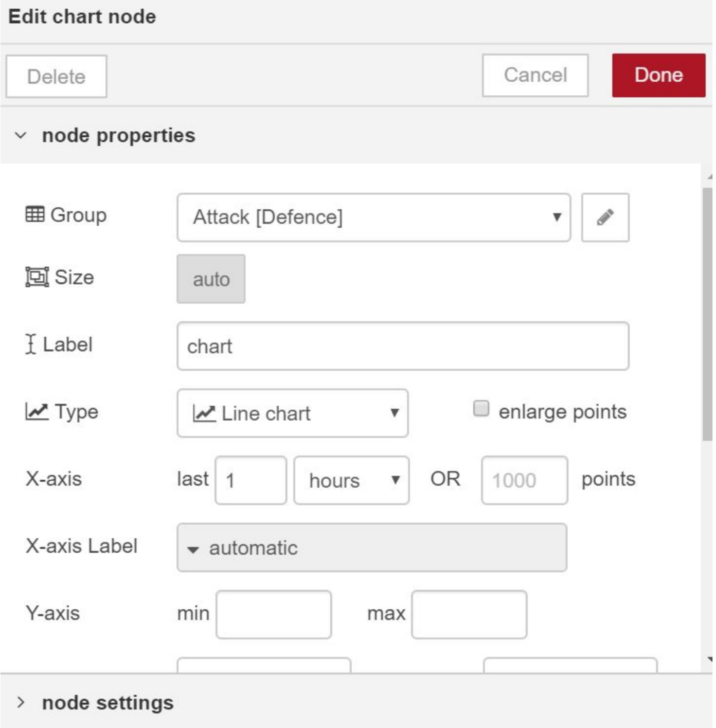
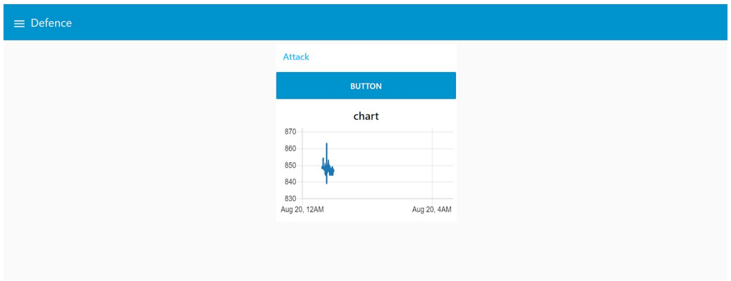
8	Parse into IOT Watson Gateway to be sent to ibm node red dashboard	 Connect as: Gateway ○ Quickstart ● Registered Credentials: RASP PI GATEWAY Device Type: gw-p1502558 Device Id: gwid-p1502558 Event type: light Format: json QoS: Name: RASPI GATEWAY

B. Create LDR Node-RED Flow on Bluemix

#	Description	Image
1	On the IBM Bluemix Console. On our App, you'll see this page. Click "Visit App URL"	 IBM Cloud Catalog Docs Support Manage Search for resource... Nurin Qistina's Account Getting started Overview Runtime Connections Logs Monitoring API Management Routes .jsCloud Foundry apps / sp-p1612806-helloworld This app is awake. Visit App URL Org: nurinqistina.16@ichat.sp.edu.sg Location: US South Space: dev Runtime .js BUILDPACK SDK for Node.js™ 1 INSTANCES All instances are running Health is 100% 256 MB MEMORY PER INSTANCE 256 TOTAL MB ALLOCATION 0 MB still available
2	You'll see this page. Click go to your Node-RED Flow Editor.	Node-RED on IBM Bluemix for IBM Watson IoT Platform Node-RED Flow-based programming for the Internet of Things Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. The version running here has been customized for the IBM Watson IoT Platform. More information about Node-RED, including documentation, can be found at nodered.org. Go to your Node-RED flow editor Learn how to customise Node-RED
3	This is the flow that we will be creating to view ldr values	
4	The nodes you will need are 1. Ibmiot input 2. Function 3. Chart dashboard	 

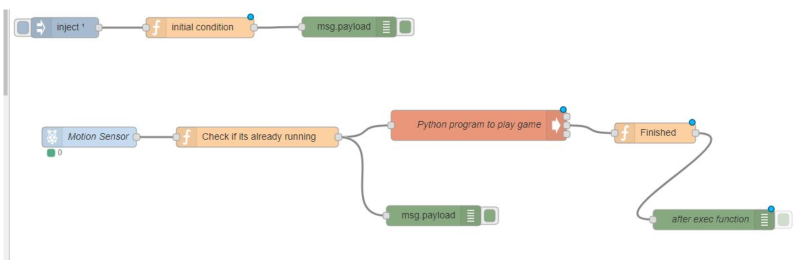
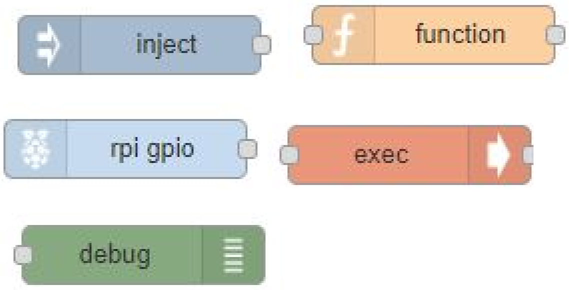
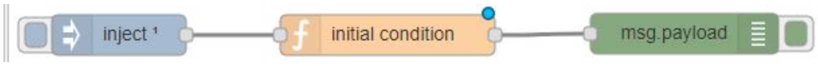
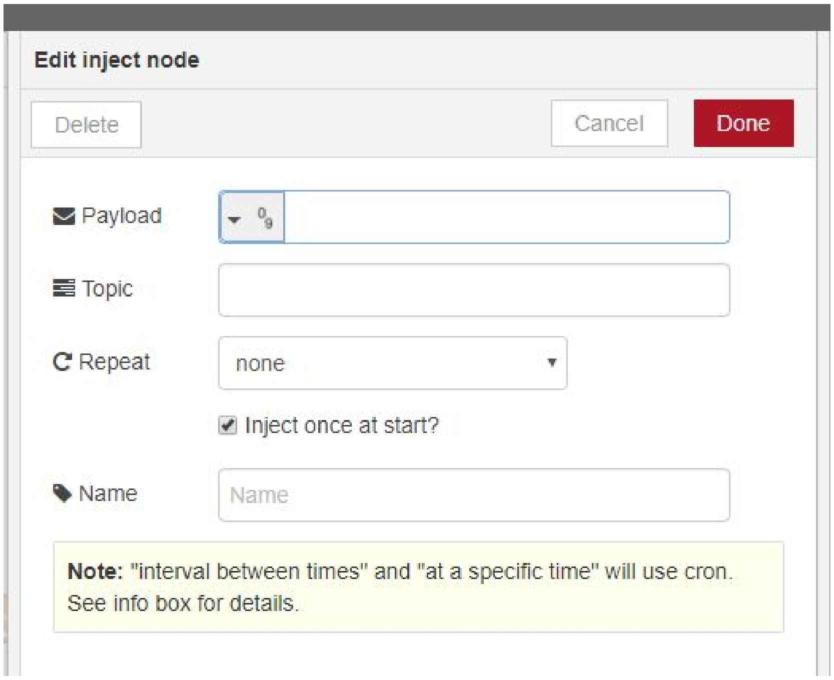
		
5	Double click the ibmiot input node. Configure it like so:	Edit ibmiot in node node properties Authentication: Bluemix Service Input Type: Device Event Device Type: □ All or gw-p1502558 Device Id: □ All or gwid-p1502558 Event: ☑ All or + Format: □ All or json QoS: 0 Name: getlight node settings

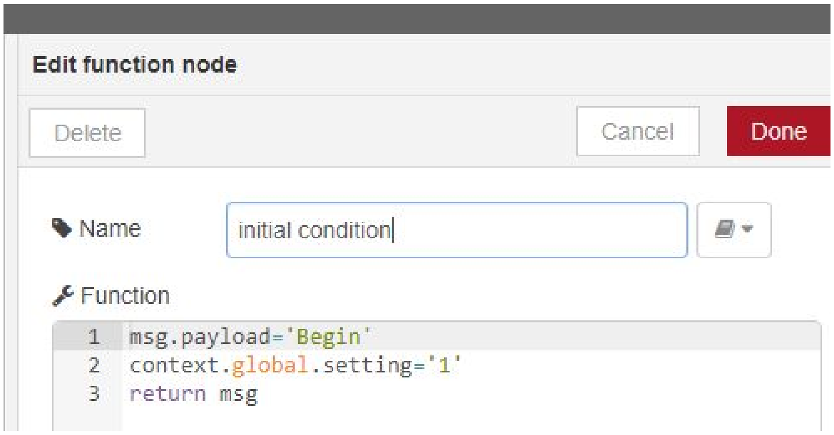
6	Double click the function node.
	Configure it like so
		Edit function node
		node properties
		Name: Name
		Function:
		`msg.payload=parseInt(msg.payload.d.light)`
		`return msg;`
		Outputs: 1

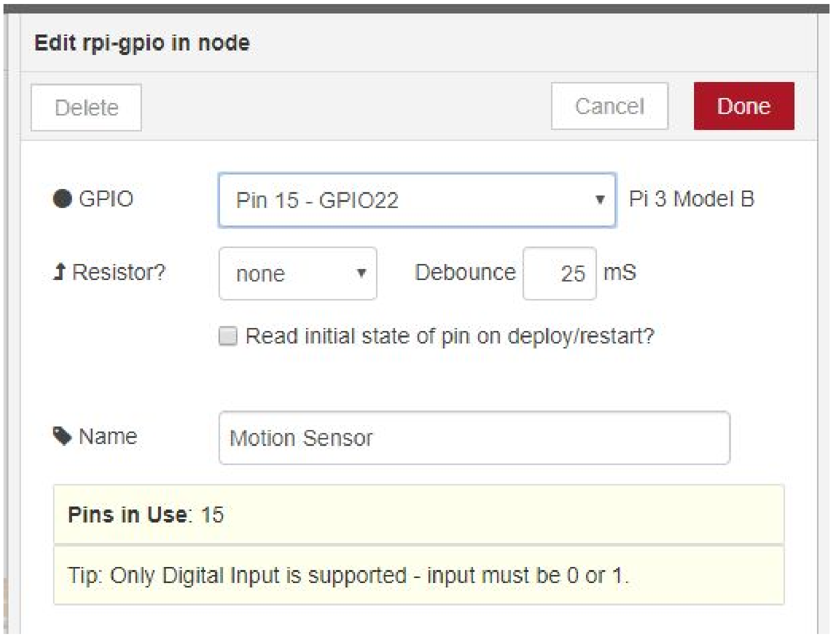
7	Double click the chart node. Configure it like so.	 Edit chart node Delete Cancel Done node properties Group Attack [Defence] Size auto Label chart Type Line chart □ enlarge points X-axis last 1 hours OR 1000 points X-axis Label automatic Y-axis min max node settings
8	Preview the chart on the dashboard! This is what you will see.	 Defence Attack BUTTON chart 870 860 850 840 830 Aug 20, 12AM Aug 20, 4AM

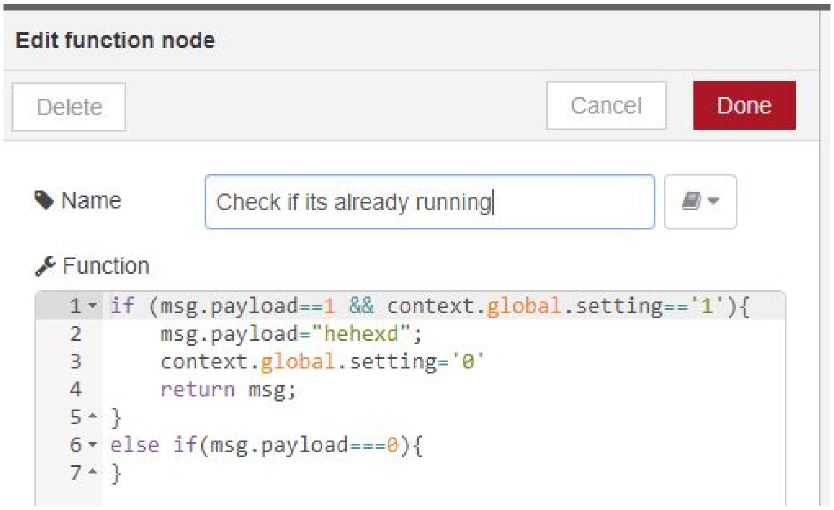
C. Create Entertainment Node-RED Flow on RPi

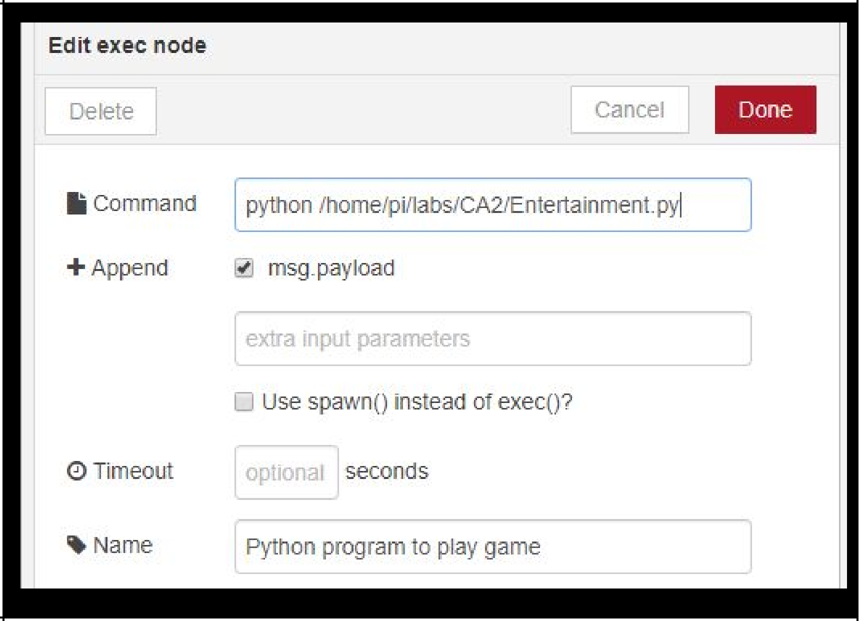
#	Description	Image

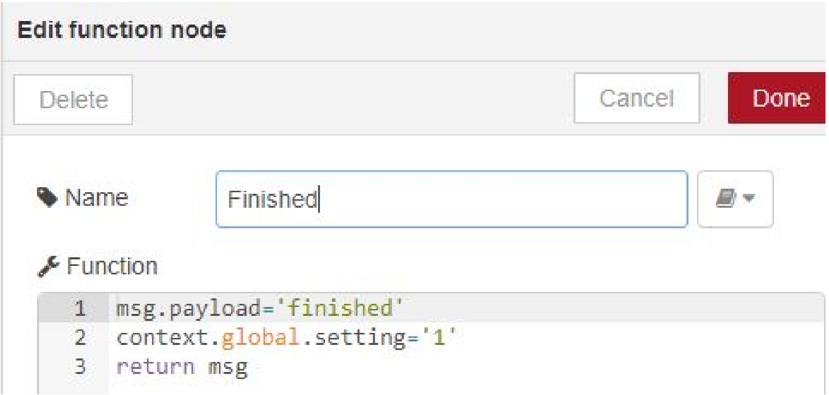
1	On the node-red RPi, create the following flow	 inject initial condition msg.payload Motion Sensor Check if its already running Python program to play game Finished msg.payload after exec function
2	You'll need the following nodes 1. Inject 2. Function 3. Rpi gpio 4. Exec 5. debug	
3	This is the branch that we will be creating to set the initial condition, whereby it will start the loop	
4	Create the inject which will inject once at start	 **Edit inject node** Payload Topic Repeat: none ☑ Inject once at start? Name: Name **Note:** "interval between times" and "at a specific time" will use cron. See info box for details.

5	Followed by setting a function to create a global variable, this will help kick start the entire program	
		Name: initial condition
		Function
		`msg.payload='Begin'`
		`context.global.setting='1'`
		`return msg`

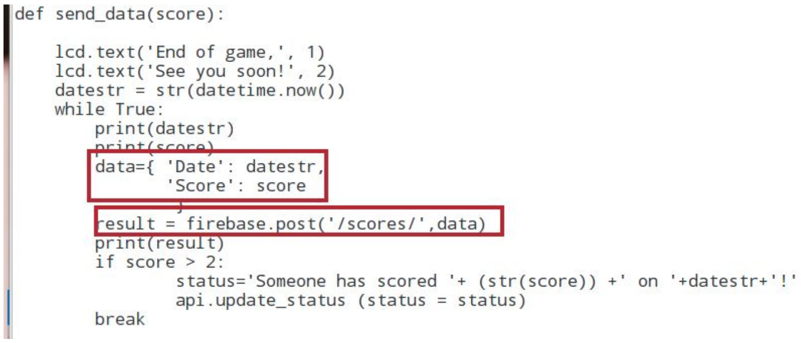
6	Create a rpi gpio with the following settings and gpio pin 22 to read from the motion sensor	 ● GPIO: Pin 15 - GPIO22 ▾ Pi 3 Model B Resistor?: none ▾ Debounce 25 mS □ Read initial state of pin on deploy/restart? Name: Motion Sensor **Pins in Use**: 15 Tip: Only Digital Input is supported - input must be 0 or 1.

7	Do a function that checks if the program is already running, else continue or wait	
		
		Name: Check if its already running
		Function:
		`if (msg.payload==1 && context.global.setting=='1'){`
		`    msg.payload="hehexd";`
		`    context.global.setting='0'`
		`    return msg;`
		`}`
		`else if(msg.payload===0){`
		`}`

8	If the program is not running, execute a python file from the entertainment RPi, which will run the gamification of simon says.	 Command: python /home/pi/labs/CA2/Entertainment.py Append: ☑ msg.payload extra input parameters □ Use spawn() instead of exec()? Timeout: optional seconds Name: Python program to play game

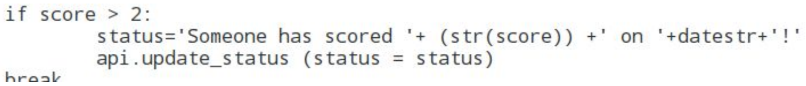
9	Finally, after the program is finished, run a function to set the global variable back to 1, in order to continue the program	
		
		Name: Finished
		Function:
		`msg.payload='finished'`
		`context.global.setting='1'`
		`return msg`

<table>
<tr>
<td>10</td>
<td>This means that when everytime motion is sensed, the program will run and if theres already an instance whereby the game is running, it will not execute another time. Additionally, game scores and timestamps will be stored into firebase nosql database in the executional file which can be seen on the right.</td>
<td>

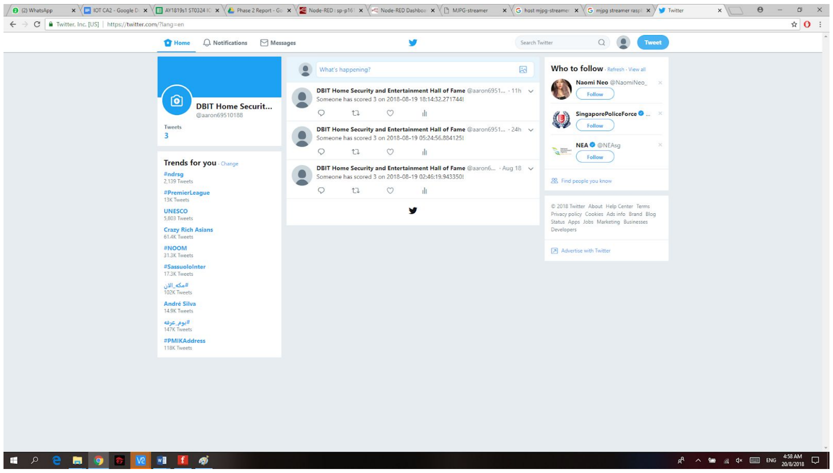


```
def send_data(score):

    lcd.text('End of game,', 1)
    lcd.text('See you soon!', 2)
    datestr = str(datetime.now())
    while True:
        print(datestr)
        print(score)
        data={ 'Date': datestr,
               'Score': score
        }
        result = firebase.post('/scores/',data)
        print(result)
        if score > 2:
                status='Someone has scored '+ (str(score)) +' on '+datestr+'!'
                api.update_status (status = status)
        break
```

</td>
</tr>
<tr>
<td>11</td>
<td>Additionally, a twitter bot was implemented as our hall of fame channel whereby the code will automatically detect people who score more than 3 points and will post it on our twitter page !</td>
<td>



```
if score > 2:
        status='Someone has scored '+ (str(score)) +' on '+datestr+'!'
        api.update_status (status = status)
break
```



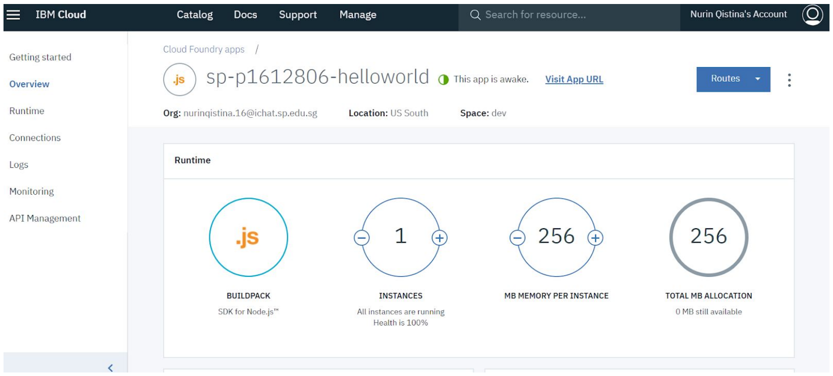
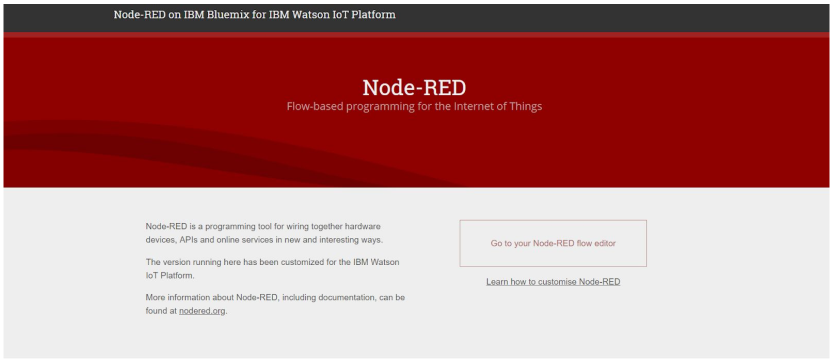
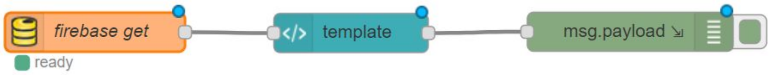
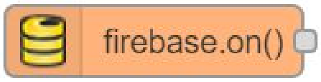
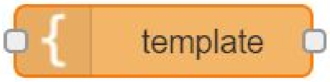
DBIT Home Security and Entertainment Hall of Fame @aaron6951... · 11h
Someone has scored 3 on 2018-08-19 18:14:32.271744!

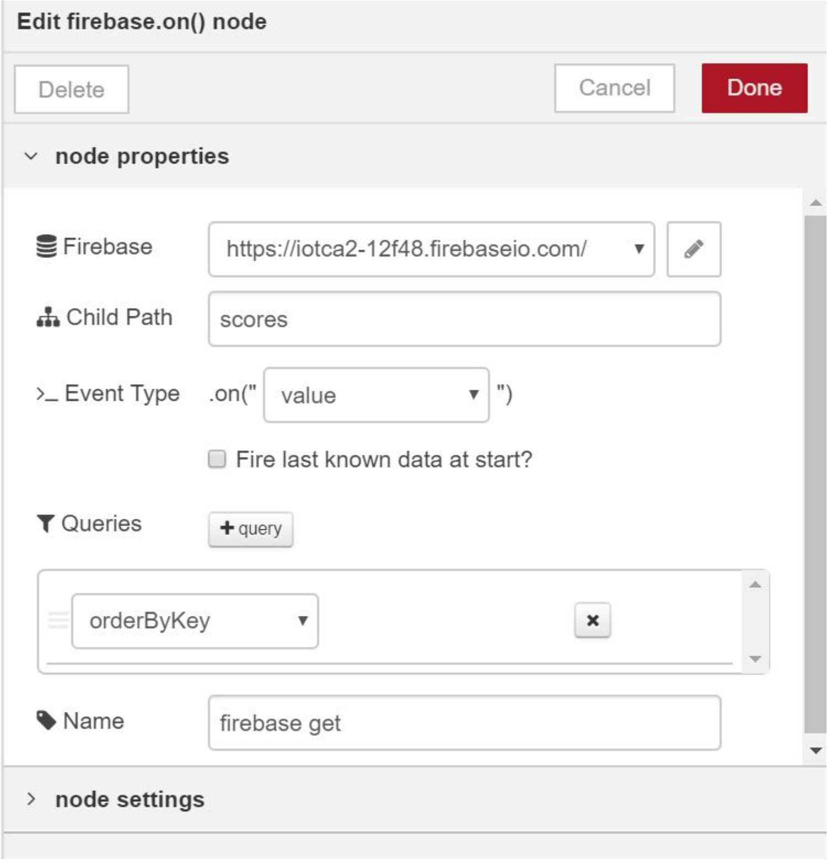
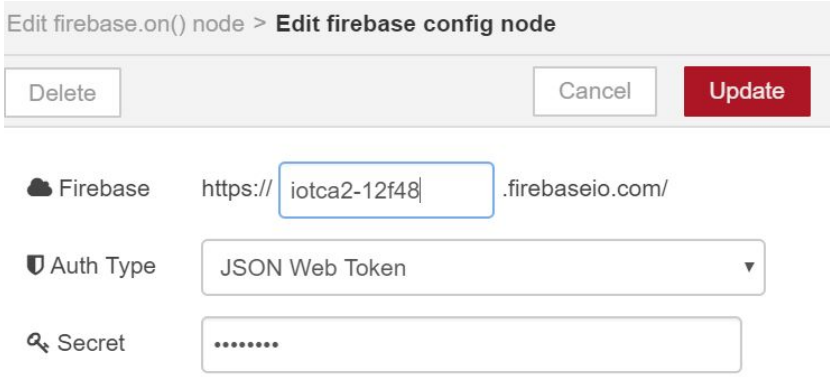
DBIT Home Security and Entertainment Hall of Fame @aaron6951... · 24h
Someone has scored 3 on 2018-08-19 05:24:56.884125!

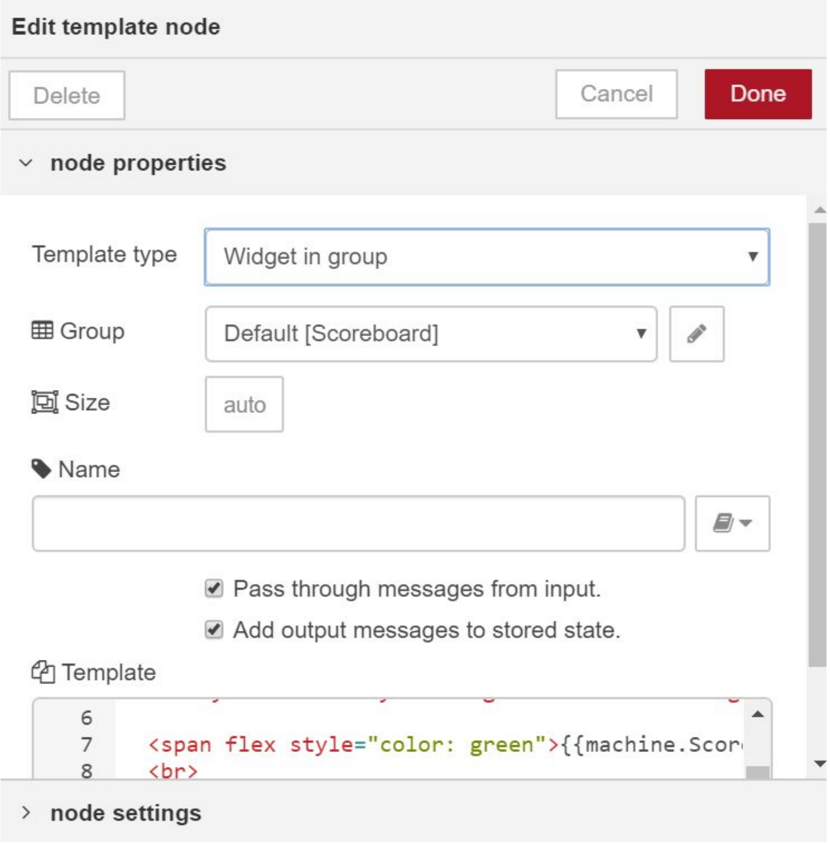
DBIT Home Security and Entertainment Hall of Fame @aaron6... · Aug 18
Someone has scored 3 on 2018-08-19 02:46:19.943350!

</td>
</tr>
</table>

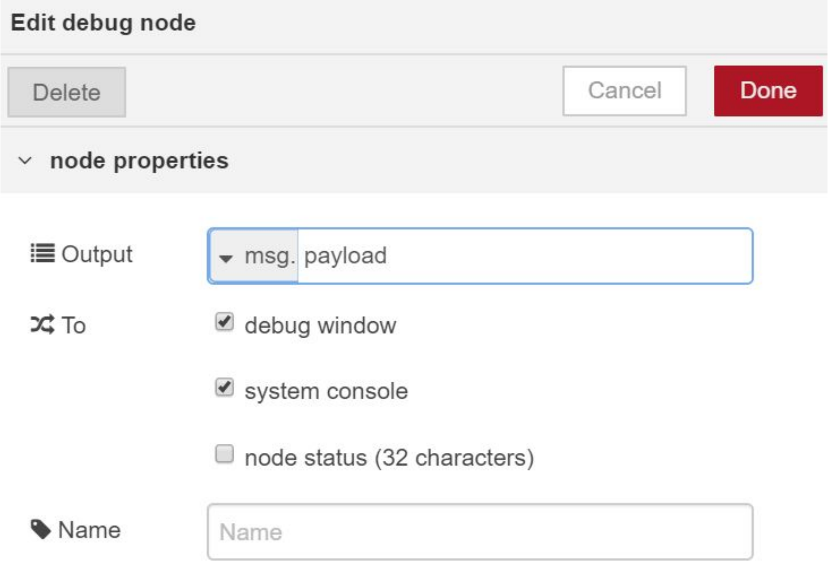
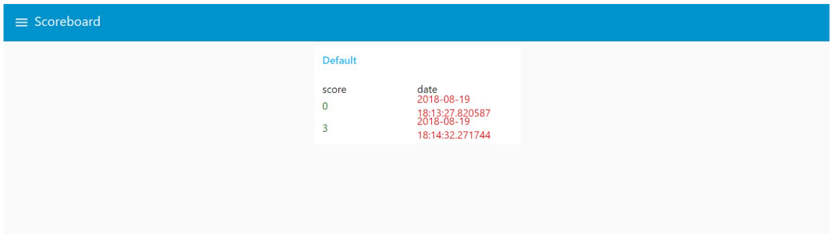
D. Create Scoreboard Node-RED Flow on Bluemix

#	Description	Image
1	On the IBM Bluemix Console. On our App, you'll see this page. Click "Visit App URL"	 IBM Cloud Catalog Docs Support Manage Search for resource... Nurin Qistina's Account Getting started Overview Runtime Connections Logs Monitoring API Management Cloud Foundry apps / Routes Runtime .js sp-p1612806-helloworld This app is awake. Visit App URL Org: nurinqistina.16@ichat.sp.edu.sg Location: US South Space: dev BUILDPACK SDK for Node.js™ 1 INSTANCES All instances are running Health is 100% 256 MB MEMORY PER INSTANCE 256 TOTAL MB ALLOCATION 0 MB still available
2	You'll see this page. Click go to your Node-RED Flow Editor.	 Node-RED on IBM Bluemix for IBM Watson IoT Platform Node-RED Flow-based programming for the Internet of Things Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. The version running here has been customized for the IBM Watson IoT Platform. More information about Node-RED, including documentation, can be found at nodered.org. Go to your Node-RED flow editor Learn how to customise Node-RED
3	This is the flow that we will be creating to list down the scores from the simon-says game.	
4	You will need the following nodes. 4. Firebase.on 5. Template 6. Debug	 

5	Double click the firebase.on node.	 Edit firebase.on() node Delete Cancel Done node properties Firebase: https://iotca2-12f48.firebaseio.com/ Child Path: scores Event Type: .on(" value ") □ Fire last known data at start? Queries: + query orderByKey Name: firebase get node settings
6	You will be prompted to config the firebase auth	 Edit firebase.on() node > Edit firebase config node Delete Cancel Update Firebase: https:// iotca2-12f48 .firebaseio.com/ Auth Type: JSON Web Token Secret: ••••••••

7	Double click the template node.
	Configure it as such.
		
		Edit template node
		
		Template type: Widget in group
		Group: Default [Scoreboard]
		Size: auto
		Name:
		☑ Pass through messages from input.
		☑ Add output messages to stored state.
		Template
		`<span flex style="color: green">{{machine.Scor`
		`<br>`

8	In the code section, we'll paste the following codes;	`<div layout="row" layout-align="start center">`
		`  <span flex>score</span>`
		`  <span flex>date</span>`
		`</div>`
		`<div layout="row" layout-align="start center"`
		`ng-repeat="machine in msg.payload">`
		<br>`  <span flex style="color:`
		`green">{{machine.Score}}</span>`<br>`
		`<br>` <span flex style="color:`
		`red">{{machine.Date}}</span>`
		`</div>`

9	In the debug code, configure it as such. This is just for our reference in case we need it later on.	 Edit debug node Delete Cancel Done node properties Output msg. payload To ☑ debug window ☑ system console ☐ node status (32 characters) Name Name
10	Preview it on the dashboard!	 Scoreboard Default score date 0 2018-08-19 18:13:27.820587 3 2018-08-19 18:14:32.271744