**SCHOOL OF COMPUTING (SOC)**

|  |  |
| --- | --- |
| **Date of Submission:** | 23rd Feburary 2020 |

|  |  |
| --- | --- |
| **Prepared for:** | Ms Dora Chua |

|  |  |
| --- | --- |
| **Class:** | DIT/FT/2A/34  DIT/FT/2B/34 |

|  |  |
| --- | --- |
| **Submitted by:** | Alvin Tan Liang Yee  Patriciab Chan Shi Hui |

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| 1650688 | Alvin Tan Liang Yee |
| 1703333 | Patriciab Chan Shi Hui |

**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)**

**2017/2018 Semester 1**

**Table of Contents**

[Section 1 Overview of project 2](#_Toc33479748)

[A. Where we have uploaded our tutorial 2](#_Toc33479749)

[B. What is the application about? 2](#_Toc33479750)

[C. How does the final RPI set-up looks like? 3](#_Toc33479751)

[D. How does the web or mobile application look like? 4](#_Toc33479752)

[E. System architecture of our system 5](#_Toc33479753)

[F. Evidence that we have met basic requirements 5](#_Toc33479754)

[G. Bonus features on top of basic requirements 6](#_Toc33479755)

[A. Quick-start guide (Readme first) 6](#_Toc33479756)

[Section 2 Hardware requirements 7](#_Toc33479757)

[Hardware checklist 7](#_Toc33479758)

[Hardware setup instructions 7](#_Toc33479759)

[Fritzing Diagram 7](#_Toc33479760)

[Section 3 Software Requirements 8](#_Toc33479761)

[Software checklist 8](#_Toc33479762)

[Software setup instructions 8](#_Toc33479763)

[Section 4 Source codes 9](#_Toc33479764)

[server.py 9](#_Toc33479765)

[CA2IOT.py 13](#_Toc33479766)

[aws.py 15](#_Toc33479767)

[dynamodb.py 16](#_Toc33479768)

[dynamodbfanlog.py 17](#_Toc33479769)

[dynamodbtemperature.py 18](#_Toc33479770)

[jsonconverter.py 19](#_Toc33479771)

[index.html 19](#_Toc33479772)

[Section 5 Task List 25](#_Toc33479773)

[Section 7 References 25](#_Toc33479774)

# Section 1 Overview of project

* 1. 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>

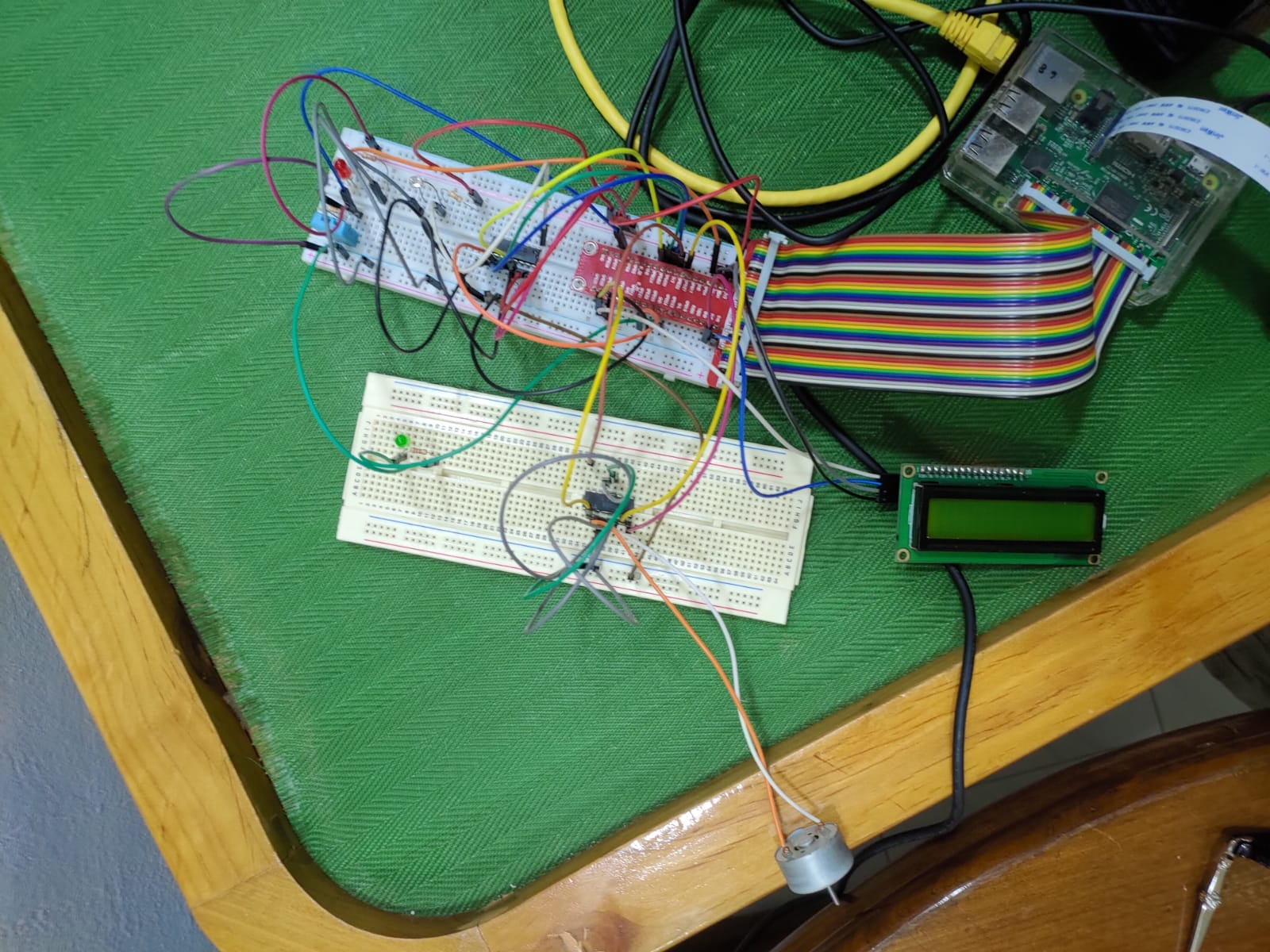
|  |  |
| --- | --- |
| **Youtube** | <https://www.youtube.com/watch?v=pMftiNG6bhg> |
| **Public tutorial link** | <https://github.com/PatriciabChan/IoTProject> |

* 1. What is the application about?

Our Project is called the HomeSystem which helps monitor the Home.

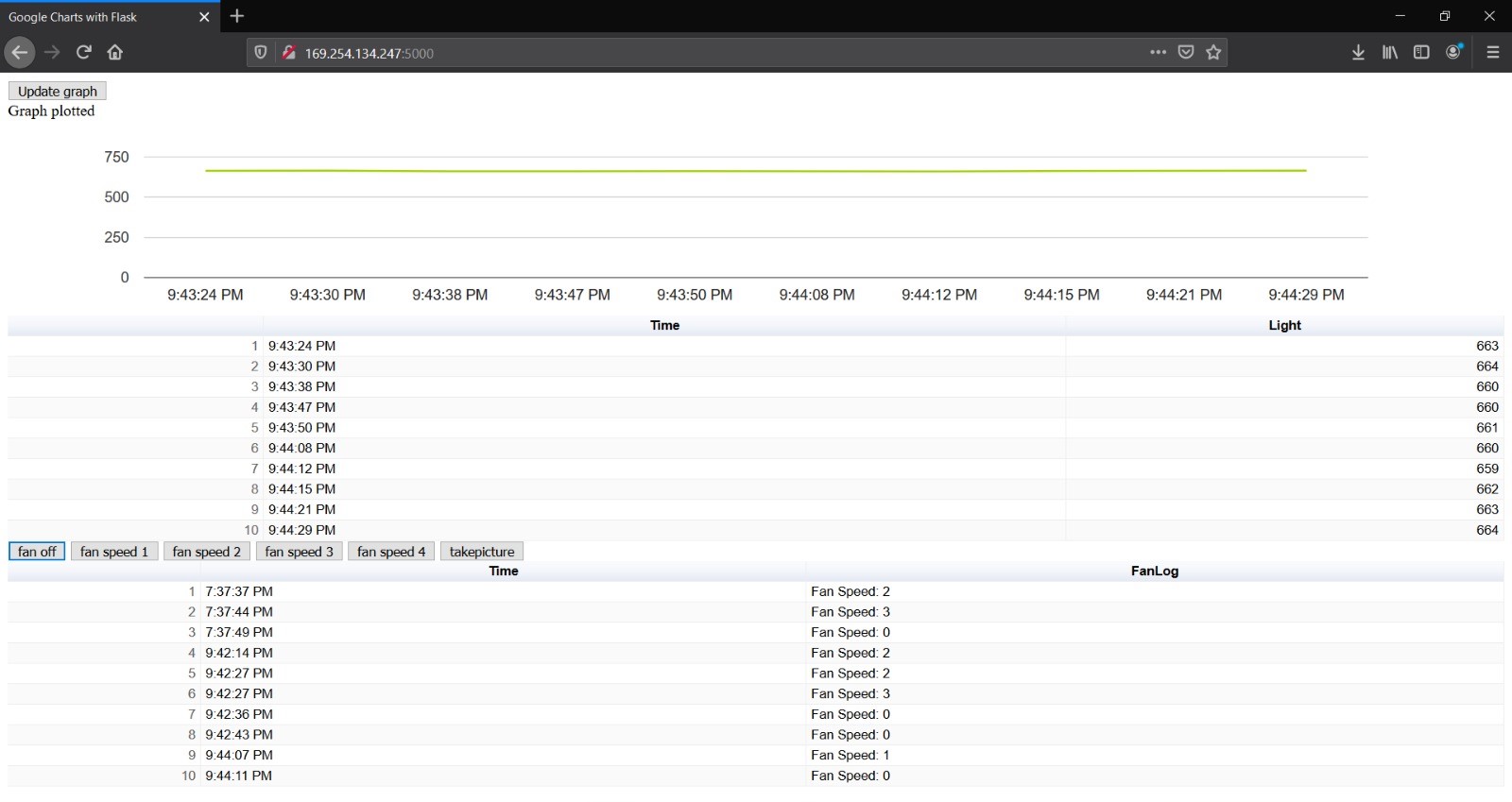
It monitors the Fan and Light System in the house, and with the user being able to see its status on the Web Interface.

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



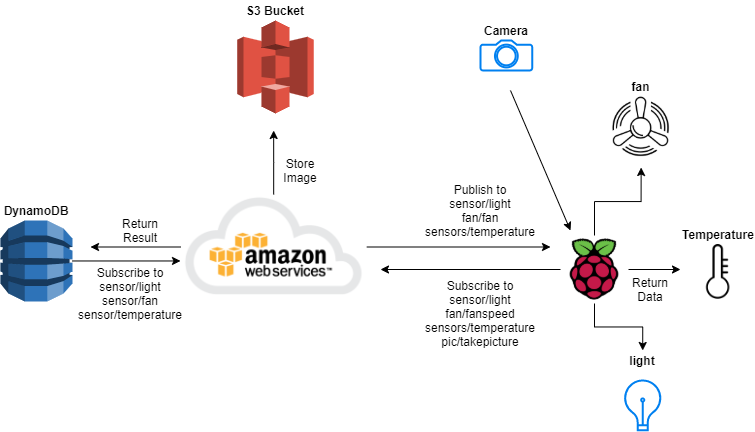
* 1. How does the web or mobile application look like?

Main Page where the data is shown.



* 1. System architecture of our system

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



* 1. Evidence that we have met basic requirements

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

|  |  |
| --- | --- |
| Requirement | Evidence |
| Used three sensors | Used sensors :  1 Light Sensor  1 Temperature Sensor  Motor |
| Used MQTT | Our MQTT endpoint -->  Example of data sent through MQTT :  Sensors/temperature  Sensors/light  Fan/fan |
| Stored data in cloud | Stored Light Data in AWS DynamoDB  Stored Fan Speed Data in AWS DynamoDB  Stored Temperature Data in AWS DynamoDB  Stored Images using AWS S3 |
| Used cloud service | AWS S3 to store images  DynamoDB to store data |
| Provide real-time sensor value / status | Show the real-time value of sensors :   * Light * Temperature * Fan Speedactu |
| Provide historical sensor value/ status | Show historical value of sensors :   * Light * Temperature * Fan Speed |
| Control actuator\*\*\*\* | Placed button on webpage to control actuator |

* 1. Bonus features on top of basic requirements

Provide bullet list of the bonus features you have added on top of basic requirements

-

* 1. 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 C
2. Login to aws to create tables,rules and roles under IOTCore
3. Create 3 tables under DynamoDB for light sensor,temperature sensor and fanlog
4. Create a unique bucket under aws S3
5. Run CA2IOT.py
6. Run server.py
7. Open webpage of the server.py <ipaddress of pi>:5000 in webbrowser

okiee

# Section 2 Hardware requirements

Hardware checklist

|  |  |
| --- | --- |
| Item \*\* | Quantity |
| Raspberry Pi | 1 |
| DHT11 Temperature & Humidity Sensor | 1 |
| LCD Display | 1 |
| Motor | 1 |
| L293D IC Chip | 1 |
| MCP3008 | 1 |
| Light sensor | 1 |
| 10k resistors | 1 |

Hardware setup instructions

Follow the reference for the motor connection and connect the bottom row of the diagram only, yellow jumper wires to pin 20 and 22 respectively. From L293D to raspberri pi

Connect the DHT11 to pin 19

Connect motor to L293D

Connect the light sensor and resistor to MCP3008

From MCP3008 connect it to raspberri pi

Connect LED to pin 24

Connect LCD panel

There should be a Pi camera connected already otherwise connect a raspberry pi camera.

Fritzing Diagram

Paste a Fritzing diagram of your setup here

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

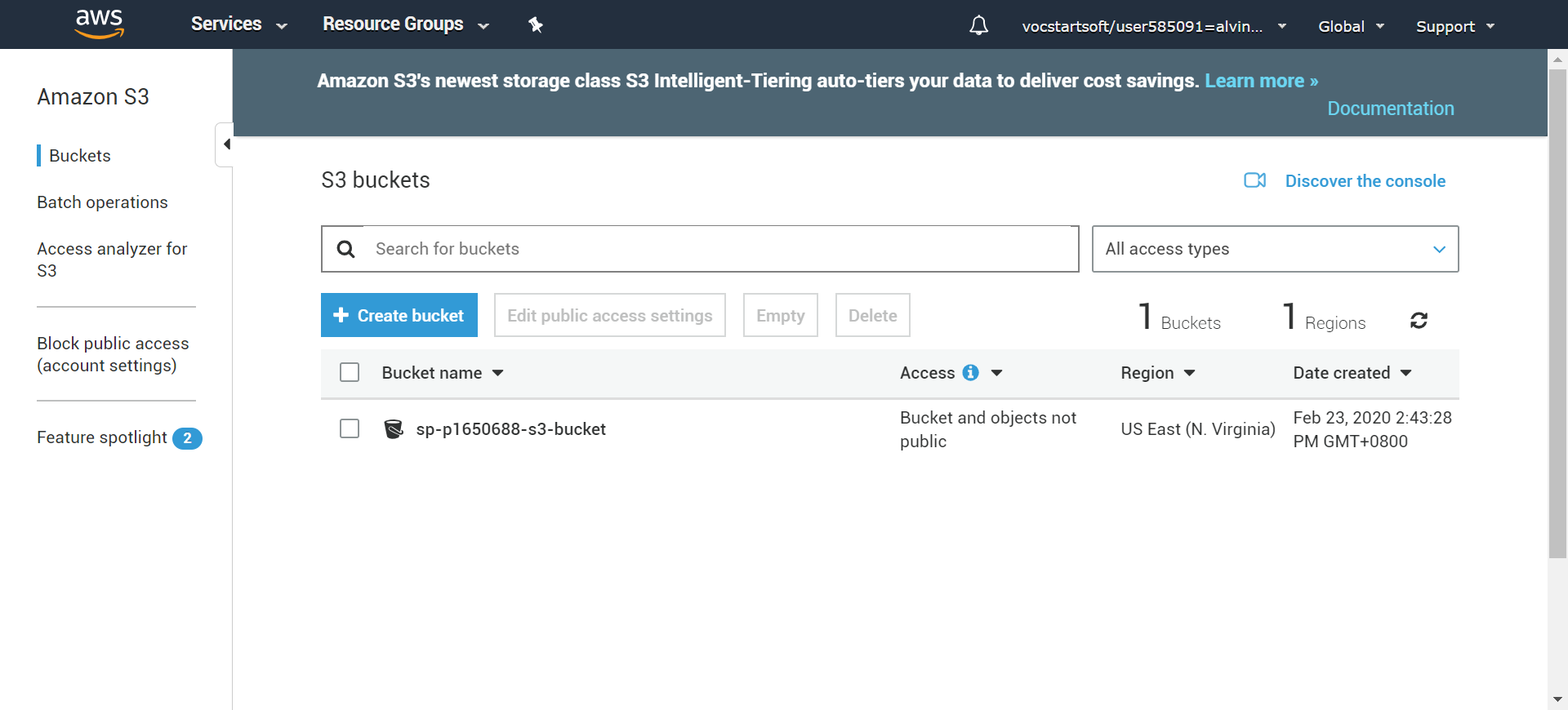
# Section 3 Software Requirements

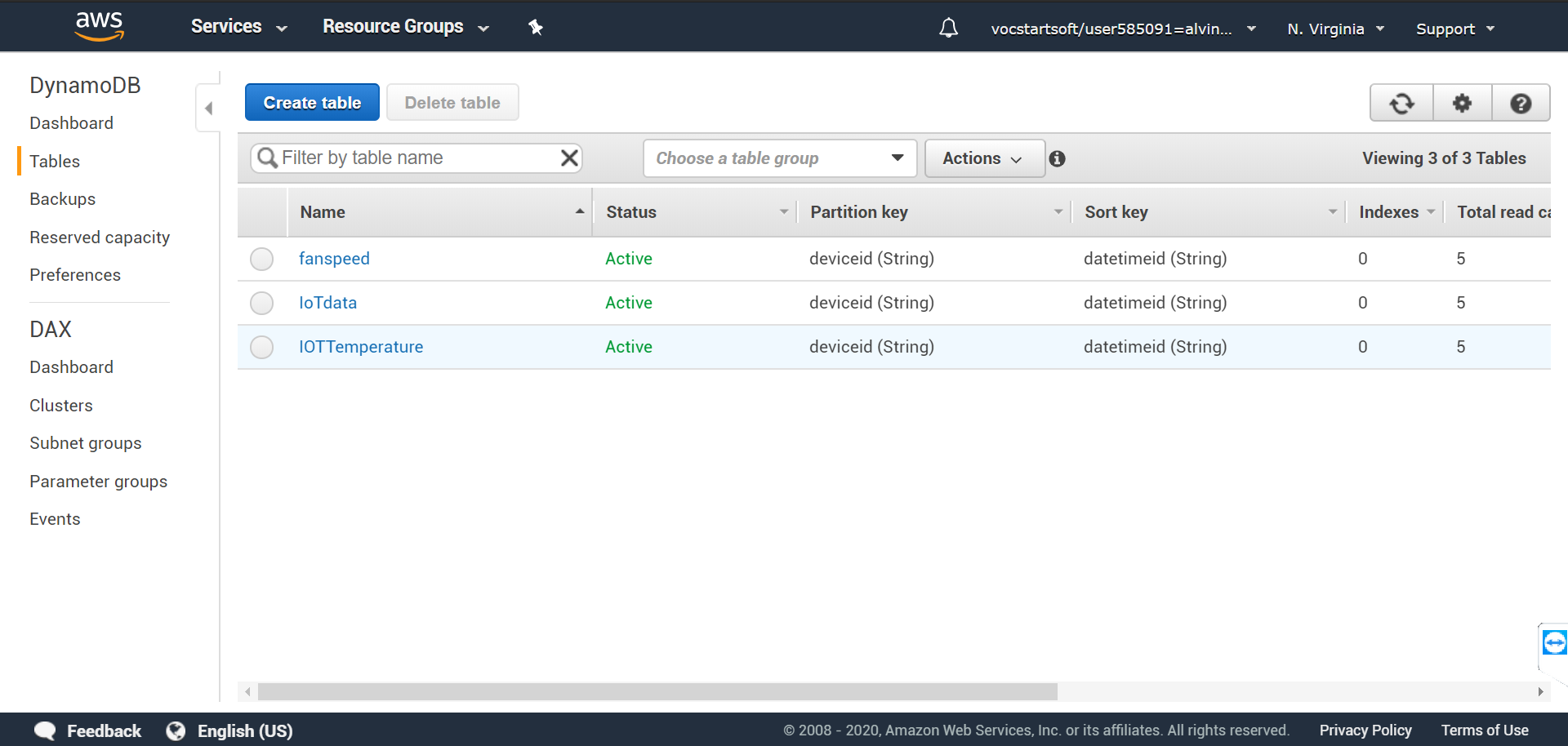
Software checklist

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

1. AWSIoTPythonSDK.MQTTLib library
2. gpiozero library
3. boto3 library
4. botocore library
5. json library
6. flash library

Software setup instructions





Most libraries are pre-installed. Only botocore needs to be created.

Do these 2 commands

Sudo pip install botocore

Sudo pip install awscli

Sudo pip install awscli –upgrade

Sudo pip install boto3 –- upgrade

Sudo pip install botocore –upgrade

Download the Security keys upon creation.

Run CA2IOT.py

# Section 4 Source codes

All source codes, including Python, HTML files etc

### server.py

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

from flask import Flask, render\_template, jsonify, request,Response

from gpiozero import Motor, Buzzer, LED, InputDevice, Button, MCP3008

from rpi\_lcd import LCD

from time import sleep

import mysql.connector

import sys

import json

import numpy

import datetime

import decimal

import gevent

import gevent.monkey

from gevent.pywsgi import WSGIServer

from picamera import PiCamera

import time, datetime

import boto3

import botocore

s3 = boto3.resource('s3')

from rpi\_lcd import LCD

app = Flask(\_\_name\_\_)

import dynamodb

import dynamodbfanlog

import dynamodbtemperature

import jsonconverter as jsonc

# 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")

lcd = LCD()

motor2 = Motor(20,22, pwm = True)

full\_path = '/home/pi/Desktop/image1.jpg'

file\_name = 'image1.jpg'

host = "a3c16s480tb79n-ats.iot.us-east-1.amazonaws.com"

rootCAPath = "rootca.pem"

certificatePath = "certificate.pem.crt"

privateKeyPath = "private.pem.key"

BUCKET = 'sp-p1650688-s3-bucket' # replace with your own unique bucket name

location = {'LocationConstraint': 'us-east-1'}

file\_path = "/home/pi/Desktop"

file\_name = "test.jpg"

best\_bet\_item = "Unknown"

my\_rpi = AWSIoTMQTTClient("PubSub-p1650688")

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

# Connect and subscribe to AWS IoT

my\_rpi.connect()

my\_rpi.subscribe("sensors/light", 1, customCallback)

my\_rpi.subscribe("sensors/temperature", 1, customCallback)

sleep(2)

def takepicture():

        takePhoto(file\_path, file\_name)

        bucket = 'sp-p1650688-s3-bucket' # replace with your own unique bucket name

        exists = True

        s3.meta.client.head\_bucket(Bucket=bucket)

        s3.Object(bucket, file\_name).put(Body=open(full\_path, 'rb'))

        print("File uploaded")

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)

@app.route("/takepicture/picture")

def takepictureroute():

    takepicture()

    print"test"

    return 'ok'

global currentfanspeed

@app.route("/writeMotor0/motoroff")

def fanoff():

    currentfanspeed = 0

    motor2.stop()

    uploadfanstatus("Fan Speed: " + str(currentfanspeed))

    return'ok'

@app.route("/writeMotor1/motorspeed1")

def fanspeed1():

    motor2.backward(speed=0.25)

    currentfanspeed = 1

    uploadfanstatus("Fan Speed: " +str(currentfanspeed))

    return'ok'

@app.route("/writeMotor2/motorspeed2")

def fanspeed2():

    motor2.backward(speed=0.50)

    currentfanspeed = 2

    uploadfanstatus("Fan Speed: " +  str(currentfanspeed))

    return'ok'

@app.route("/writeMotor3/motorspeed3")

def fanspeed3():

    motor2.backward(speed=0.75)

    currentfanspeed = 3

    uploadfanstatus("Fan Speed: " + str(currentfanspeed))

    return'ok'

@app.route("/writeMotor4/motorspeed4")

def fanspeed4():

    motor2.backward(speed=1)

    currentfanspeed = 4

    uploadfanstatus("Fan Speed: " + str(currentfanspeed))

    return'ok'

@app.route("/writeMotor5/motorauto")

def fanspeedauto():

    print("test")

def uploadfanstatus(speedvalue):

    message = {}

    message["deviceid"] = "deviceid\_Alvin\_pat"

    import datetime as datetime

    now = datetime.datetime.now()

    message["datetimeid"] = now.isoformat()

    message["value"] = speedvalue

    import json

    my\_rpi.publish("fan/fanspeed", json.dumps(message), 1)

@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("/api/getfandata",methods=['POST','GET'])

def apidata\_getfandata():

    if request.method == 'POST' or request.method == 'GET':

        try:

            data = {'chart\_data': jsonc.data\_to\_json(dynamodbfanlog.get\_data\_from\_dynamodb()),

             'title': "Fan Logs"}

            return jsonify(data)

        except:

            import sys

            print(sys.exc\_info()[0])

            print(sys.exc\_info()[1])

@app.route("/api/gettemperature",methods=['POST','GET'])

def apidata\_gettemperature():

    if request.method == 'POST' or request.method == 'GET':

        try:

            data = {'chart\_data': jsonc.data\_to\_json(dynamodbtemperature.get\_data\_from\_dynamodb()),

             'title': "temperate Logs"}

            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")

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

### CA2IOT.py

# Import SDK packages

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

from time import sleep

from gpiozero import Motor, Buzzer, LED, InputDevice, Button, MCP3008

from signal import pause

import sys

import Adafruit\_DHT

from rpi\_lcd import LCD

import time, datetime

import boto3

import botocore

s3 = boto3.resource('s3')

motor2 = Motor(20,22, pwm = True)

adc = MCP3008(channel=0)

temppin = 19

lcd = LCD()

led = LED(24)

full\_path = '/home/pi/Desktop/image1.jpg'

file\_name = 'image1.jpg'

# 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")

def lighttemperature():

    humidity, temperature = Adafruit\_DHT.read\_retry(11, temppin)

    print('Temp: {:.1f} C'.format(temperature))

    print('Humidity: {:.1f}'.format(humidity))

    return temperature

def displaytime(hour2,minutes2):

    hour\_as\_string = str(hour2)

    h1\_as\_string = str(minutes2)

    lcd.text(hour\_as\_string, 1)

    lcd.text(h1\_as\_string, 2)

host = "a3c16s480tb79n-ats.iot.us-east-1.amazonaws.com"

rootCAPath = "rootca.pem"

certificatePath = "certificate.pem.crt"

privateKeyPath = "private.pem.key"

my\_rpi = AWSIoTMQTTClient("PubSub-p1812345")

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

# Connect and subscribe to AWS IoT

my\_rpi.connect()

my\_rpi.subscribe("sensors/light", 1, customCallback)

my\_rpi.subscribe("sensors/temperature", 1, customCallback)

sleep(2)

# Publish to the same topic in a loop forever

loopCount = 0

while True:

    now = datetime.datetime.now()

    hour = now.hour

    minute = now.minute

    h1 = hour/10

    h2 = hour % 10

    m1 = minute /10

    m2 = minute % 10

    displaytime(hour,minute)

    light = round(1024-(adc.value\*1024))

    temperature2 = lighttemperature()

    message = {}

    message["deviceid"] = "deviceid\_Alvin\_pat"

    import datetime as datetime

    now = datetime.datetime.now()

    message["datetimeid"] = now.isoformat()

    message["value"] = light

    import json

    my\_rpi.publish("sensors/light", json.dumps(message), 1)

    message2 = {}

    message2["deviceid"] = "deviceid\_Alvin\_pat"

    now = datetime.datetime.now()

    message2["datetimeid"] = now.isoformat()

    message2["value"] = temperature2

    my\_rpi.publish("sensors/temperature", json.dumps(message2), 1)

sleep(5)

### aws.py

# Import SDK packages

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

from time import sleep

from gpiozero import MCP3008

import sys

import Adafruit\_DHT

from time import sleep

adc = MCP3008(channel=0)

pin = 19

# 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")

host = "a3c16s480tb79n-ats.iot.us-east-1.amazonaws.com"

rootCAPath = "rootca.pem"

certificatePath = "certificate.pem.crt"

privateKeyPath = "private.pem.key"

my\_rpi = AWSIoTMQTTClient("P1650688")

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

# Connect and subscribe to AWS IoT

my\_rpi.connect()

my\_rpi.subscribe("sensors/light", 1, customCallback)

sleep(2)

# Publish to the same topic in a loop forever

loopCount = 0

while True:

      light = round(1024-(adc.value\*1024))

      loopCount = loopCount+1

      message = {}

      message["deviceid"] = "deviceid\_dorachua"

      import datetime as datetime

      now = datetime.datetime.now()

      message["datetimeid"] = now.isoformat()

      message["value"] = light

      import json

      my\_rpi.publish("sensors/light", json.dumps(message), 1)

      humidity, temperature = Adafruit\_DHT.read\_retry(11, pin)

      print('Temp: {:.1f} C'.format(temperature))

      print('Humidity: {:.1f}'.format(humidity))

      sleep(5)

### dynamodb.py

def get\_data\_from\_dynamodb():

    try:

            import boto3

            from boto3.dynamodb.conditions import Key, Attr

            dynamodb = boto3.resource('dynamodb', region\_name='us-east-1')

            table = dynamodb.Table('IoTdata')

            startdate = '2020-02'

            response = table.query(

                KeyConditionExpression=Key('deviceid').eq('deviceid\_Alvin\_pat')

                                      & Key('datetimeid').begins\_with(startdate),

                ScanIndexForward=False

            )

            items = response['Items']

            n=10 # limit to last 10 items

            data = items[:n]

            data\_reversed = data[::-1]

            return data\_reversed

    except:

        import sys

        print(sys.exc\_info()[0])

        print(sys.exc\_info()[1])

if \_\_name\_\_ == "\_\_main\_\_":

    get\_data\_from\_dynamodb()

### dynamodbfanlog.py

def get\_data\_from\_dynamodb():

    try:

            import boto3

            from boto3.dynamodb.conditions import Key, Attr

            dynamodb = boto3.resource('dynamodb', region\_name='us-east-1')

            table = dynamodb.Table('fanspeed')

            startdate = '2020-02'

            response = table.query(

                KeyConditionExpression=Key('deviceid').eq('deviceid\_Alvin\_pat')

                                      & Key('datetimeid').begins\_with(startdate),

                ScanIndexForward=False

            )

            items = response['Items']

            n=10 # limit to last 10 items

            data = items[:n]

            data\_reversed = data[::-1]

            return data\_reversed

    except:

        import sys

        print(sys.exc\_info()[0])

        print(sys.exc\_info()[1])

if \_\_name\_\_ == "\_\_main\_\_":

    get\_data\_from\_dynamodb()

### dynamodbtemperature.py

def get\_data\_from\_dynamodb():

    try:

            import boto3

            from boto3.dynamodb.conditions import Key, Attr

            dynamodb = boto3.resource('dynamodb', region\_name='us-east-1')

            table = dynamodb.Table('IoTdata')

            startdate = '2020-02'

            response = table.query(

                KeyConditionExpression=Key('deviceid').eq('deviceid\_Alvin\_pat')

                                      & Key('datetimeid').begins\_with(startdate),

                ScanIndexForward=False

            )

            items = response['Items']

            n=10 # limit to last 10 items

            data = items[:n]

            data\_reversed = data[::-1]

            return data\_reversed

    except:

        import sys

        print(sys.exc\_info()[0])

        print(sys.exc\_info()[1])

if \_\_name\_\_ == "\_\_main\_\_":

    get\_data\_from\_dynamodb()

### jsonconverter.py

from decimal import Decimal

import json

import datetime

import numpy

class GenericEncoder(json.JSONEncoder):

    def default(self, obj):

        if isinstance(obj, numpy.generic):

            return numpy.asscalar(obj)

        elif isinstance(obj, Decimal):

            return str(obj)

        elif isinstance(obj, datetime.datetime):

            return obj.strftime('%Y-%m-%d %H:%M:%S')

        elif isinstance(obj, Decimal):

            return float(obj)

        else:

            return json.JSONEncoder.default(self, obj)

def data\_to\_json(data):

    json\_data = json.dumps(data,cls=GenericEncoder)

    print(json\_data)

    return json\_data

### 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 chart2;

        var graphdata;

        var graphdata2;

        function reset\_status\_messages(){

            $("#status").html("")

            $("#status2").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 getNewData2(){

            $("#status2").html("Fetching data to plot graph...");

            jQuery.ajax({

                url: "/api/getfandata" ,

                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)

                    chartdata2 = ndata.chart\_data

                    graphdata2 = createDataTable2(chartdata2)

                    drawDataTable2(graphdata2)

                }//end success

            });//end ajax

          } //end getNewData

        function turnofffan() {

            $.ajax({

                url: "/writeMotor0/motoroff",

                success: function (result) {

                }

            })

        }

        function fanspeed1() {

            $.ajax({

                url: "/writeMotor1/motorspeed1",

                success: function (result){

                }

            })

        }

        function fanspeed2() {

            $.ajax({

                url: "/writeMotor2/motorspeed2",

                success: function (result){

                }

            })

        }

        function fanspeed3() {

            $.ajax({

                url: "/writeMotor3/motorspeed3",

                success: function (result){

                }

            })

        }

        function takepicture()

        {

            $.ajax({

                url: "/takepicture/picture",

                success: function (result){

                }

            })

        }

        function fanspeed4() {

            $.ajax({

                url: "/writeMotor4/motorspeed4",

                success: function (result){

                }

            })

        }

        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 createDataTable2(newdata2){

            graphdata2 = new google.visualization.DataTable();

            graphdata2.addColumn('string', 'Time');

            graphdata2.addColumn('string', 'FanLog');

            var newdata2 = JSON.parse(newdata2);

            for (index=0;index<newdata2.length;index++){

                datetime2 = (newdata2[index].datetimeid)

                datetime2 = datetime2.substring(0, 19) //+ "+0000"

                jsdatetime2 = new Date(Date.parse(datetime2));

                jstime2 = jsdatetime2.toLocaleTimeString();

                fanlog = newdata2[index].value;

                graphdata2.addRows([[jstime2,fanlog]]);

            }//end for

            return graphdata2

        }

        function drawDataTable2(graphdata2){

            var table2 = new google.visualization.Table(document.getElementById('table\_div2'));

            table2.draw(graphdata2, {showRowNumber: true, width: '100%', height: '100%'});

        }//end drawTable

        function drawDataTable(graphdata2){

            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()

                getNewData2()

            }, 10000);

        });

</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>

        <input id="fanspeedoff" type="button" onclick="turnofffan()" value="fan off">

        <input id="fanspeed1" type="button" onclick="fanspeed1()" value="fan speed 1">

        <input id="fanspeed2" type="button" onclick="fanspeed2()" value="fan speed 2">

        <input id="fanspeed3" type="button" onclick="fanspeed3()" value="fan speed 3">

        <input id="fanspeed4" type="button" onclick="fanspeed4()" value="fan speed 4">

        <input id="takepicture" type="button" onclick="takepicture()" value="takepicture">

        <div id="table\_div2" style="width:100%"></div>

</body>

</html>

# 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 |
| Alvin Tan Liang Yee | HTML  server.py  CA2IOT.py  dynamodb.py  dynamofanlog.py  dynamotemperature.py  jsonconverter.py  Documentation  Creation of AWS DynamoDB  Creation of AWS S3  Hardware Setup | 60% |
| Patriciab Chan Shi Hui | HTML  server.py  CA2IOT.py  dynamodb.py  dynamofanlog.py  dynamotemperature.py  jsonconverter.py  Documentation  Hardware Setup | 40% |

# Section 7 References

https://medium.com/@Keithweaver\_/controlling-dc-motors-using-python-with-a-raspberry-pi-40-pin-f6fa891dc3d

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