SCHOOL OF COMPUTING (SOC)

IOT CA2
Step-by-step Tutorial

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

ST0324 Internet of Things (IOT)

Date of Submission: 23 Feb 2020 (Sunday)

Prepared for: Ms Dora

Class: DIT/FT/2A/34

Submitted by: Group 1

Student ID	Name
1841495	Jason Yeo Chee Kang
1828993	Huang Renjie
1844331	Felicia Tee Li Yi

Table of Contents

Section 1
Overview of project

A. Where we have uploaded our tutorial

Fill up the Google form here to submit your links and then paste the links here of your Youtube and tutorial document here as well.

http://bit.ly/1910s2iotca2

Youtube	
Public tutorial link	https://github.com/Feliciaaaaa/IOTCA2

B. What is the application about?

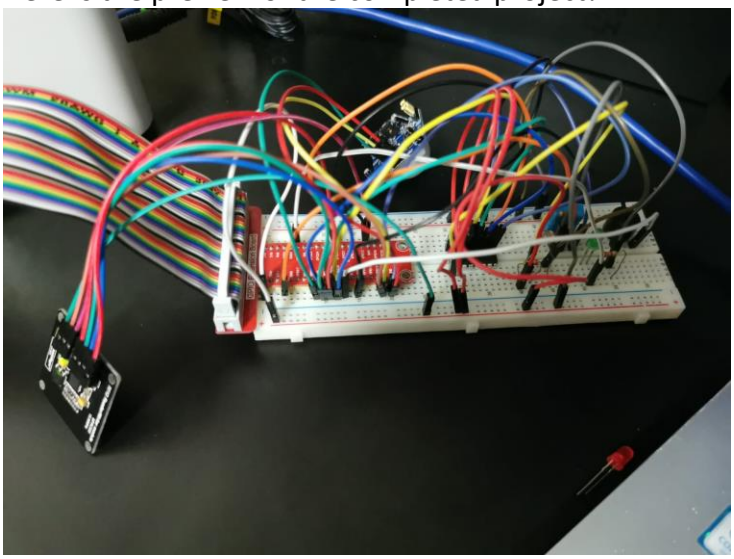
Security Room

This application provides function to remotely turn on/off the main light, view real-time state of temperature and humidity, and view historical light values as a graph and table. Besides, it has additional functions such as take a picture and send it to telegram when motion is detected, and store the picture taken in a cloud object storage called AWS S3. The application also takes a picture at the current time and send it through Telegram, and as all the staff has left the room, the room will be dark and you'll get a alert on Telegram.

This application is open for everyone, as it is simple for all generations to contorl the lights in their room hence ensure the safety of their workplace.

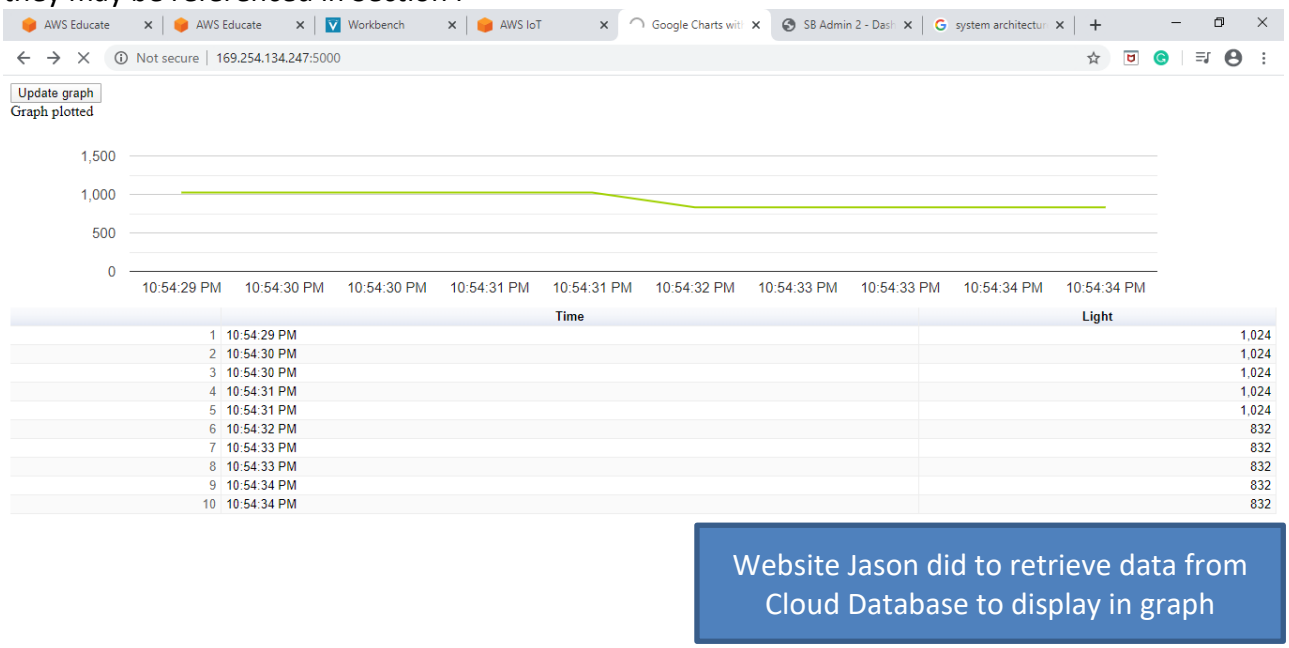
C. How does the final RPI set-up looks like?

Here is the preview of the completed project!



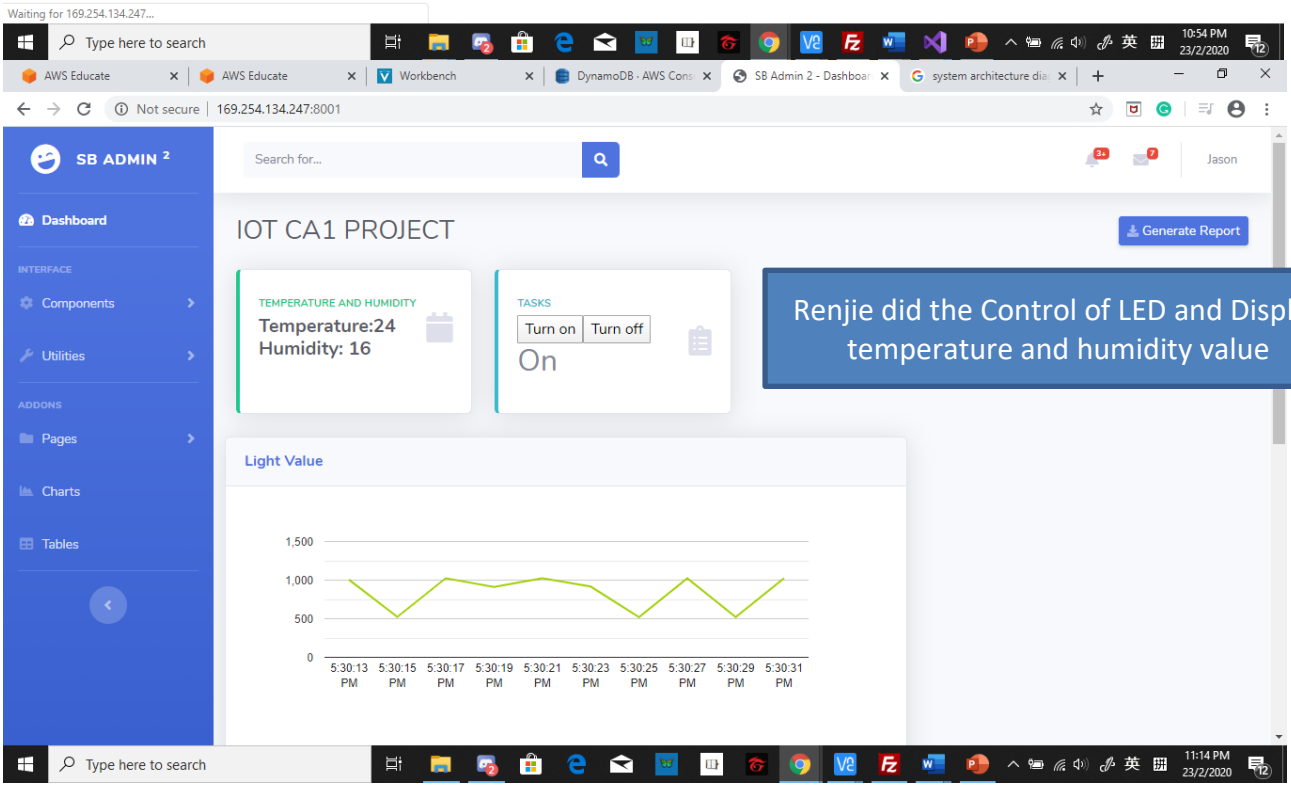
D. How does the web or mobile application look like?

Provide at least one screenshot of your web app, and more if your web app consists of more than 1 page. Otherwise, I will assume your webapp only can show 1 page. Label your screenshots so that they may be referenced in Section F



	Time	Light
1	10:54:29 PM	1,024
2	10:54:30 PM	1,024
3	10:54:30 PM	1,024
4	10:54:31 PM	1,024
5	10:54:31 PM	1,024
6	10:54:32 PM	832
7	10:54:33 PM	832
8	10:54:33 PM	832
9	10:54:34 PM	832
10	10:54:34 PM	832

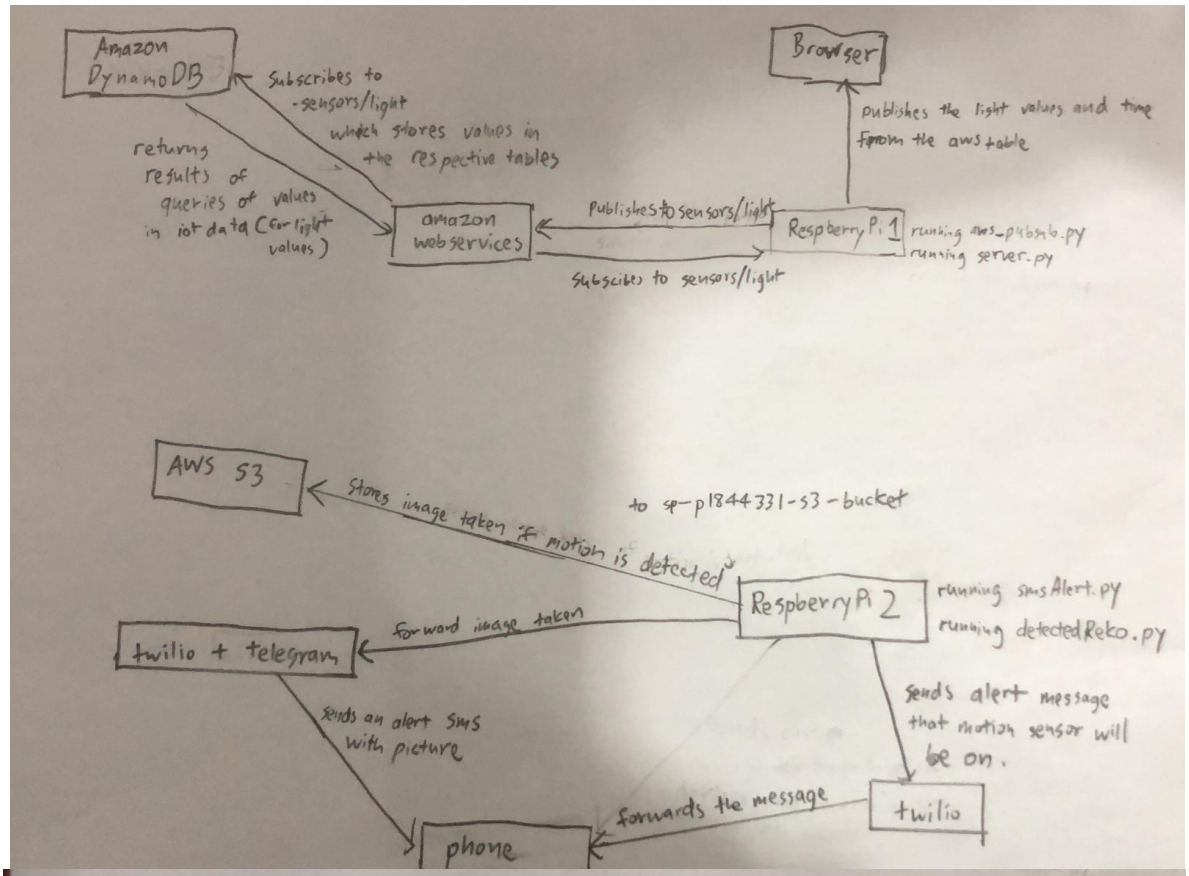
Website Jason did to retrieve data from Cloud Database to display in graph

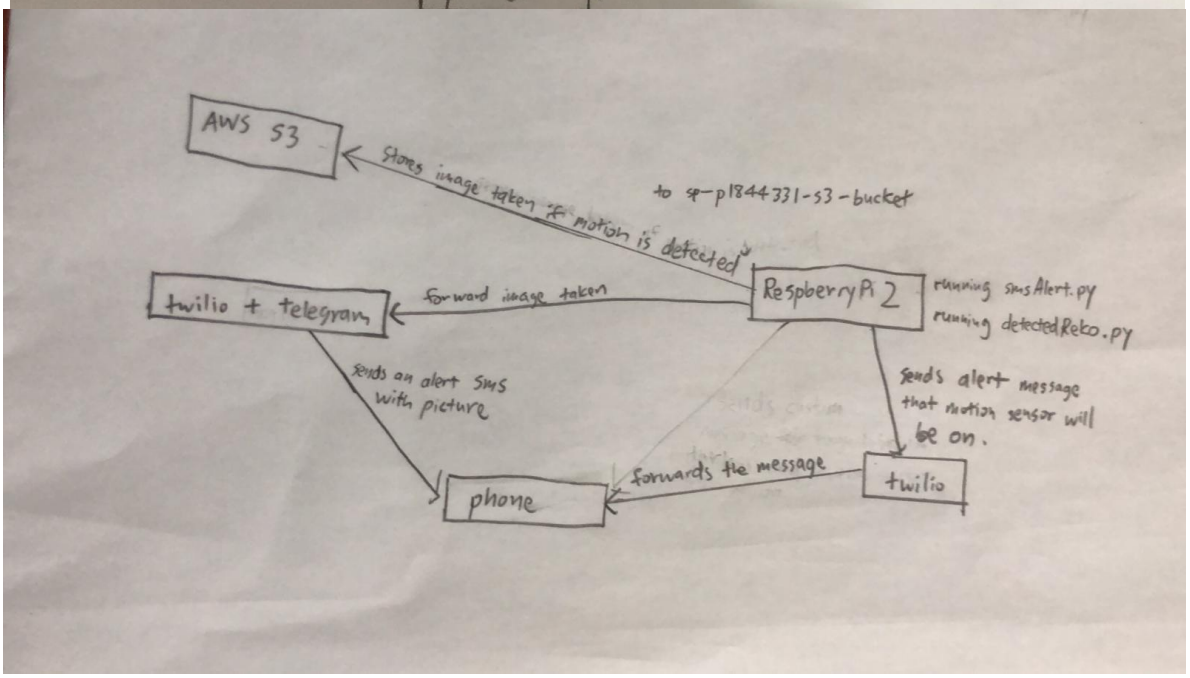


Renjie did the Control of LED and Display temperature and humidity value

E. System architecture of our system

Provide a hand-drawn or computer-drawn system architecture diagram please. Example given below.





F. Evidence that we have met basic requirements

Provide bullet list to describe how your group has met basic requirements

Requirement	Evidence
Used three sensors	LightSensor, Temperature and Humidity Sensor, Motion Sensor
Used MQTT	MQTT PAHO
Stored data in cloud	Stored light data in Cloudant database in IBM cloud Store images in AWS S3
Used cloud service	Use AWS Rekognition, hosted web server on EC@
Provide real-time sensor value / status	Show the real-time value of light sensor
Provide historical sensor value/ status	Show historical vlaue of light sensor
Control actuator	Placed button on webpage to control LED actuator

G. Bonus features on top of basic requirements

- Easy – Mobile App to control the door
- Medium – Able to send photo to telegram when motion sensor detect someone trying to move the device and buzzer will make sounds. Photo is upload to Amazon S3.
- Easy – Card reader, to open door for registered user

A. Quick-start guide (Readme first)

Give a few lines of basic instructions on how I need to run your app, e.g

1) First connect hardware as in Section 2
2) Run aws_pubsub_edited.py to record the light values into the database
3) Then run the server.py file at separate tab for web server
4) Run the local host server to see the results
5) **Additional : Run 3 more bonus feature mentioned later below to see more functions
6) Run rpi_blynk to control LED from phone app and see newest light value
7) Run smsAlert, sms send to user when room is dark.
8) Run detectedReko for security purpose, when motion detect picam will take photo and send to user telegram and SMS, photo will store in Amazon A3.

Section 2
Hardware requirements

Hardware checklist

A simple one like this is sufficient. **No need** to describe the hardware in details.

1. One DHT sensor
2. One LED
3. One motion sensor
4. One buzzer
5. One light sensor
6. One MCP3008
7. One MFRC522 Card Reader
8. Three resistors (One 10K ohm, Two 330 ohm)

Hardware setup instructions

Describe any special setup instructions here

Fritzing Diagram

Paste a Fritzing diagram of your setup here

You can get the Fritzing software at Blackboard Labs folder (third link from top)

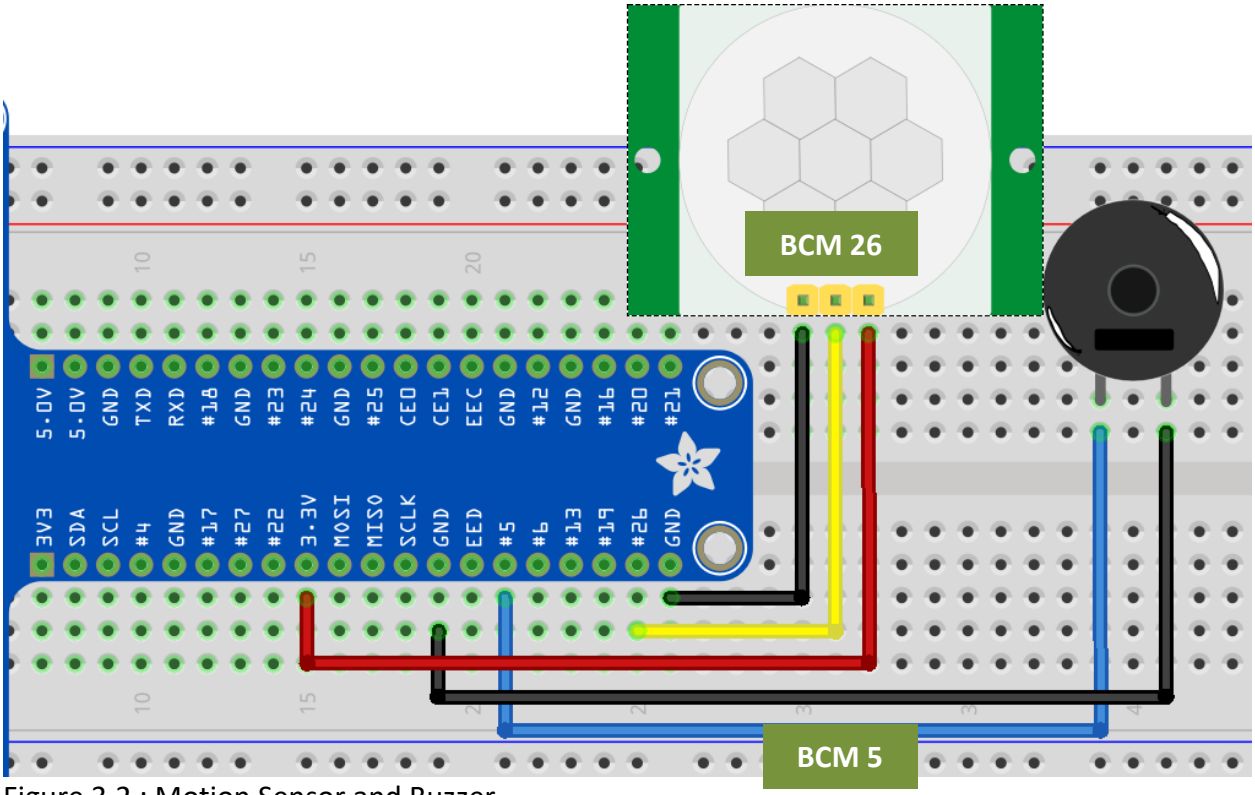
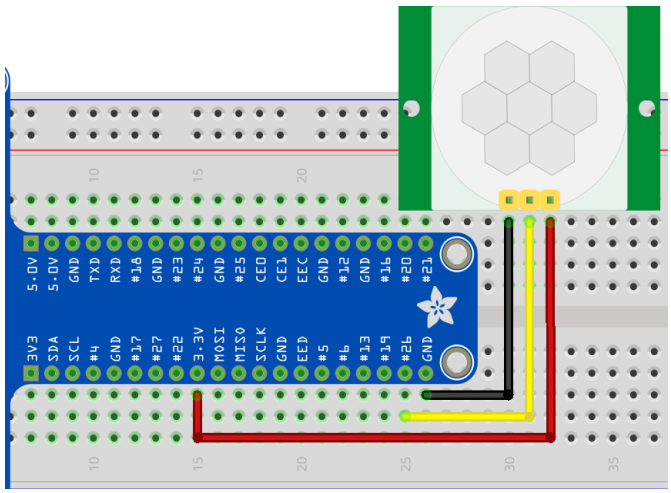
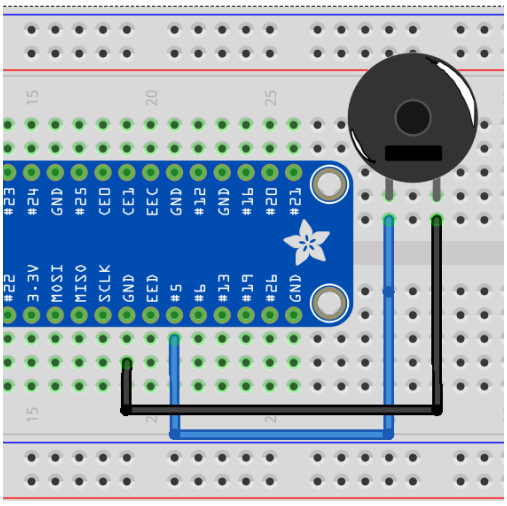


Figure 3.2 : Motion Sensor and Buzzer



Jumper color	PIR pin	RPi pin
Red	VCC	3.3V
Black	GND	GND
Yellow	VOUT	BCM26

Figure 3.2.1 : Motion Sensor



Jumper color	PIR pin	RPi pin
Black	GND	GND
Blue	VOUT	BCM5

Figure 3.2.2 : Buzzer

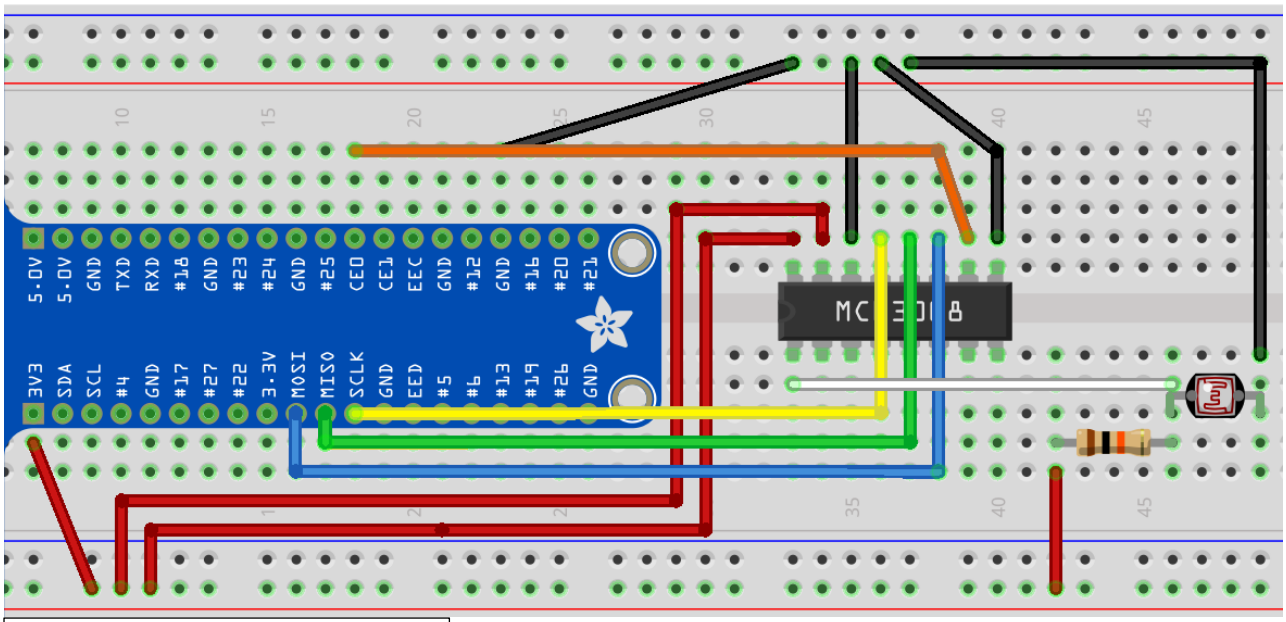
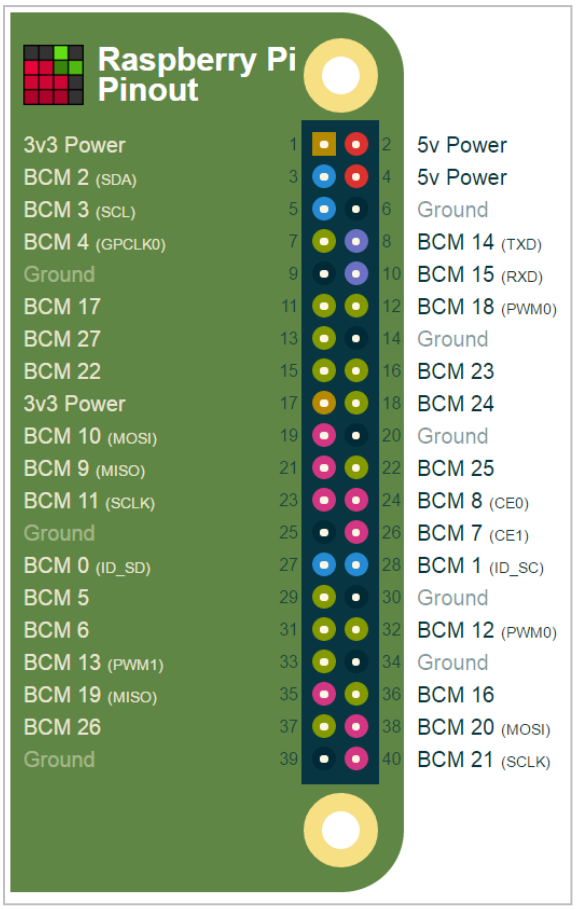


Figure 3.3 : LDR and MCP3008

MCP3008	Raspberry Pi	Jumper Color
Pin 16 VDD	3V3	Red
Pin 15 VREF	3V3	Red
Pin 14 AGND	GND	Black
Pin 13 CLK	BCM11 SCLK	Yellow
Pin 12 DOUT	BCM9 MISO	Green
Pin 11 DIN	BCM10 MOSI	Blue
Pin 10 CS/SHDN	BCM8 CE0	Orange
Pin 9 DGND	GND	Black
Pin 1	To LDR	White



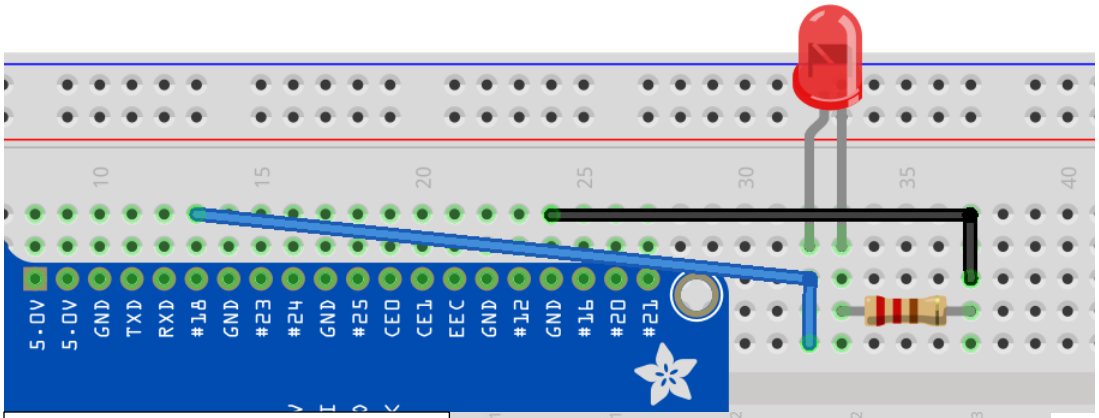
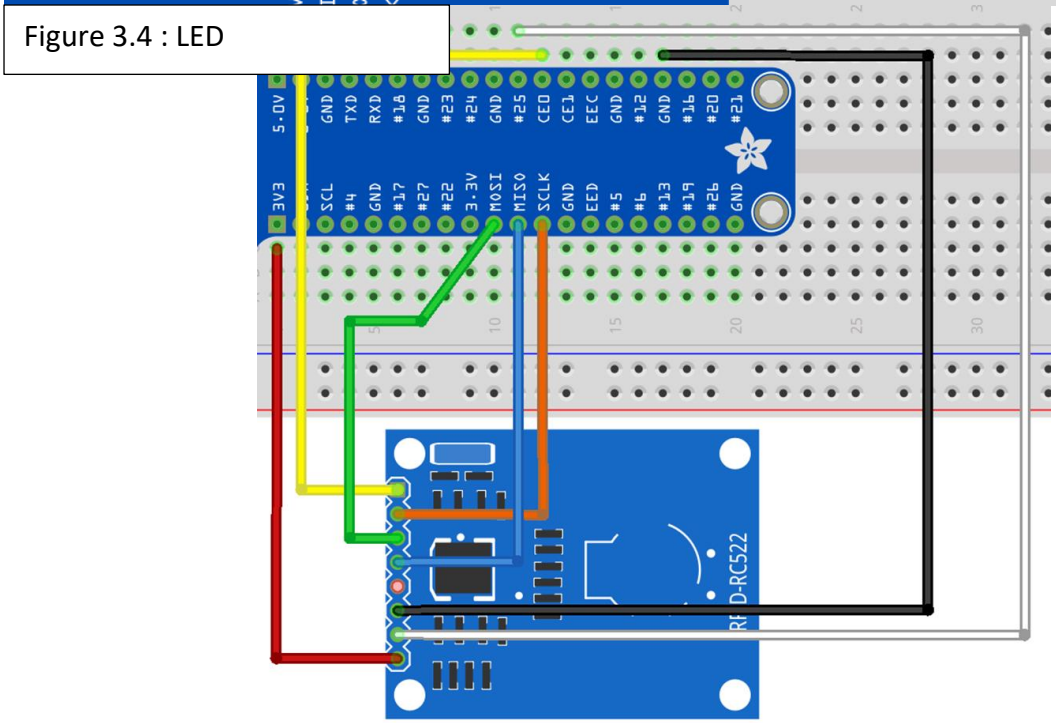
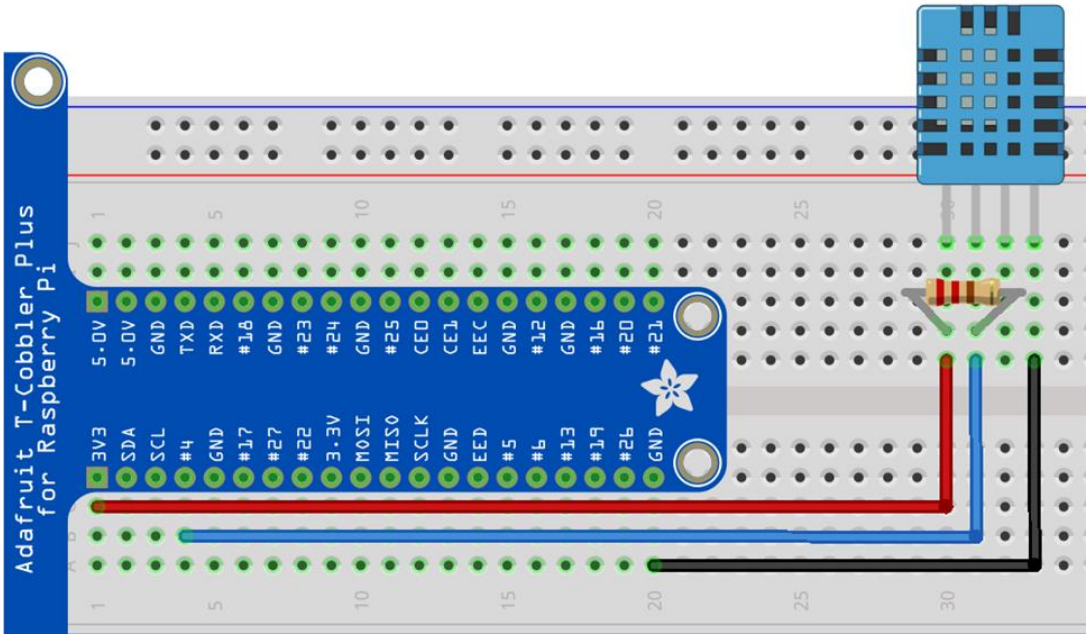


Figure 3.4 : LED





Section 3
Software Requirements

Software checklist

If your applications needs the user to install additional Python or other libraries, pleasse provide here. A simple one like this is sufficient.

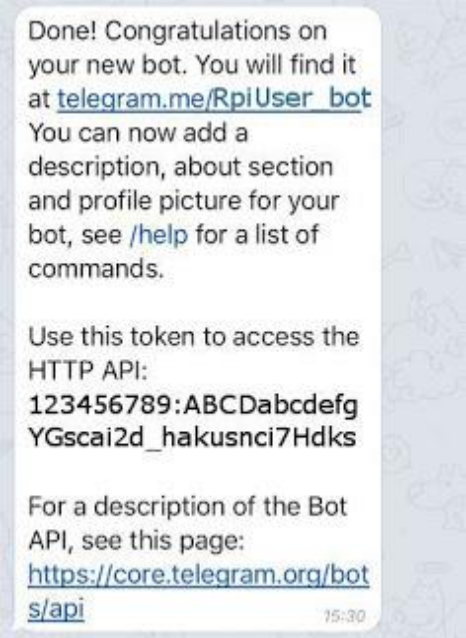
1. python-telegram-bot
2. python-pip
3. sudo pip install blynkapi –upgrade
4. blynk app on your phone
5. sudo pip install --upgrade --force-reinstall pip==9.0.3
6. sudo pip install AWSIoTPythonSDK --upgrade --disable-pip-version-check
7. sudo pip install --upgrade pip
8. sudo pip install paho-mqtt
9. sudo pip install awscli
10. sudo apt-get install python-dev
11. cd ~
    git clone https://github.com/lthiery/SPI-Py.git
    cd ~/SPI-Py
    sudo python setup.py install
12. cd ~
    git clone https://github.com/pimylifeup/MFRC522-python.git
    cd ~/MFRC522-python
    sudo python setup.py install

Software setup instructions

Describe any special setup instructions here, e.g some libraries you need to pip install or some API key you need to create/request etc

Software setup instructions –detectedReko.py

Then start a chat with **@BotFather**. This is the main **Telegram bot** and you can use it to create your **bots**. To create a new bot, just send the command */newbot* and **BotFather** wil ask you to give it a name and then a user name.



The token (in the picture above it's just an example) will be used by our scripts to use the bot

To use a **Telegram bot** with a **Raspberry Pi** you need to install the Python Telegram Bot library. To do so just use the following command from the shell:

```
pip install python-telegram-bot
```

if you do not have pip installed, just use this before the previous statement:

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

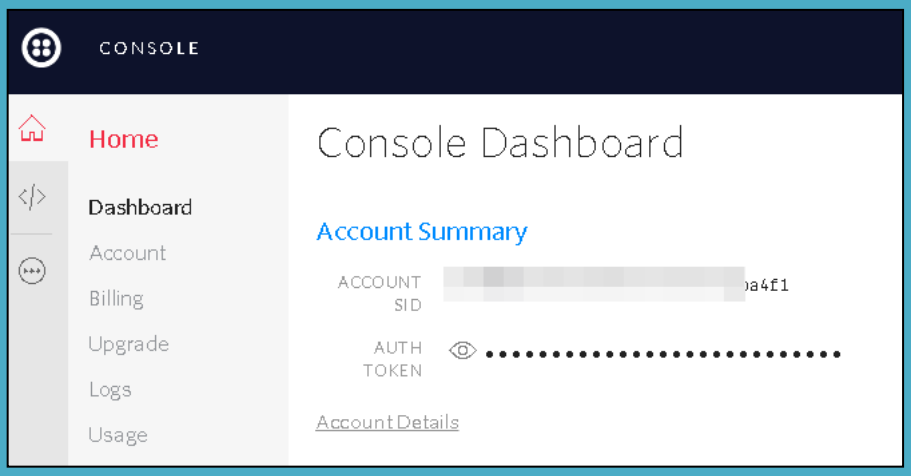
If the **Telegram** library gives an error when trying to install, just install this one before and then retry:

```
sudo pip install future
```

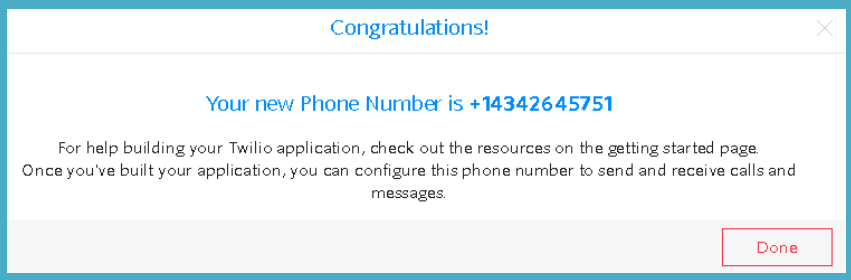
To get your chat id, get to this link with your token:
https://api.telegram.org/bot<YOUR TOKEN>/getUpdates
For example mine would be like this :
https://api.telegram.org/bot778518288:AAEpuWpCgYH2YF3bK0CZDGg6tEvVjjiT43E/getUpdates
Send a message to your bot and refresh the screen. This will be your chat id.


```
{"ok":true,"result":[{"update_id":670475781,
"message":{"message_id":6,"from":{"id":638292737,"is_bot":false,"first_name":"Felicia","username":"ft_ly00","language_code":"en"},"chat":
{"id":638292737,"first_name":"Felicia","username":"ft_ly00","type":"private"},"date":1578223222,"text":"/start","entities":[{"offset":0,"length":6,"type":"bot_command"}]}},
{"update_id":670475782,
"message":{"message_id":7,"from":{"id":638292737,"is_bot":false,"first_name":"Felicia","username":"ft_ly00","language_code":"en"},"chat":
{"id":638292737,"first_name":"Felicia","username":"ft_ly00","type":"private"},"date":1578223781,"text":"Hi"}},{"update_id":670475783,
"message":{"message_id":8,"from":{"id":638292737,"is_bot":false,"first_name":"Felicia","username":"ft_ly00","language_code":"en"},"chat":
{"id":638292737,"first_name":"Felicia","username":"ft_ly00","type":"private"},"date":1578223793,"text":"/start","entities":[{"offset":0,"length":6,"type":"bot_command"}]}},
{"update_id":670475784,
"message":{"message_id":10,"from":{"id":638292737,"is_bot":false,"first_name":"Felicia","username":"ft_ly00","language_code":"en"},"chat":
{"id":638292737,"first_name":"Felicia","username":"ft_ly00","type":"private"},"date":1578298776,"text":"Hi"}}]}
```

Copy your Twilio Account SID **and** Auth Token **which should be shown on the dashboard when you log in.**



After creating your phone number, copy it down your, you will need it later

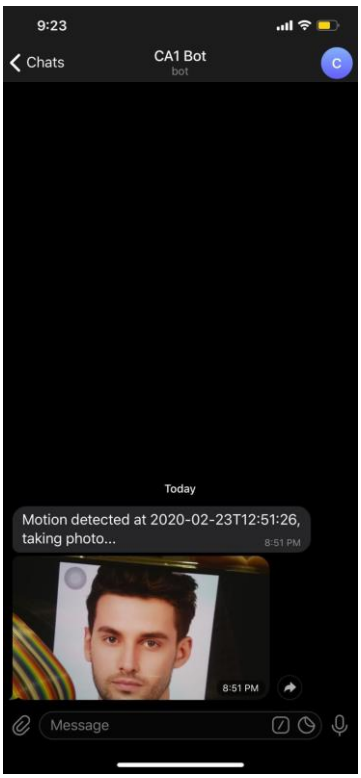


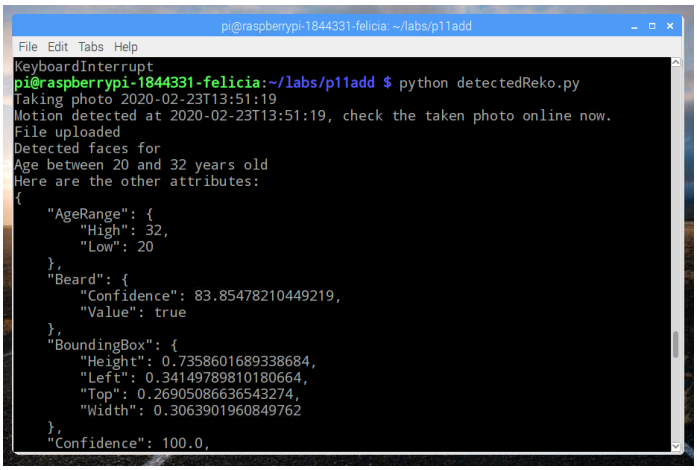
Now you can fill in details of your token and chat id to continue with detectedReko.py and smsAlert.py.

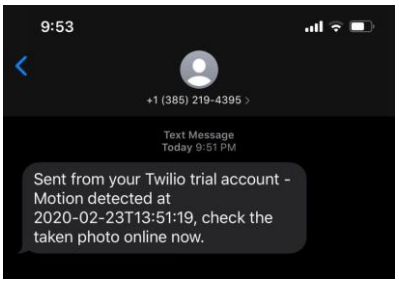
**Remember! Ensure you DO NOT use sudo for detectedReko.py.

```
python detectedReko.py
```

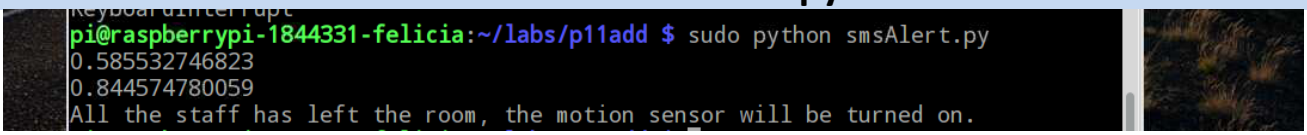
Result of detectedReko.py

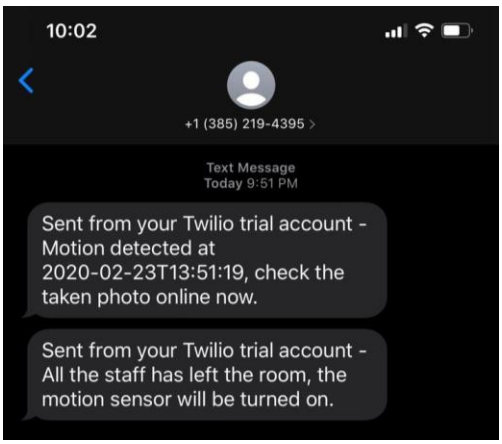






Result of smsAlert.py





Section 4
Source codes

All source codes, including Python, HTML files etc

server.py

```
from flask import Flask, render_template, jsonify,request

app = Flask(__name__)

import dynamodb
import jsonconverter as jsonc

@app.route("/api/getdata",methods=['POST','GET'])
def apidata_getdata():
   if request.method == 'POST' or request.method == 'GET':
      try:
         data = {'chart_data': jsonc.data_to_json(dynamodb.get_data_from_dynamodb()),
          'title': "IOT Data"}
         return jsonify(data)

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

@app.route("/")
def home():
   return render_template("index.html")


from gpiozero import LED
led = LED(18)

def ledOn():
  led.blink()
  return "On"

def ledOff():
  led.off()
  return "Off"

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

   if status == 'On':
     response = ledOn()
   else:
     response = ledOff()

   return response
```

```
app.run(debug=True,host="0.0.0.0")
```

index.html

```
<!doctype html>
<head>
   <style> #chartDiv {width:100%;}</style>
   <title>Google Charts with Flask</title>
   <script type="text/javascript" src="https://code.jquery.com/jquery-3.2.1.js"></script>
   <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
   <script type="text/javascript">
       google.charts.load('current', {'packages':['corechart','table']});
      // Set a callback to run when the Google Visualization API is loaded.
      google.charts.setOnLoadCallback(googlecharts_is_ready);

      var chart;
      var graphdata;

      function reset_status_messages(){
         $("#status").html("")
      }

      function googlecharts_is_ready(){
         $("#buttonloadchart").show()
         $("#buttonloadchart").click()
         $("#status").html("Google charts is ready")
      }

      function loadChart(){
            getData_and_drawChart()
      }

      function getData_and_drawChart(){
         getNewData()
      }


      function getNewData(){
         $("#status").html("Fetching data to plot graph...");

         jQuery.ajax({
            url: "/api/getdata" ,
            type: 'POST',
            error: function(jqXHR,textStatus, errorThrown ){
                console.log("Error while ajax:" + textStatus)
            },
```

```
        success: function(ndata, textStatus, xhr){
          //console.log(ndata)

          //console.log(ndata.chart_data)
          $("#status").html("Data fetched! Now plotting graph!");
          chartdata = ndata.chart_data
          graphdata = createDataTable(chartdata)
          drawLineChart(graphdata)
          drawDataTable(graphdata)
          $("#status").html("Graph plotted");
        }//end success
      });//end ajax
    } //end getNewData

  function createDataTable(newdata){
    graphdata = new google.visualization.DataTable();
    graphdata.addColumn('string', 'Time');
    graphdata.addColumn('number', 'Light');
    var newdata = JSON.parse(newdata);

    for (index=0;index<newdata.length;index++){

      datetime = (newdata[index].datetimeid)
      datetime = datetime.substring(0, 19) //+ "+0000"
      jsdatetime = new Date(Date.parse(datetime));
      jstime = jsdatetime.toLocaleTimeString();
      light = parseInt(newdata[index].value);
      graphdata.addRows([[jstime,light]]);
    }//end for
    return graphdata
  }

  function drawDataTable(graphdata){
    var table = new google.visualization.Table(document.getElementById('table_div'));
    table.draw(graphdata, {showRowNumber: true, width: '100%', height: '100%'});

  }//end drawTable

  function drawLineChart(graphdata) {
    chart = new google.visualization.LineChart(
    document.getElementById('chart_div'));
    chart.draw(graphdata, {legend: 'none', vAxis: {baseline: 0},
      colors: ['#A0D100']});
    return
  } //end drawChart

  $(document).ready(function(){
```

```
        reset_status_messages()

        setInterval(function () {
            loadChart()
        }, 3000);
    });

</script>

</head>
<body>
    <input id="buttonloadchart" type="button" onclick="loadChart()" value="Update graph">
    <div id="status"></div>
    <div id="chart_div" style="width:100%"></div>
    <div id="table_div" style="width:100%"></div>

</body>
</html>
```

detectedReko.py

```
from gpiozero import MotionSensor,Buzzer
import picamera
import telegram
import time
import boto3
import botocore
import json

# Sends a message to the chat
# Set the filename and bucket name
BUCKET = 'sp-p1844331-s3-bucket' # replace with your own unique bucket name
location = {'LocationConstraint': 'us-east-1'}
file_path = "."
file_name = "test.jpg"

# Connect to motion sensor and buzzer
pir = MotionSensor(26, sample_rate=5,queue_len=1)
bz = Buzzer(5)

# Connect to our bot
bot = telegram.Bot(token="778518288:AAEpuWpCgYH2YF3bK0CZDGg6tEvVjjiT43E")

# Sets the id for the active chat
chat_id=638292737

# Minimum time between captures
DELAY = 5
```

```
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-east-1"):
        rekognition = boto3.client("rekognition", region)
        response = rekognition.detect_labels(
                Image={
                        "S3Object": {
                                "Bucket": bucket,
                                "Name": key,
                        }
                },
                MaxLabels=max_labels,
                MinConfidence=min_confidence,
        )
        return response['Labels']


def detect_faces(bucket, key, max_labels=10, min_confidence=90, region="us-east-1"):
        rekognition = boto3.client("rekognition", region)
        response = rekognition.detect_faces(
                Image={
                        "S3Object": {
                                "Bucket": bucket,
                                "Name": key,
                        }
                },
                Attributes=['ALL']
        )
        return response['FaceDetails']
```

```
def takePhoto(file_path,file_name):
   with picamera.PiCamera() as camera:
      while True:
         full_path = file_path + "/" + file_name
         bz.off()
         pir.wait_for_motion()
         timestring = time.strftime("%Y-%m-%dT%H:%M:%S", time.gmtime())
         bz.on()
         print ("Taking photo " +timestring)
         camera.capture(full_path)
         bz.off()
         message = "Motion detected at " + timestring + ", taking photo..."
         bot.sendMessage(chat_id=chat_id, text=message)
         bot.sendPhoto(chat_id=chat_id, photo=open(full_path, 'rb'))
         uploadToS3(file_path,file_name, BUCKET,location)
         print('Detected faces for')
         for faceDetail in detect_faces(BUCKET, file_name):
            ageLow = faceDetail['AgeRange']['Low']
            ageHigh = faceDetail['AgeRange']['High']
            print('Age between {} and {} years old'.format(ageLow,ageHigh))
            print('Here are the other attributes:')
            print(json.dumps(faceDetail, indent=4, sort_keys=True))
         time.sleep(DELAY)


takePhoto(file_path, file_name)
```

card.py

```
import RPi.GPIO as GPIO
import MFRC522
import signal

uid = None
prev_uid = None
continue_reading = True

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print "Ctrl+C captured, ending read."
    continue_reading = False
    GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

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

# Welcome message
```

```
print "Welcome to the MFRC522 data read example"
print "Press Ctrl-C to stop."

# This loop keeps checking for chips.
# If one is near it will get the UID

while continue_reading:

    # Scan for cards
    (status,TagType) = mfrc522.MFRC522_Request(mfrc522.PICC_REQIDL)

    # If a card is found
    if status == mfrc522.MI_OK:
        # Get the UID of the card
        (status,uid) = mfrc522.MFRC522_Anticoll()
        if uid!=prev_uid:
           prev_uid = uid
           print("New card detected! UID of card is {}".format(uid))
```

smsAlert.py

```
from gpiozero import MCP3008
from time import sleep

from twilio.rest import Client

account_sid = "AC7d12b851012a4f9eb3858b26239678c5"
auth_token = "eabc0e5e668edbc2601dc1848c134cea"
client = Client(account_sid, auth_token)

my_hp = "+6588692298"
twilio_hp = "+13852194395"

adc = MCP3008(channel=0)

while True:
  light_value = adc.value
  print(light_value)
  if adc.value>0.8:
    sms = "All the staff has left the room, the motion sensor will be turned on."
    print(sms)
    message = client.api.account.messages.create(to=my_hp,
                                 from_=twilio_hp,
                                 body=sms)
    break
  else:
    sleep(3)
```

Section 5
Task List

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

Name of member	Part of project worked on	Contribution percentage
Jason Yeo	Website	50%
Renjie		50%
Jason Yeo	Security	30%
Felicia		70%
Jason	Database	100%

Section 6
Any other section you want to add

Delete this portion if you don't have additional sections

Section 7
References

References to online materials used

https://rpihome.blogspot.com/2016/06/send-photos-with-telegram.html

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