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 February 2020

Prepared for: Dora Chua

Class: DIT/FT/2B34

Submitted by:

Student ID	Name
1844638	Jiang Lei
1846726	Lim Zi Yun
1851591	Seah Sock Wen Jorin

Table of Contents

Section 1
Overview of project

A.Where we have uploaded our tutorial

http://bit.ly/1910s2iotca2

Youtube	https://youtu.be/CSHWzWKVpXg
Public tutorial link	https://github.com/leileijng/SmartGarden_IOTCA2

B. What is the application about?

Our project is called Smart Garden, which aims to help people to take care of their plants.

This project is able to monitor the temperature, humidity, soil moisture and light level of the plant. The real-time data will be displayed on the LCD screen. The user can also tap his valid NFC card to receive a whatsApp message with these values.

When there is not enough light, the plants cannot photosynthesis. In order to resolve such problem, a portable LED light is implemented that when the light level is too low, it will be auto on. At the same time, the user is able to receive an alert message through the telegram.

In order to make sure the plant has enough moisture, a water pump has been set up. It has two systems: automated system and manual system. When the automated system is on, the system detects the soil moisture of the plant, and then decide whether to water the plant or not. And the user can also water the plant manually by switching to the manual control and on or off the pump on the website. Besides, whenever the plant is too dry (soil moisture < 500), the user will receive alerting email.

The user can also remotely view images of their plants. There are 2 cameras set up, one is for taking picture of the plant, the other one is for flower. The user can send a telegram message to request a picture for either plant or flower, then the system would check the command and decide which camera should be activated. And a picture of their plants or flowers will be sent to the user as well as uploaded to Twitter.

Besides taking picture, the system can also detect whether there is a bug on the plants/ flowers. Once the telegram command “detect” is received, all the cameras would be activated and capture the photos for 5 successive times. The system would scan and recognize the objects in the pictures. If a bug is detected, the image would be sent back to the user with an alerting message on the

telegram. At the same time, the image would also be uploaded to S3 storage, and then be displayed on the website.

The objective of this application is to help maintain an ideal and consistent light and soil condition for the plant, and provide convenience for people who have to go for business trip often and tend to forget to water their plants regularly. And the users are allowed to remotely view and take care of their plants with our features “taking pictures of whichever you want”, and detect the bugs. With this application, the user doesn’t need to take much care of their plants and at the same time they can easily visualise the conditions of their plants remotely through the web application.

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

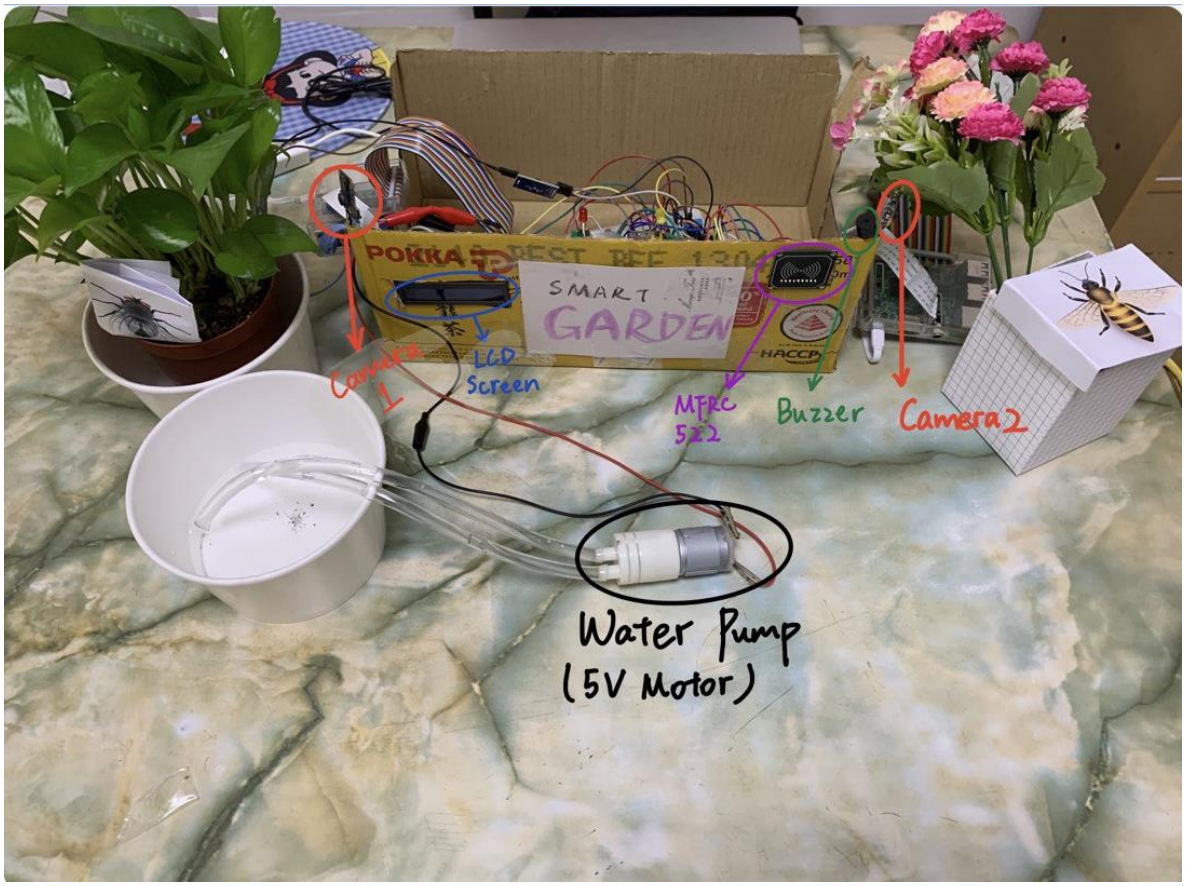


Figure 1: Hardware Setup - (1A)

Figure 1: Hardware Setup (1B)

D. How does the web application look like?

1. Login Page:

The user must log in before accessing the application. It ensures the security of the application. (Username: u1; Password: 111). Without login, even the url is for index page, the user will still be redirected to login page to verify his/her credentials.

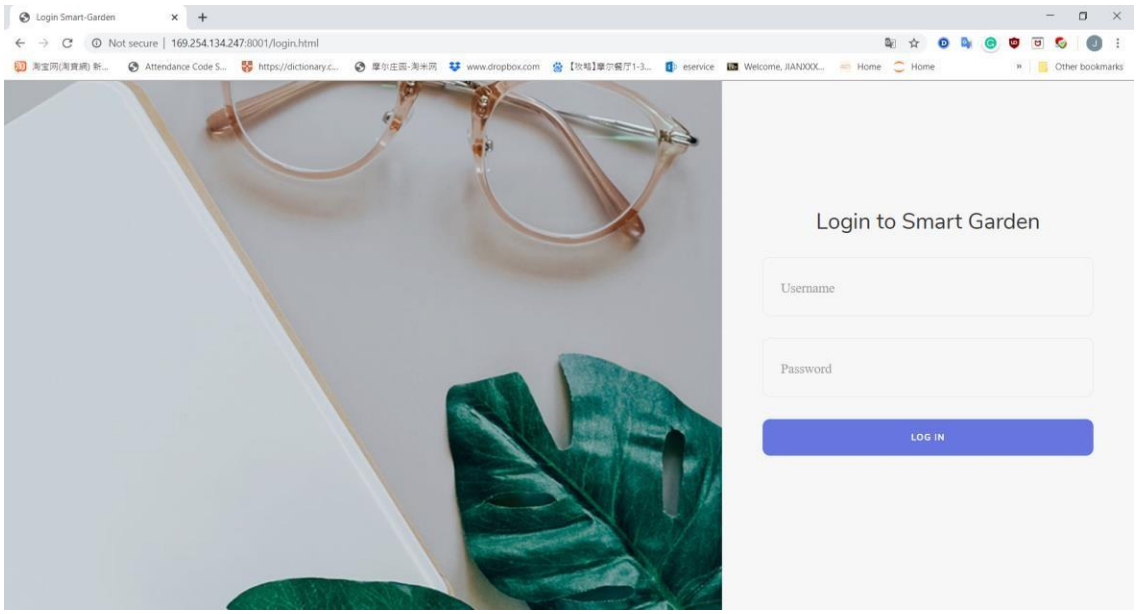


Figure 2: Web Login page (1C.1)

2. Index Page:

This page shows the current data collected from sensors, including temperature, humidity, soil moisture and light level. It also allows users to control the status of water pump, either automated or manual. The auto status enables the system to detect the soil moisture of plant and then decide whether to on/off the pump. Once the user switch off the automated system, it will enable the user to manually control the water pump.

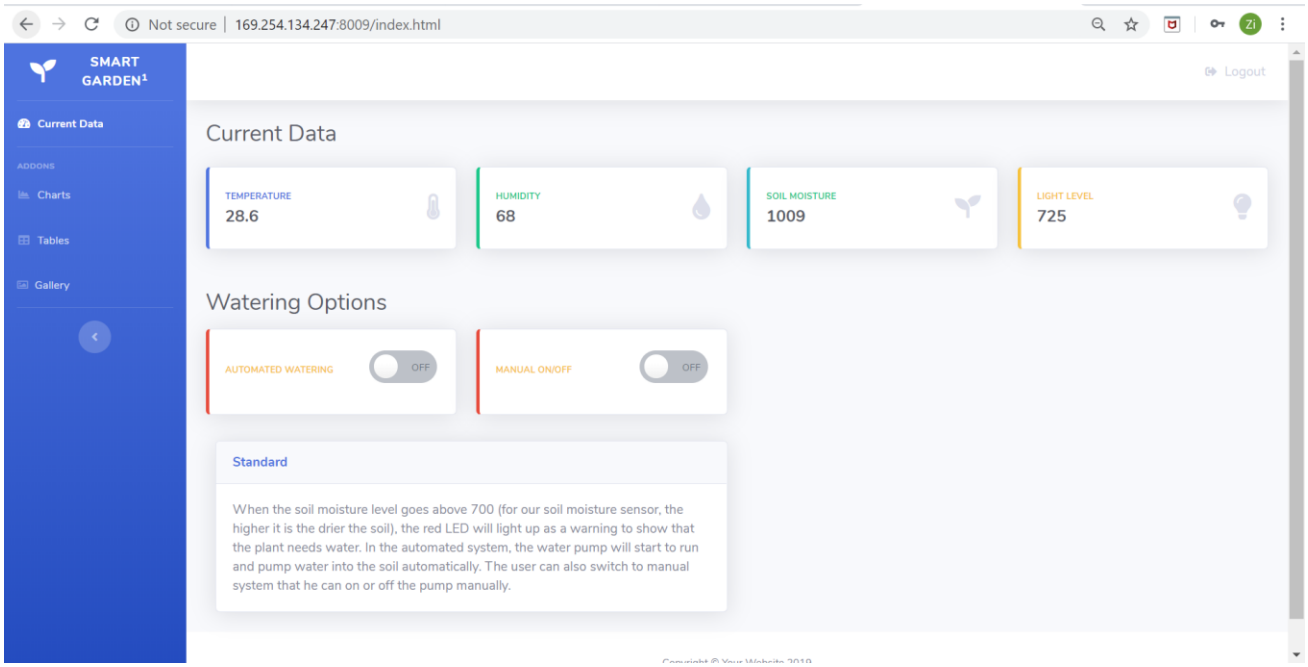
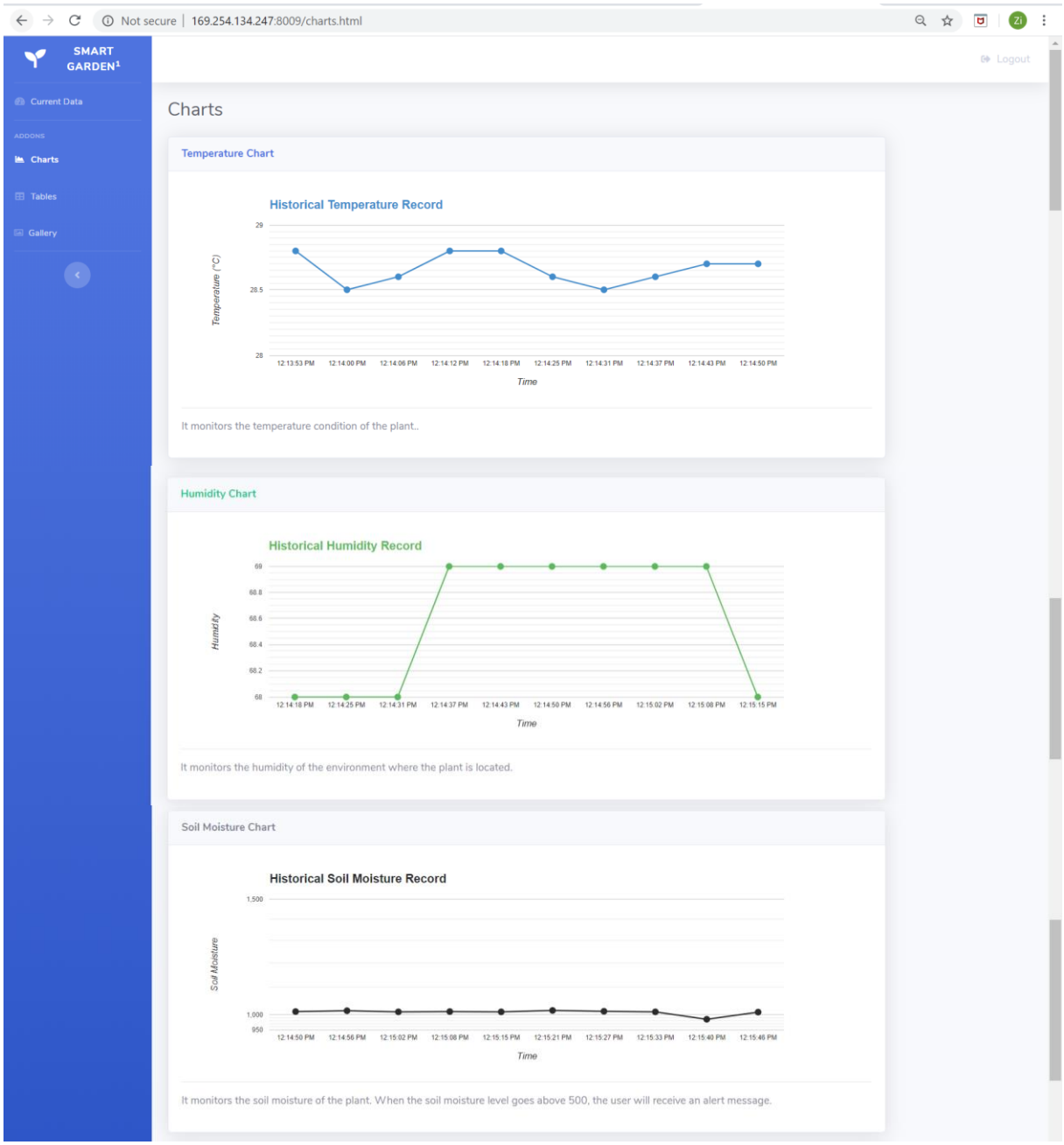


Figure 3: Web Index page (1C.2)

3. Chart Page

This page shows the historical data (10 records) collected from sensors by extracting from MySQL database. It uses google chart to draw the line chart. There are 4 charts, temperature, humidity, soil moisture and light level. With these charts, the user can visualise the conditions of their plants.



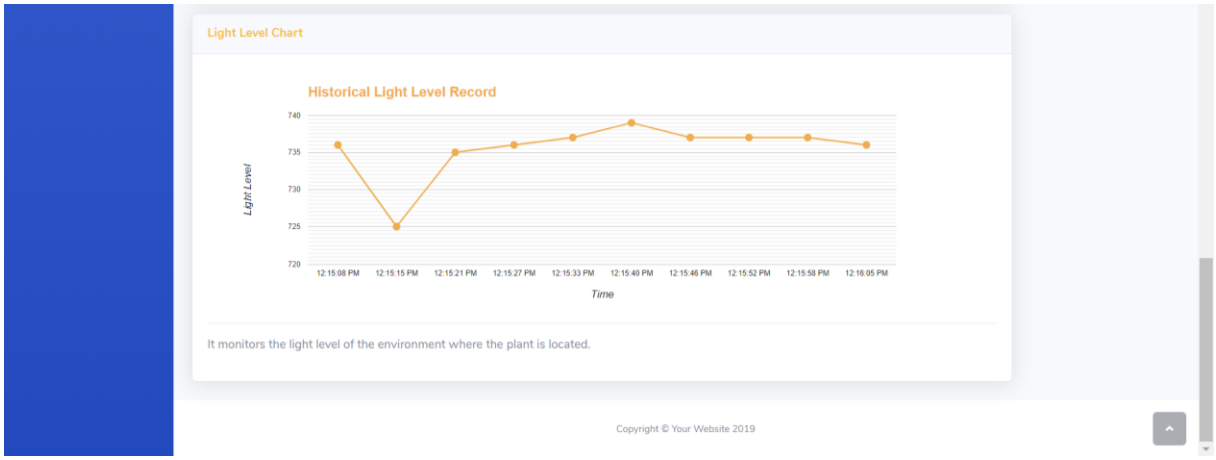


Figure 4: Web Chart page (1C.3)

4. Table Page

This page shows the historical data in the table format using google data table. It allows the user to view all 4 conditions of plant in one table and sort the data by column. It also keeps track of the status of pump, thus the user can know better about their behaviours, as shown in 2nd table.

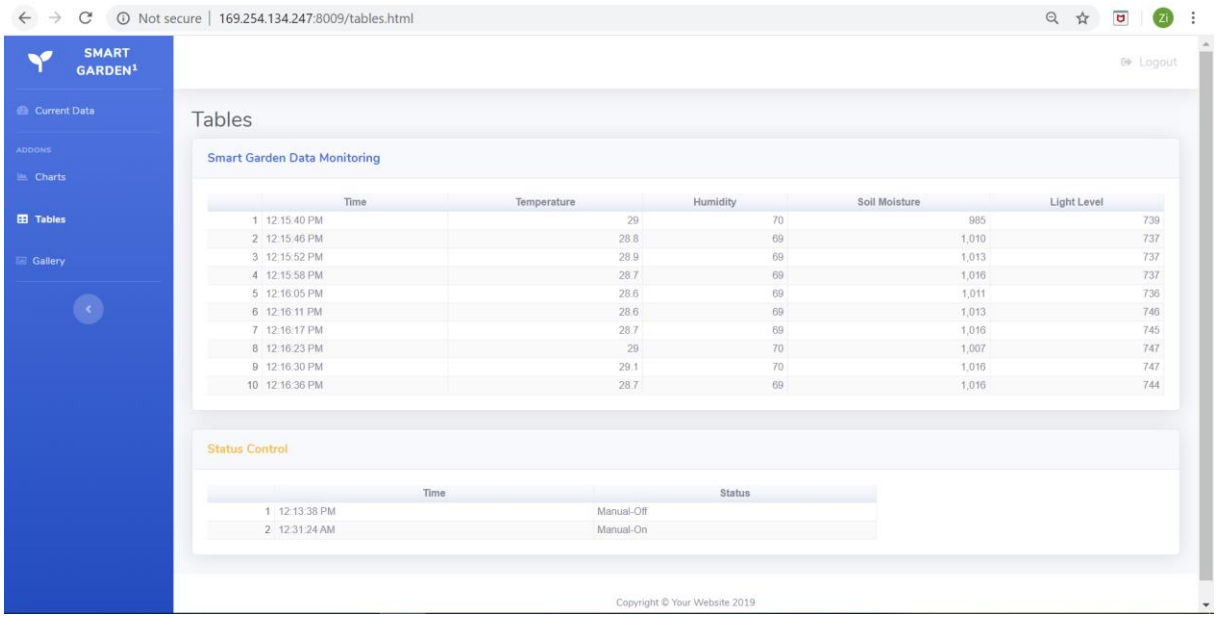


Figure 5: Web Table page(1C.4)

5. Gallery Page

This page shows all the images taken by the 2 piCams whenever there is a bug detected on the plant or flower, when the user sends 'detect' command on the telegram (see 6. Gallery Page – Bugs on Plant (1C.5)). This page displays the bug images, with info about who initiated the detecting process and the detected date and time. This page allows user to keep track of when it happens (date and time) and where it happens (the plant or flower that was detected with bugs).

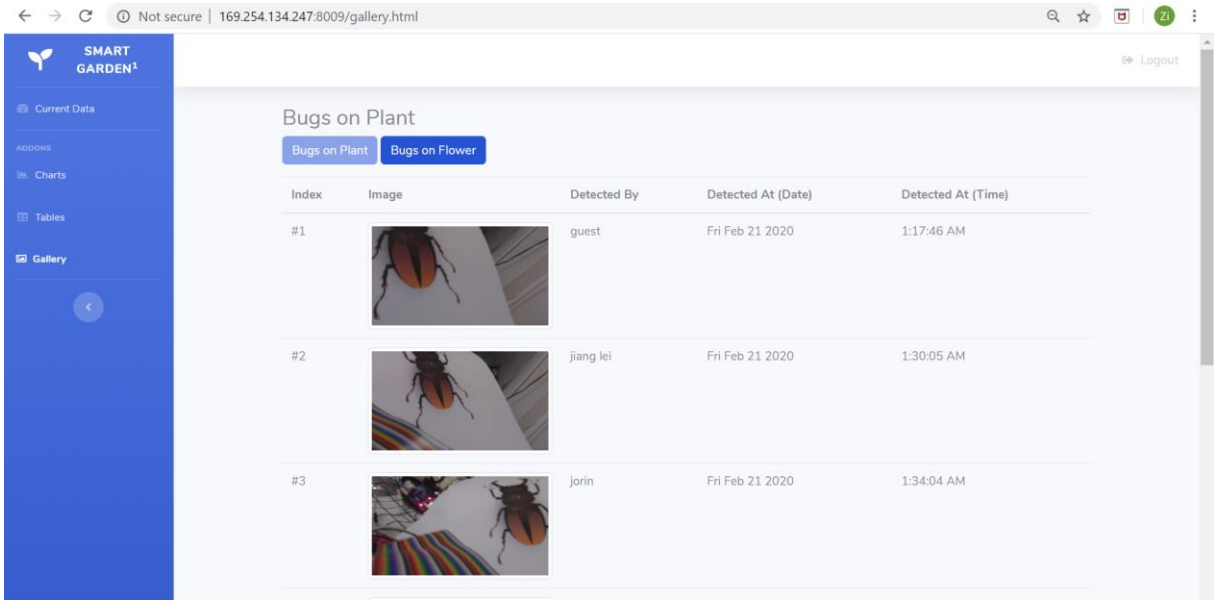


Figure 6: Gallery Page – Bugs on Plant (1C.5)

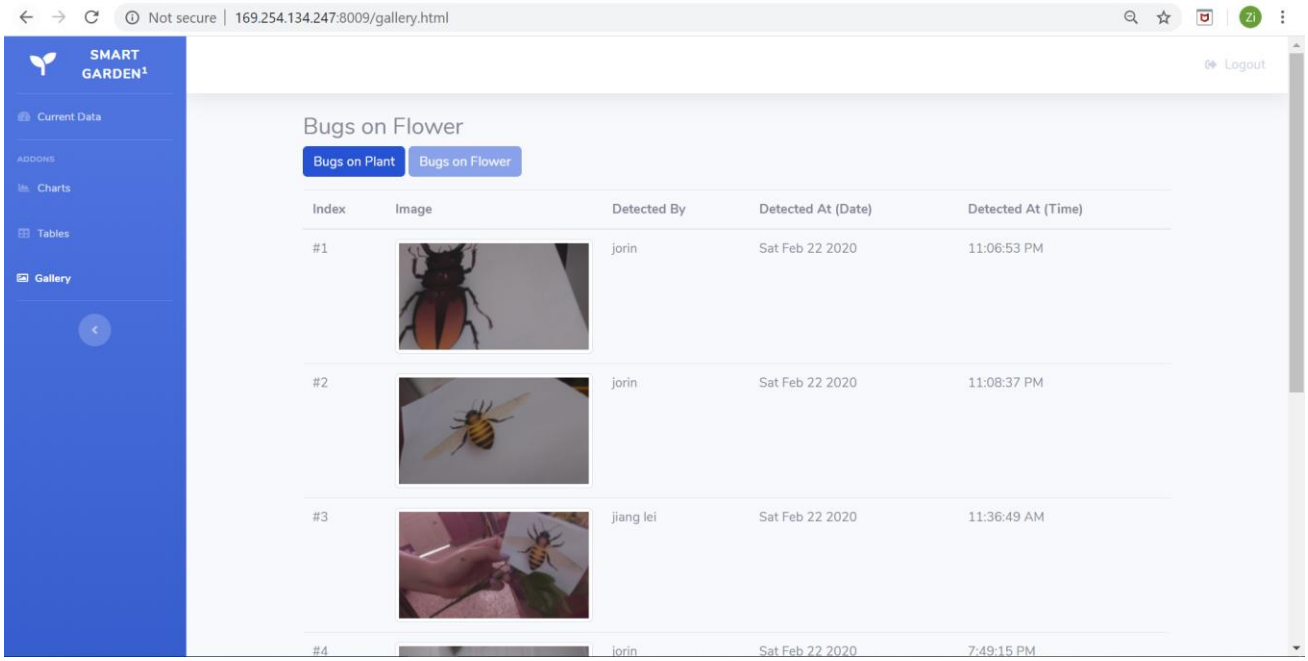


Figure 6: Gallery Page – Bugs on Flower (1C.6)

E. How does the features look like?

1. Real-time light alerting Message – Telegram

When the LDR detects light value less than 600, it will send user a telegram message stating that there is not enough light for the plant and the light value that was detected.

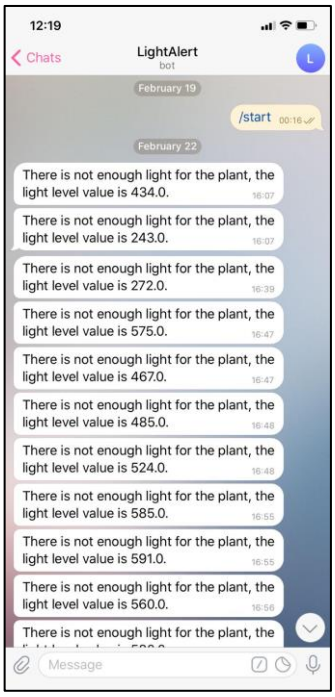


Figure 7: Real-time light alert message

2. Email

When the Soil Moisture Sensor detects the Soil Moisture value is more than 500, it will send the user an email alert with the soil moisture value and a text 'Your plant is too dry now'. Email alert will only be sent when the Soil Moisture Sensor detects the soil is more than 500.

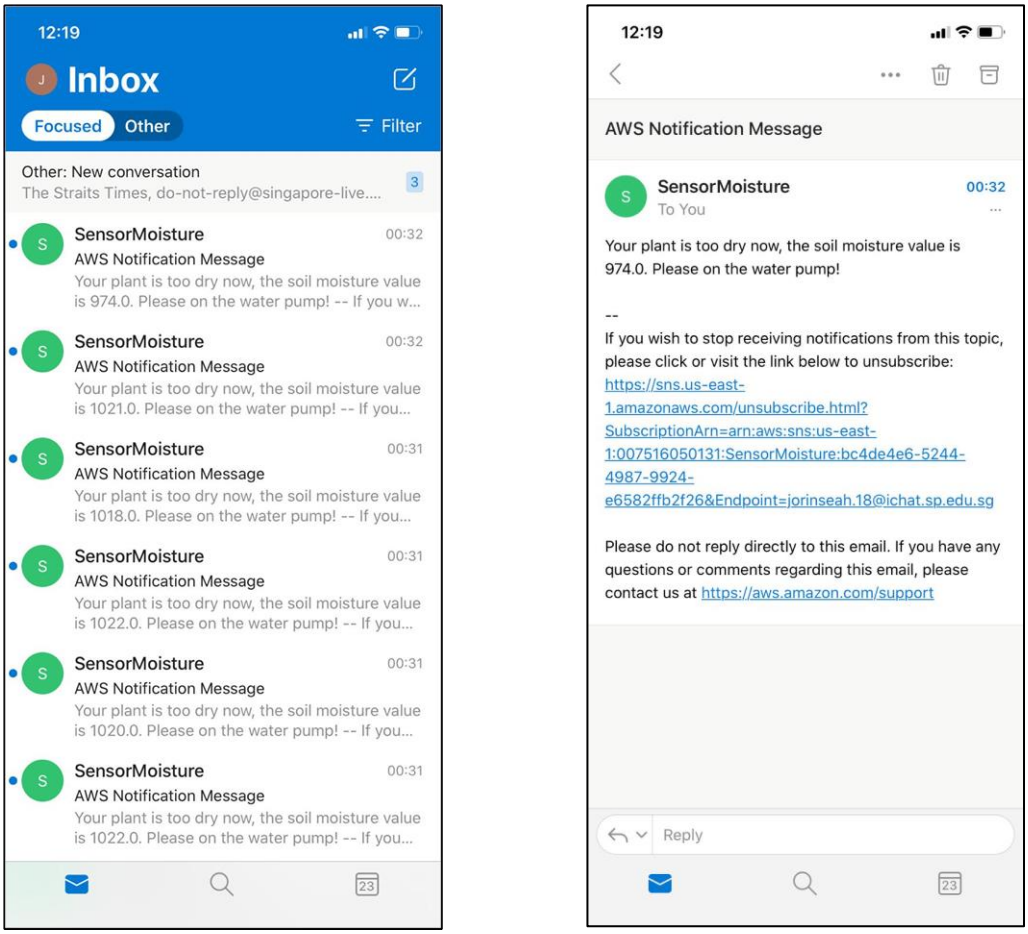


Figure 8: Email alert message

3. Whatsapp

Authorization using RFID card reader when tapping NFC card, the user will also receive a message of current plant condition on Whatsapp using Twilio services.

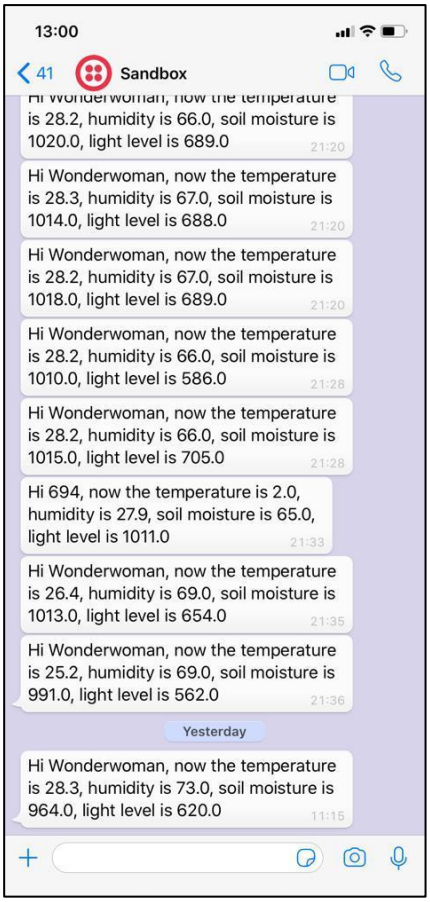


Figure 9: Whatsapp message received with 4 Plant conditions

4. Taking picture of flowers and plants

If user just want to view flower only, the user just need to type 'flower' (Figure 10) and the piCam will take a picture of the flower and send it to the user regardless of whether there is any bugs on it. It is the same for plant, where user just type 'plant' (Figure 11).

Figure 10: Taking picture of the flower

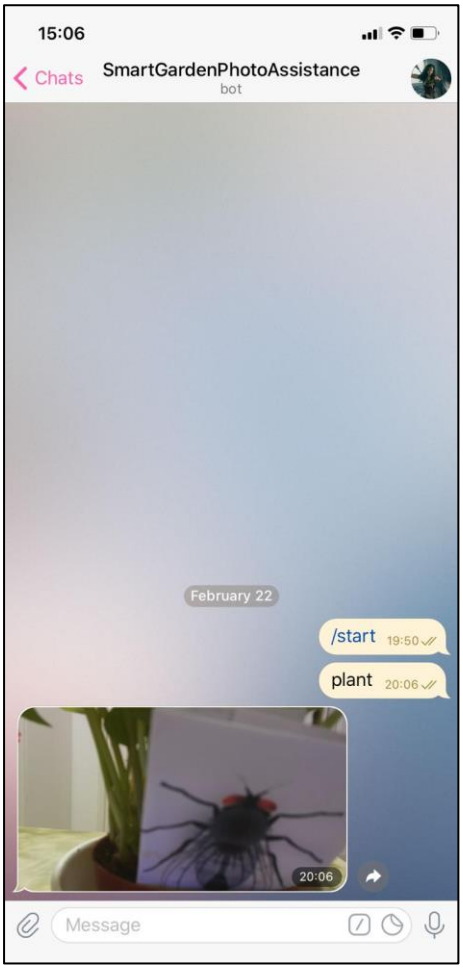


Figure 11: Taking picture of the Plant

5. Upload to twitter

Continued from last step, the picture requested by the user would also be auto uploaded to the twitter, with the message stating the picture type (plant / flower), and uploaded person, the one asking for the picture.

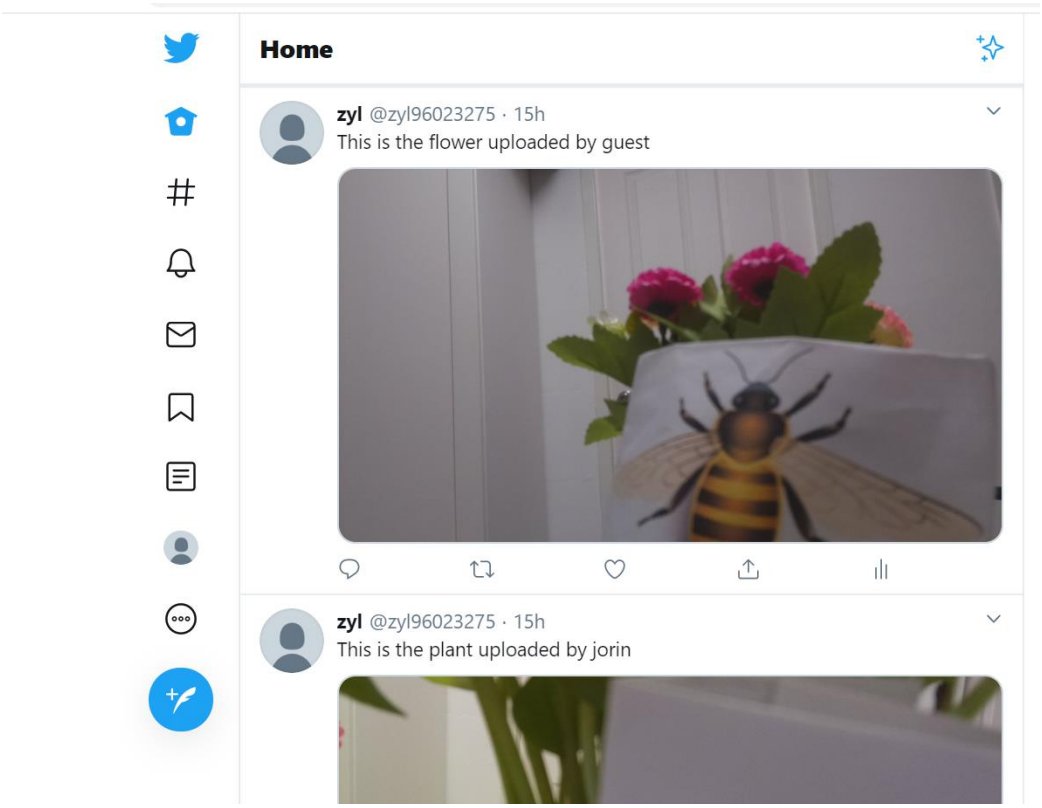


Figure 12: Twitter – Pictures uploaded

6. Detecting bugs – Telegram

When the user is not at home and he/she wants to see if there is any bug on the plant or flower, the user can use this telegram bot (Figure 13) to detect. Once the piCam detect bugs on the plant or flower, it will take and send a pic of the bugs to the user.

User type 'detect' and both of the piCam will start to detect for bugs each for 5 times. It will only take and send picture when bugs are detected on either the plant or flower. And these pictures (with bugs) will be stored into S3 bucket, and displayed in the web server.

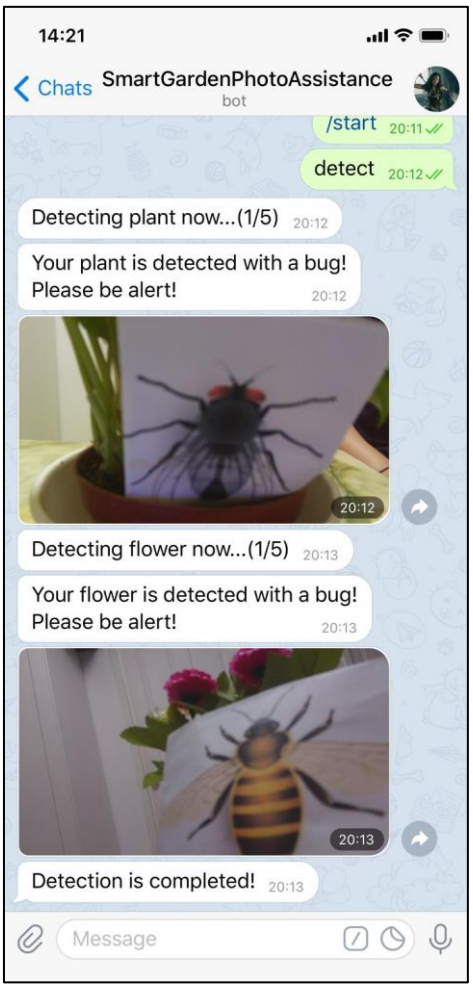
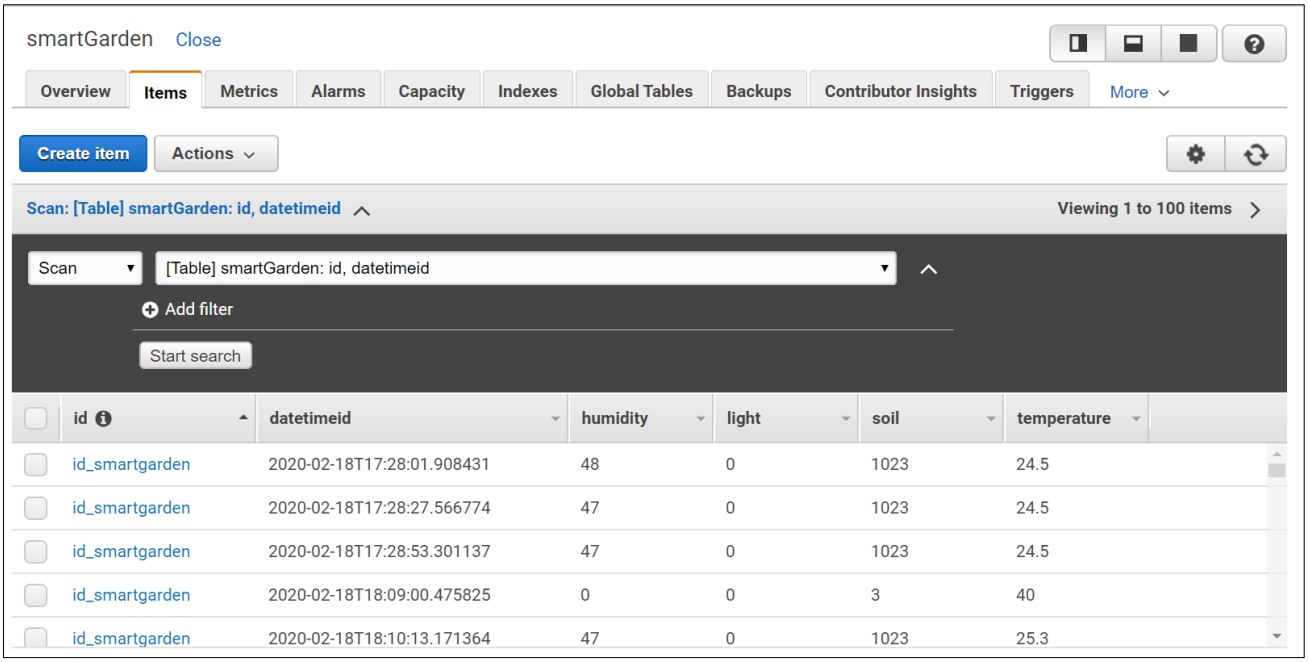
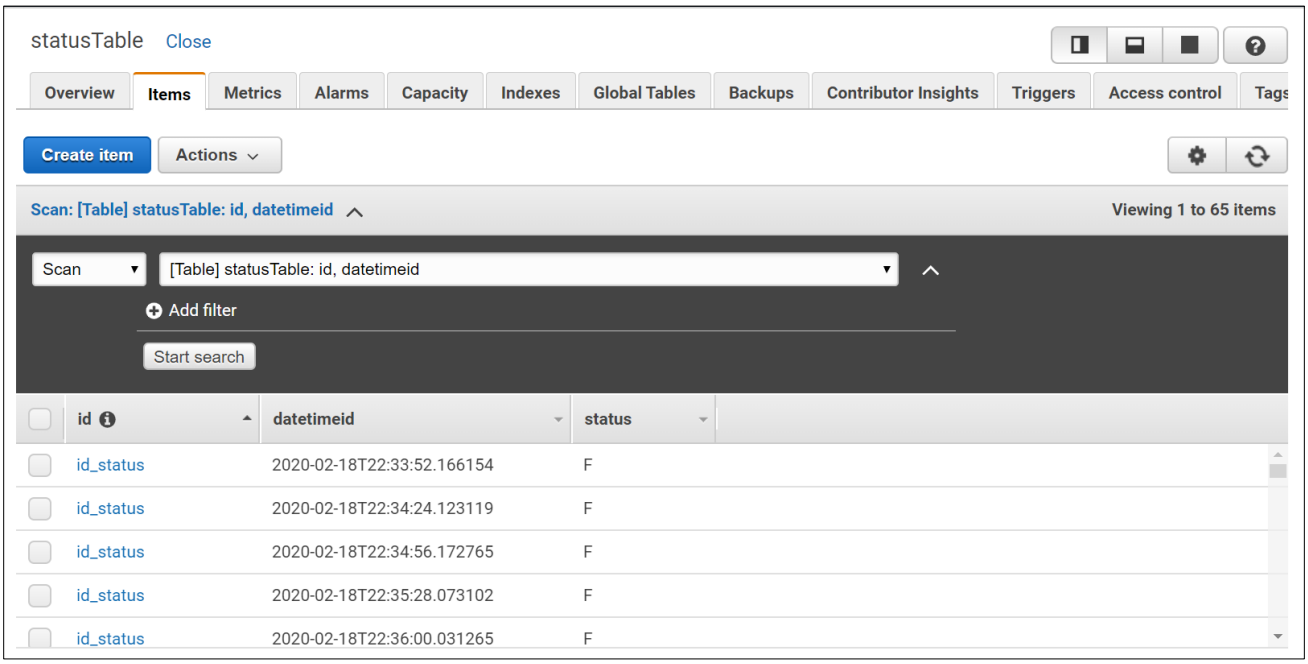


Figure 13: Telegram – Detecting bugs

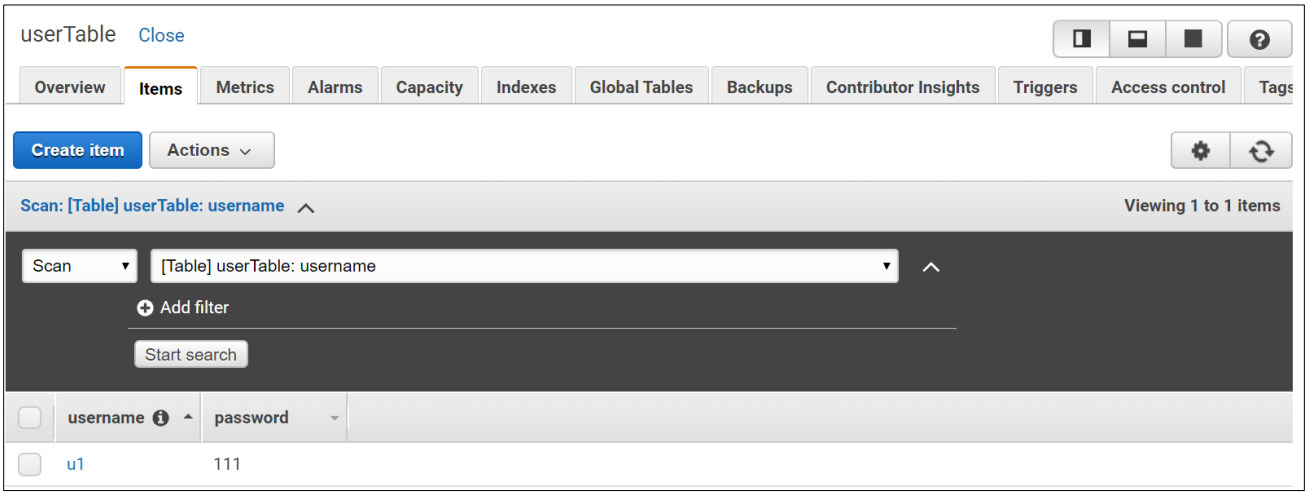
F. How does the database look like?

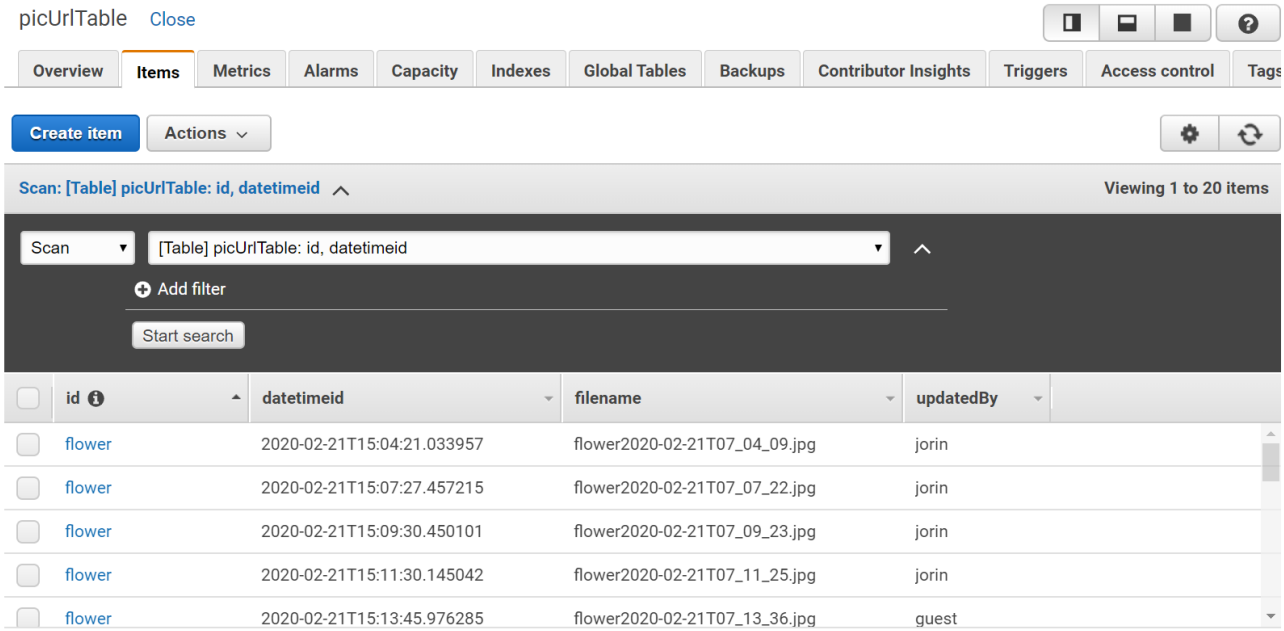


id	datetimeid	humidity	light	soil	temperature
id_smartgarden	2020-02-18T17:28:01.908431	48	0	1023	24.5
id_smartgarden	2020-02-18T17:28:27.566774	47	0	1023	24.5
id_smartgarden	2020-02-18T17:28:53.301137	47	0	1023	24.5
id_smartgarden	2020-02-18T18:09:00.475825	0	0	3	40
id_smartgarden	2020-02-18T18:10:13.171364	47	0	1023	25.3

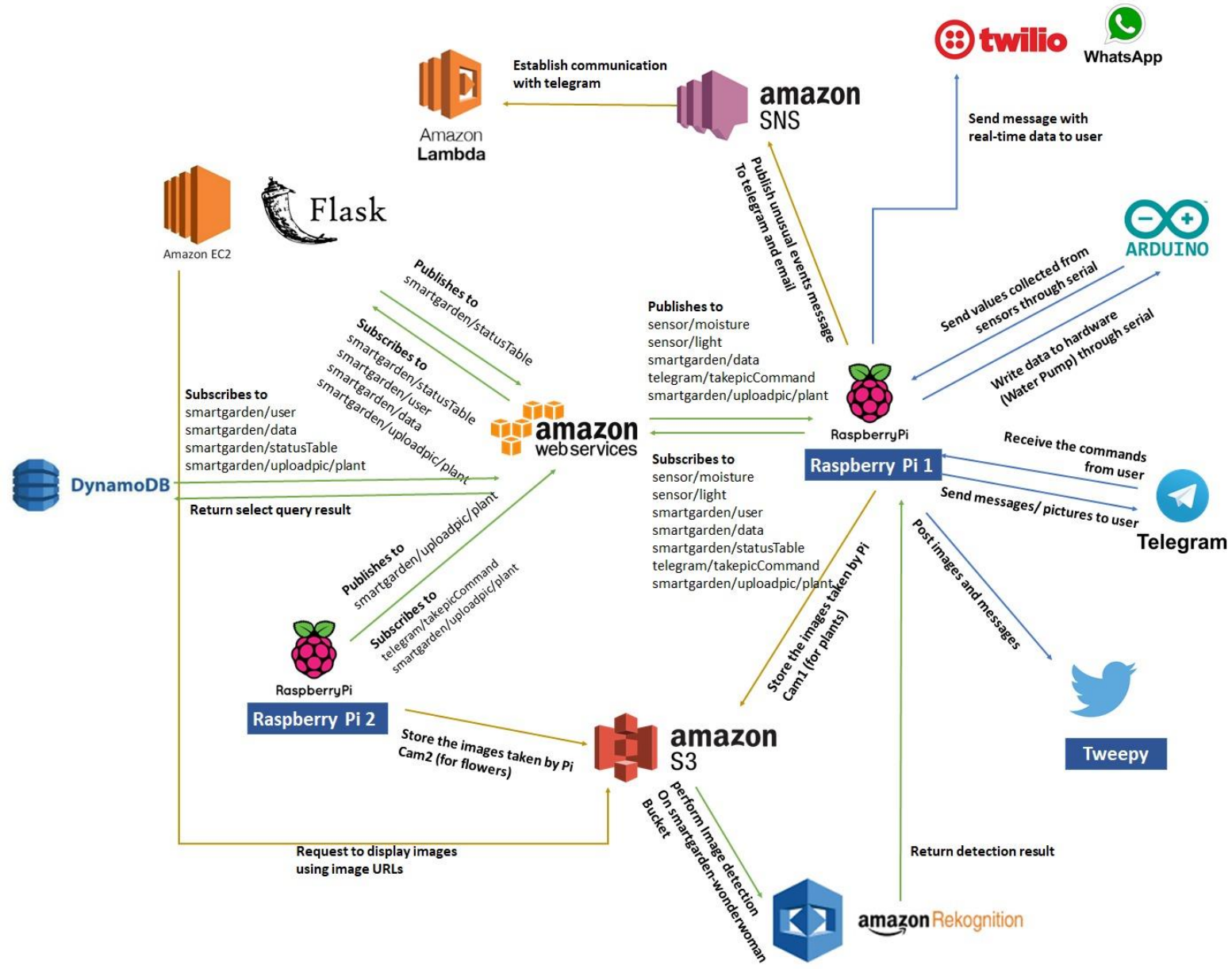


id	datetimeid	status
id_status	2020-02-18T22:33:52.166154	F
id_status	2020-02-18T22:34:24.123119	F
id_status	2020-02-18T22:34:56.172765	F
id_status	2020-02-18T22:35:28.073102	F
id_status	2020-02-18T22:36:00.031265	F


userTable
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Global Tables
Backups
Contributor Insights
Triggers
Access control
Tags
Create item
Actions
Scan: [Table] userTable: username
Viewing 1 to 1 items
Scan
[Table] userTable: username
Add filter
Start search
username
password
u1
111


picUrlTable
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Global Tables
Backups
Contributor Insights
Triggers
Access control
Tags
Create item
Actions
Scan: [Table] picUrlTable: id, datetimeid
Viewing 1 to 20 items
Scan
[Table] picUrlTable: id, datetimeid
Add filter
Start search
id
datetimeid
filename
updatedBy
flower
2020-02-21T15:04:21.033957
flower2020-02-21T07_04_09.jpg
jorin
flower
2020-02-21T15:07:27.457215
flower2020-02-21T07_07_22.jpg
jorin
flower
2020-02-21T15:09:30.450101
flower2020-02-21T07_09_23.jpg
jorin
flower
2020-02-21T15:11:30.145042
flower2020-02-21T07_11_25.jpg
jorin
flower
2020-02-21T15:13:45.976285
flower2020-02-21T07_13_36.jpg
guest

G. System architecture of our system



H. Evidence that we have met basic requirements

Requirement	Evidence
Used three sensors	Used 6 Sensors: • Soil Moisture Sensor • DHT11 Sensor • LDR sensor • MFRC522 Card Reader • 2 Pi Cameras Refer to screenshot labelled, Figure 1: Hardware Setup - (1A) and Figure 1: Hardware Setup – (1B).
Used MQTT	Our MQTT endpoint to publish and subscribe the data from AWS services. • sensor/moisture • sensor/light • smartgarden/user • smartgarden/data • smartgarden/statusTable • telegram/takepicCommand • smartgarden/uploadpic/plant Refer to system architecture, Section1 Part G
Stored data in cloud	Implemented DynamoDB to store the data Refer to system architecture, Section1 Part F
Used cloud service	• AWS Rekognition to perform image detection • Amazon EC2 to host the server • AWS SNS to send alerting email to user • AWS Lambda to establish the communication with Telegram • AWS S3 to store the images taken by user • DynamoDB to store data • Telepot to conduct 2-way communication with telegram • Teepy to upload images and messages to twitter account

	• Twilio to send message to whatsapp Refer to system architecture, **Section1 Part F**
Provide real-time sensor value / status	Show the real-time value of 4 sensor : • Temperature • Humidity • Soil Mositure • Light Refer to screenshot labelled, **Figure 3: Web Index page (1C.2)**
Provide historical sensor value/ status	• Show 4 charts of historical vlaue of Temperature, Humidity, Soil Moisture and Light value • Show 2 tables of historical vlaue of Temperature, Humidity, Soil Moisture and Light value, and status of water pump Refer to screenshot labelled, **Figure 4: Web Index page (1C.3)** and **Figure 5: Web Index page (1C.4)**
Control actuator	Refer to screenshot labelled, **Figure 3: Web Index page (1C.2)** • Placed toggle button on webpage to control water pump actuator. • Scan the NFC card and view the welcome message with name of owner on LCD screen • Scan the NFC card to trigger the buzzer with customised music

I. Bonus features on top of basic requirements

a) Log in system

Able to authenticate the user before granting access to the application, as shown in Figure 2: Web Login page (1C.1). Use session to manage it.

b) 6 Sensors & 6 Actuators

We used a total of 6 Sensors & 6 Actuators as can be seen in Figure 1: Hardware Setup - (1A) and Figure 2: Hardware Setup – (1B).

c) Read from & Write to Arduino

We set up most of the hardware with Arduino, such as water pump, LCD light, soil moisture sensor, MFRC522 Card Reader, DHT11, Buzzer, etc.

d) Customised Buzzer Sound

We have identified some of the NFC card owners as authorised user while the rest is guest. When the authorised user scan the card with MFR522 card reader, the buzzer will make a customised pleasing sound, while a normal sound for others.

e) AWS S3 Bucket

Whenever there is a bug detected, the images would be uploaded to the S3 storage. Since we have used multiple raspberry pi, the S3 is implemented as a platform that allows 2 Pi to share the images.

f) Object Rekonigition

We used AWS Rekonigition to detect bugs on plant and flower as can be seen in Figure 13: Telegram – Detecting bugs

g) Telepot

We used telegram as a 2-way communication with the user. The user can send command to request the picture of any plant/flower, as well as detect bugs, seen in Figure 10: Taking picture of flower , Figure 11: Taking picture of Plant and Figure 13: Detecting Bugs

h) SNS

If the soil is too dry, the system will automatically send an email to user to alert him about the unusual event. Figure 8: Email alert message

i) Lambda

This service is used to establish the commuication with the telegram, that users can get

alert message and take picture as shown in Figure 7: Real-time light alert message

j) 2 Pi Cameras

2 Raspberry pi has been set up, and they can communicate using MQTT by subscribing and publishing to the same topic. And both of them uploaded images to the same bucket in the AWS S3 service. Cameras are used for taking pictures of the plant and flower as seen in Figure 10: Taking picture of flower , Figure 11: Taking picture of Plant and Figure 13: Detecting Bugs

J. 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) Install all the software libraries required
3) Upload smartGarden.ino to Arduino
4) Run scripts.py file
5) Then run the server.py file for web server
6) Run the telegram_scripts.py for Telegram bot
7) Run the tele_flower.py on another rasphberry pi to activate another pi camera.

Section 2
Hardware requirements

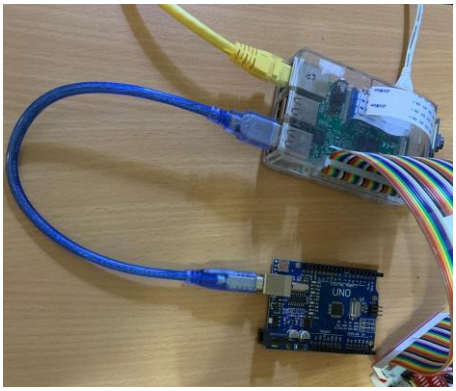
Hardware checklist

	Item	Quantity
1)	Arduino UNO	1
2)	DHT11 Temperature & Humidity Sensor	1
3)	Soil Moisture Sensor	1
4)	Water Pump (5V DC Motor)	1
5)	Silicone Tubes	2
6)	Buzzer	1
7)	LED (red)	1
8)	LED (yellow)	1
9)	i2c LCD Screen (16x2)	1
10)	Light-Dependant Resistor (LDR)	1
11)	MFRC522 RFID	1
12)	PN2222 Transistor	1
13)	1N4001 Diode	1
14)	220 Ω Resistor	3
15)	10k Ω Resistor	2
16)	Alligator jumper wires	8
17)	USB 2.0 Cable	1
18)	Pi Camera	2
19)	Raspberry Pi	2

Hardware setup instructions

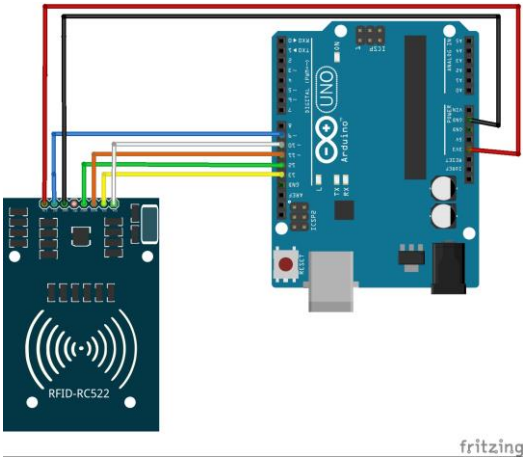
Connect Arduino to Raspberry Pi

a) Connect Arduino to Raspberry Pi via a USB 2.0 Cable as shown in the figure.



MFRC522 Card Reader

a) Connect the MFRC522 Card Reader to the Arduino UNO as shown.

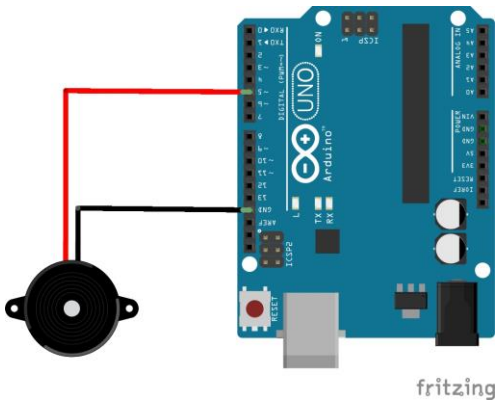


Jumper color	buzzer pin	Arduino pin
Red	3.3V	3V3
Blue	RST	Digital Pin 9
Black	GND	GND
Orange	MOSI	Digital Pin 11
Green	MISO	Digital Pin 12
Yellow	SCK	Digital Pin 13
White	SDA	Digital Pin 10

Buzzer

a) Connect the buzzer to the Arduino UNO as shown.

Jumper color	buzzer pin	Arduino pin
Black	GND	GND
Red	VOUT	Digital Pin 5

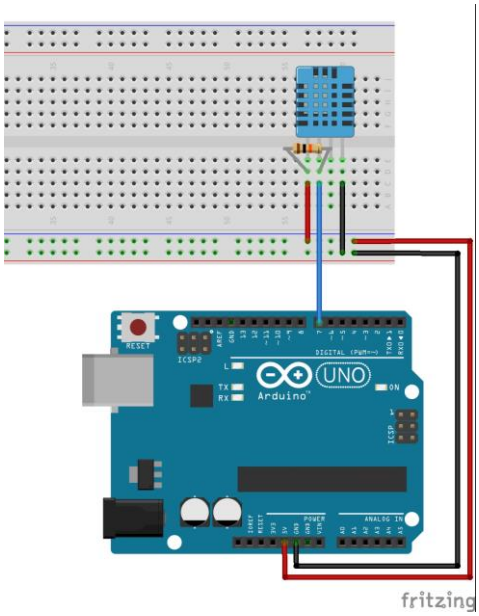


DHT11

a) Add a DHT11 sensor near the end of the breadboard as seen in the figure.

Connect them to the Arduino pins with the corresponding color jumper cables as shown in the diagram below.

DHT11 Sensor	Arduino Pin	Jumper color
VCC	5V	Red
DATA	D7	Blue
NC		
GND	GND	Black



LED

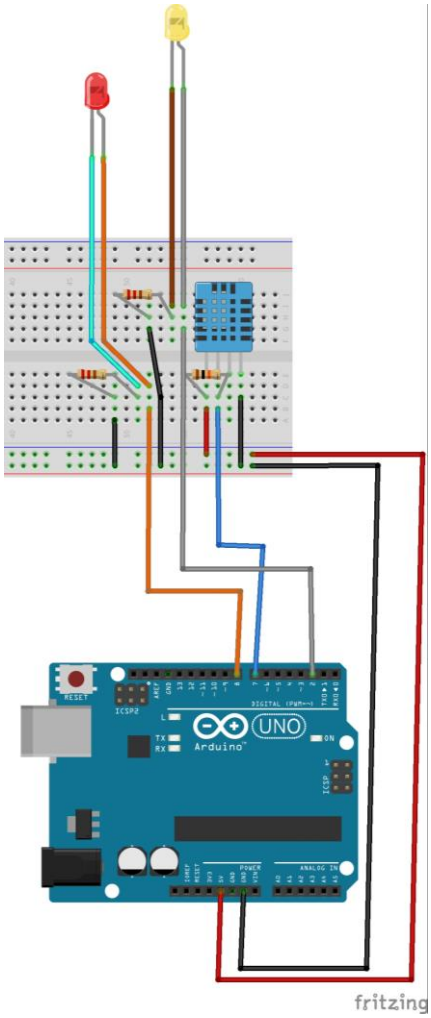
a) Insert a Yellow LED and Red LED beside the DHT11 sensor as seen in the figure.

Add in 2 220 ohms resistor for the LED, with one end connected to the longer end of the LED.

Connect them to the Arduino pins with the corresponding color jumper cables as shown in the diagram below.

LED (red)	Arduino Pin	Jumper color
Long-end	D8	Orange
Short-end	GND	Black

LED (Yellow)	Arduino Pin	Jumper color
Long-end	D2	Grey
Short-end	GND	Black



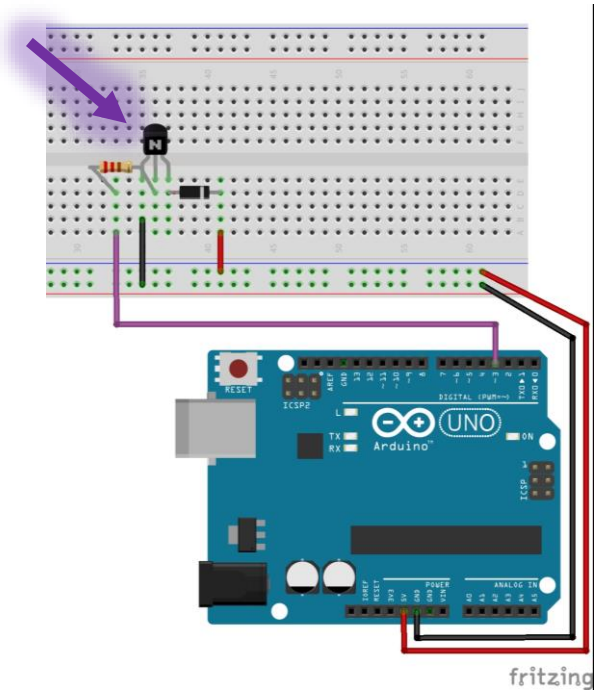
DC Motor

a) Add a PN2222 transistor in the breadboard as shown in the figure.

The emittor (left pin) of the transistor should connect to the GND pin of the Arduino.

The base (middle pin) should connect to one end of the resistor.

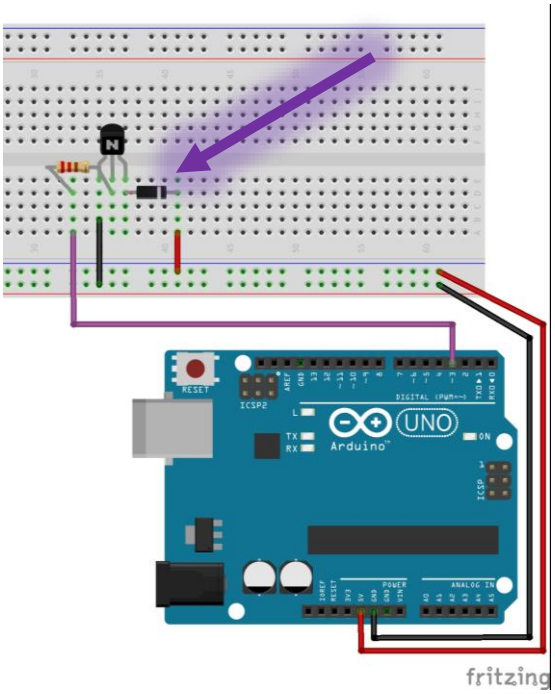
The collector (right pin) should connect to the cathode of the diode.



b) Add a 1N4001 diode to the breadboard as shown in the figure.

The cathode (white end) of the diode should connect to one end of the motor and the 5V pin of the Arduino.

The anode pin of the diode should connect to the collector of the transistor and the other end of the DC motor.

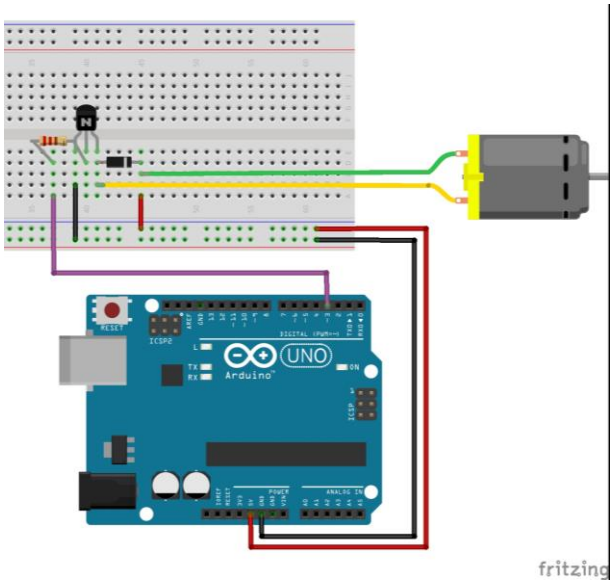


DC Motor

c) Connect the DC Motor to the breadboard as shown in the figure using alligator jumper cables.

The motor can be connected either way around.

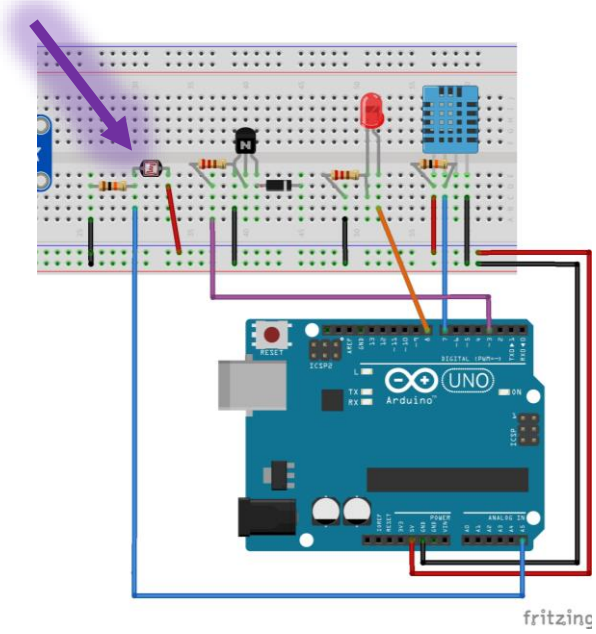
One end should connect to the anode pin of diode and the other end connect to the cathode (white end) of diode.



LDR

a) Insert a LDR beside the PN2222 sensor as seen in figure.

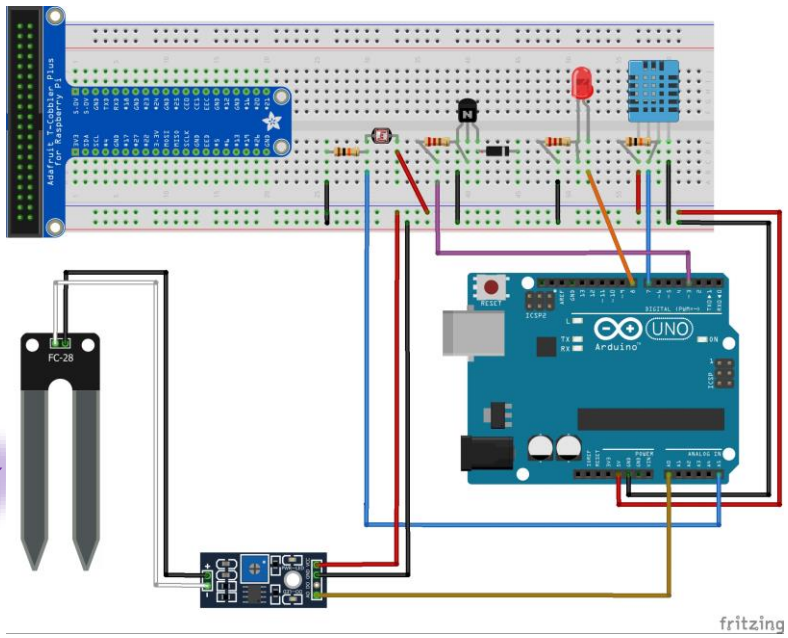
Add in 10k ohms resistor for the LDR, with one end connected to the end of the resistor and another connect to 5V.



Soil Moisture Sensor

a) Insert a Soil Moisture sensor as seen in figure.

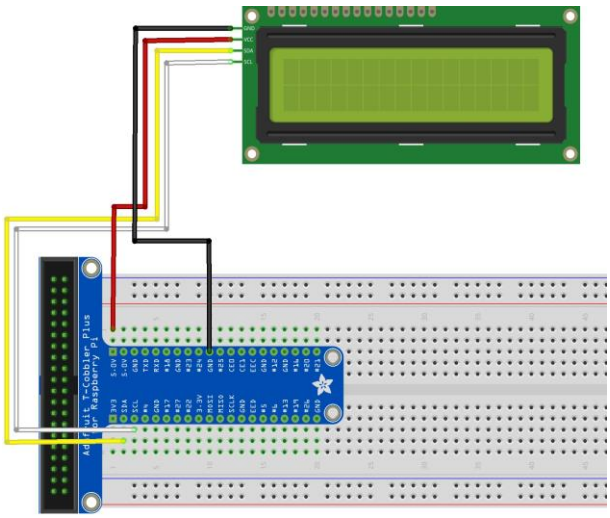
Add in 10k ohms resistor for the LDR, with one end connected to the end of the resistor and another connect to 5V.



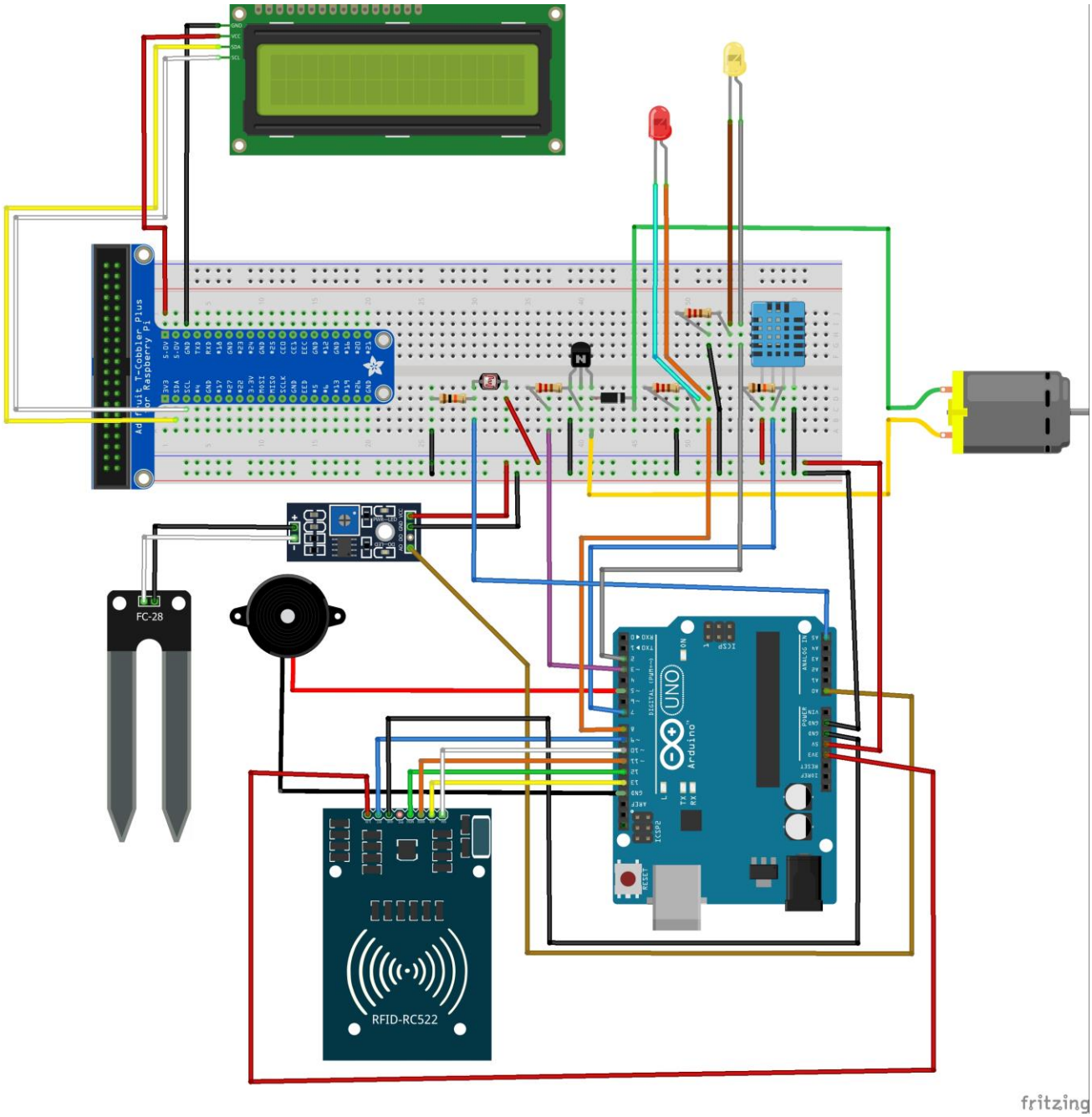
LCD

b) Connect the pins on the LCD to the RPi as follows:

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



Fritzing Diagram



Section 3
Software Requirements

Software checklist

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

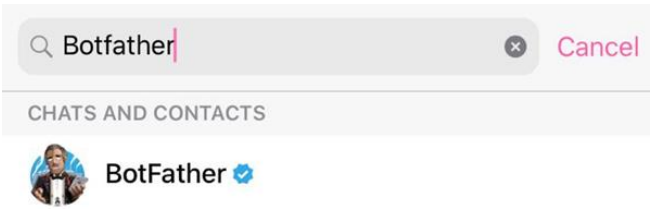
1. Telegram Bot setup
2. Twilio Setup
3. Twitter setup
4. MFRC522 library Installation

Software setup instructions

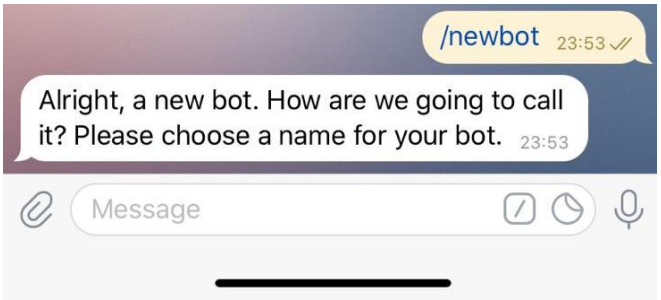
Telegram Bot

Task

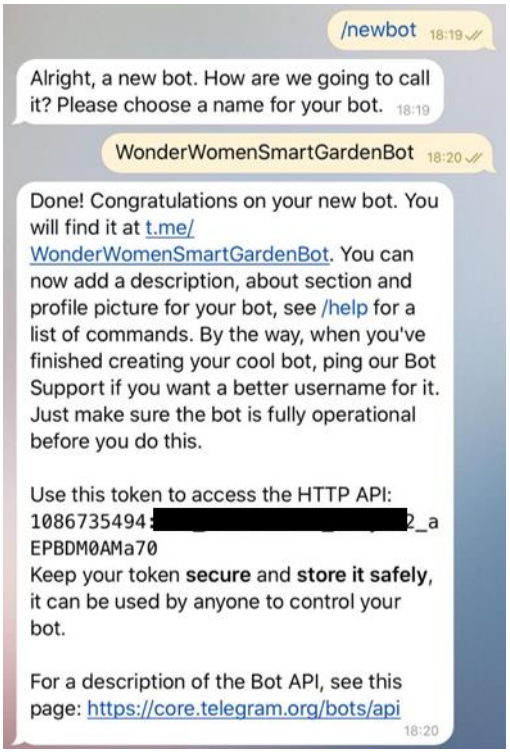
a) Open Telegram app in your laptop or mobile and search “BotFather”



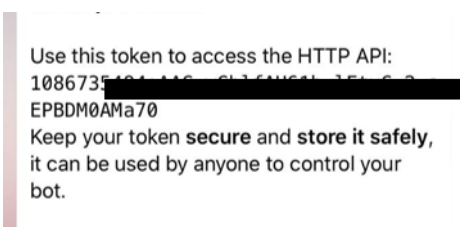
b) Type /newbot to create a new bot



c) Give a username to your bot. Make sure it ends with _bot.

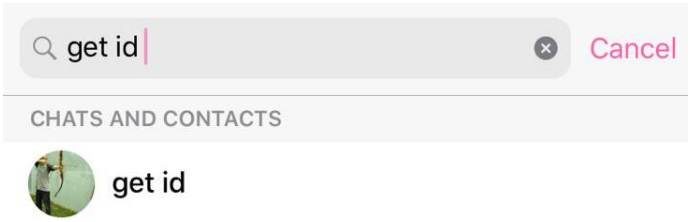


d) Copy down the access token that is issued by BotFather

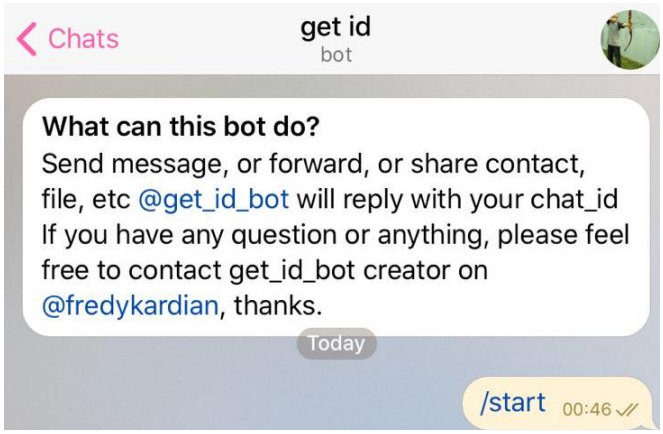


e) Repeat step A to D to created another bot for Light Alert.

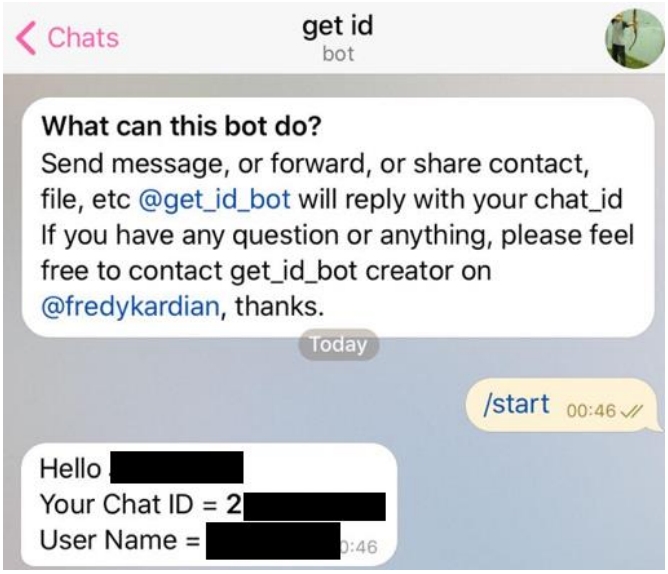
f) start another bot called “get_id_bot”



g) Type /start to start the bot



h) Copy down the Chat ID gotten by the bot



i) Install the Telegram API on your Raspberry Pi

```
sudo pip install telepot
```

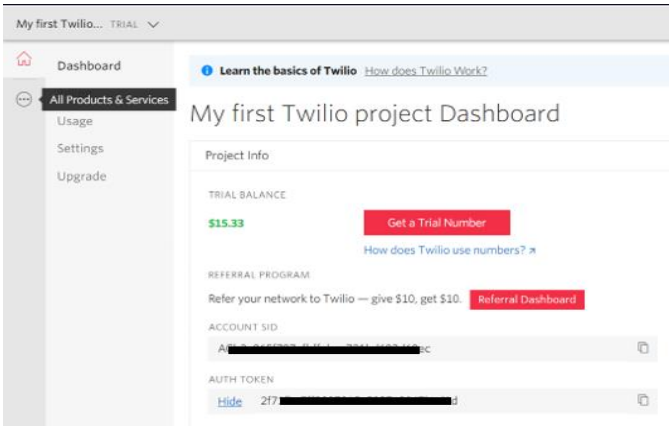
Twilio

Task

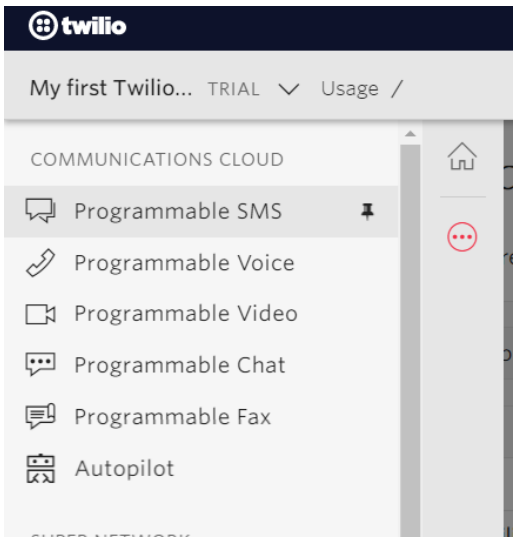
a) Set up a new Twilio account at https://www.twilio.com/try-twilio and log into the console.

b) Create Twilio account

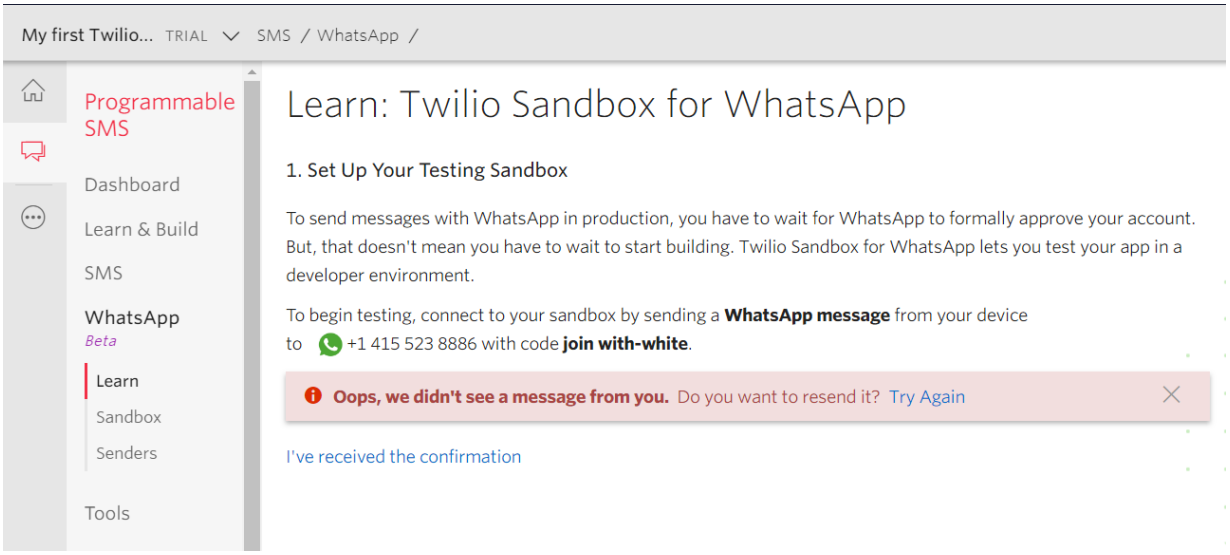
Copy your Twilio Account SID and Auth Token which should be shown on the dashboard when you log in. Save it in a notepad.



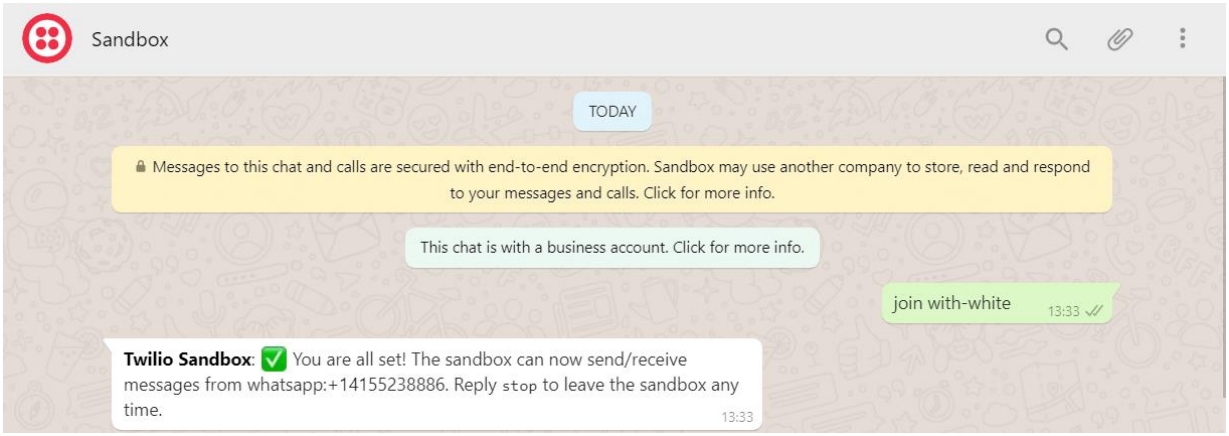
c) Click on the 3-dot icon. Select "Programmable SMS"



d) Select “Whatsapp” > “Learn”.
e) To begin testing, connect to sandbox by sending a Whatsapp message from your device.
f) Add “+1 415 523 8886” into your contact.



g) Send “join with-white” message to sandbox on Whatsapp.

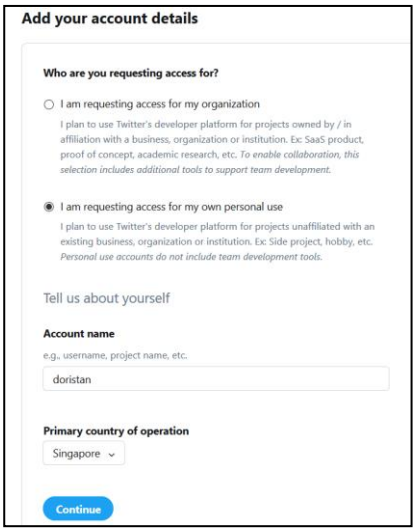
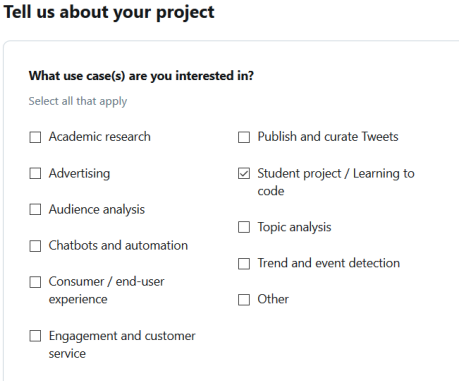


h) Install twilio on raspberry pi.

```
sudo pip install twilio
```

Twitter

A. Apply for a Twitter Developer account

Task
a) Go to https://apps.twitter.com/ and log in with your Twitter account password
b) If you don't have one, you will be asked to apply for a Developer account and redirected to a URL similar to this one (https://developer.twitter.com/en/apply/user). Follow the instructions to apply for the account so that you can access the Twitter APIs.
c) When prompted to add your account details, you can choose the option "I am requesting access for my own personal use" and indicate a preferred username for it.   
d) You can indicate your use case for this project as "Student project"   
e) Twitter will ask you for the reason for requesting for the developer account. 1) I'm using Twitter's APIs to do my lab in Singapore Polytechnic. I need to curate a collection of tweets based on keywords which I will analyse with the skills and knowledge I learn in my course.

Task

You can use this sample response as shown on the right

Describe in your own words what you are building

Please describe what you would like to build with Twitter's APIs. Be sure to give detailed answers to the following questions. If the question does not apply to your solution, please explicitly state that. The more detailed the response, the easier it is to review and approve.

2) I plan to analyze Tweets, Twitter users and their content. Some of the questions I plan to answer with my analysis would be, who are the top influencers, what are the active timezones etc
3) Yes, I will be Tweeting content but not Retweeting or liking content. I do not expect any automated interaction with Twitter users in my project.
4) Any Twitter data would be presented as visualizations or captured in Powerpoint slides to my professor only.

f) At this question, just indicate 'No'

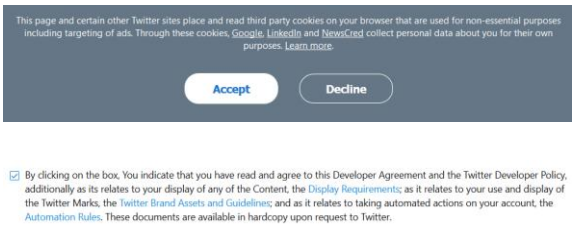
Will your product, service, or analysis make Twitter content or derived information available to a government entity?

In general, schools, colleges, or universities do *not* fall under this category.

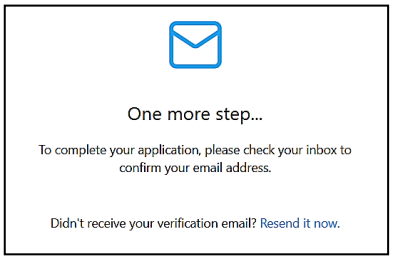
(•) No
( ) Yes

g) On the next page, make sure you scroll through the Terms and conditions fully, then click the "Accept" button as well as the checkbox below.

Click the "Submit" button

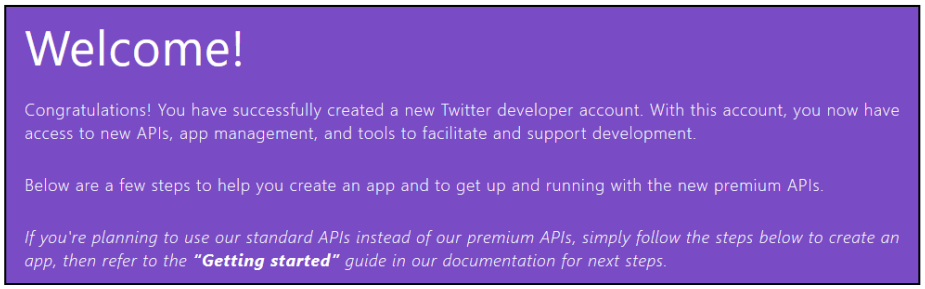


h) After the previous step, Twitter will send you a verification email.

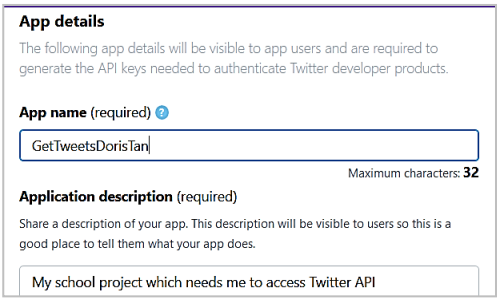
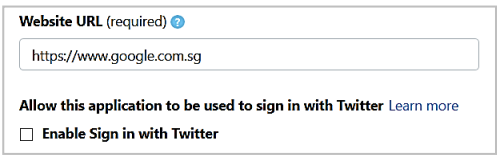


i) Verify your account by clicking the link in the email Twitter sends you.

j) After clicking the link above, you should automatically be redirected to the Developer website https://developer.twitter.com and see a welcome message like this.

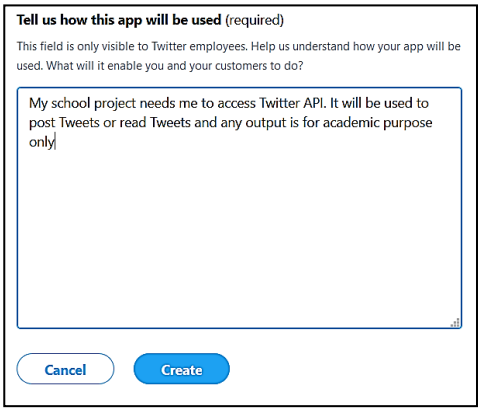
Task
Congrats, you are now ready to proceed to the next section :)   

B. Set up a Twitter Account

Task	
a)	Ensure you have an approved Twitter Developer account before you perform this task. Refer to Part A for details. Go to https://apps.twitter.com/ and log in with your Twitter account password
b)	Look for a "Create New App" button or link and click on it.
c)	Fill in your app details Attached is a sample how you can fill up the App name and Application description sections. Note that the App name must be a unique one across the whole world. 
d)	You can just specify a random URL like what I did. Do not check "Enable sign in with Twitter" 

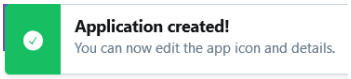
Task

e) Here is a sample how you can fill up the "Tell us how this app will be used" section



f) Click the "Create" button

g) You should see a "Application created" message like this.



C. Install Tweepy Python library

Task

a) Install Tweepy Python library, we will need this to send tweets out later

```
sudo pip install tweepy
```

MFRC522 Library

Task

A. Install the MFRC522 library from https://github.com/ljos/MFRC522.

B. Once downloaded, extract the content of the zip files inside your "sketchbook/libraries" folder.

Section 4
Setup MQTT,SNS,S3,Lambda and DynamoDB

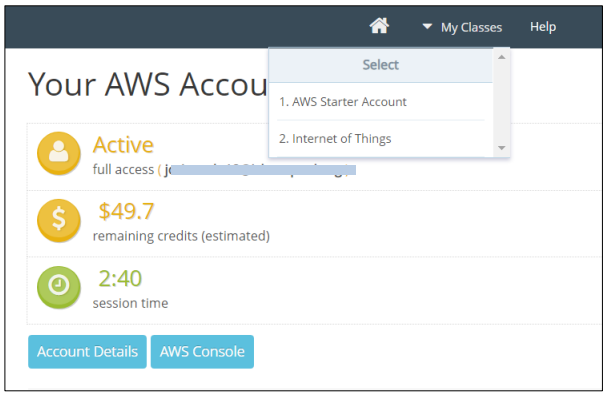
1. Sign in to the AWS IoT Console

Task

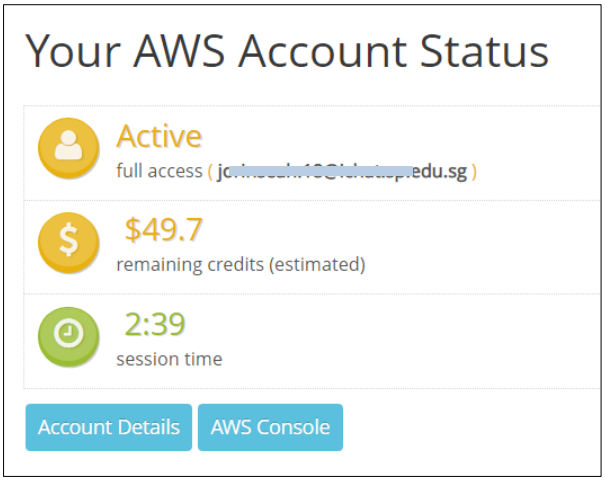
a) Turn on your Raspberry Pi and confirm you have an Internet connection.

b) Sign in with your AWS console at https://aws.amazon.com

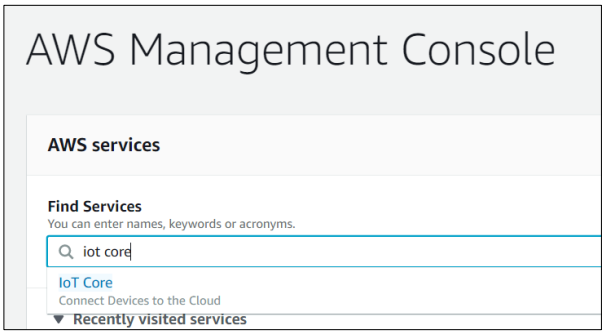
c) Select " Internet of Things" Under My Classes.



d) Click on "AWS Console"

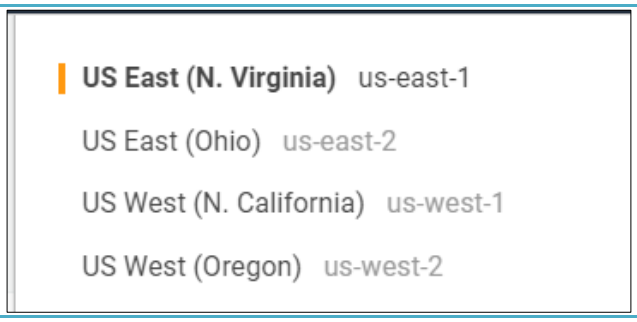


e) After a while, you will be brought in the AWS Management Console.
In the AWS dashboard, type " IoT Core" to access the AWS IoT service.



f) On the Welcome page, choose Get started

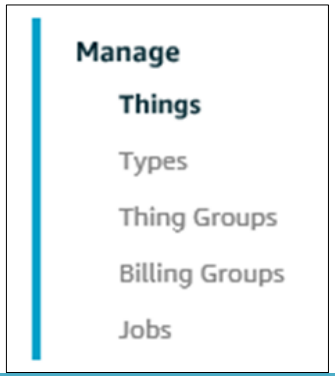
g) Ensure that the region selected at the top right is "US East (N. Virginia) us -east-1"



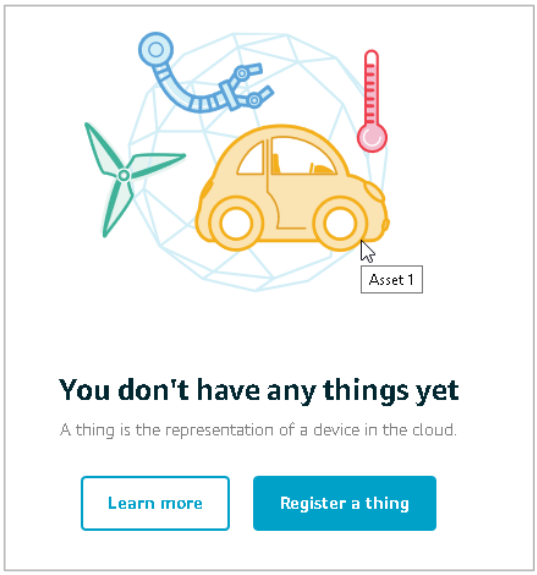
A. Create and register your "Thing"

Task

a) In the left navigation pane, click "Manage" to expand it, then choose "Things".

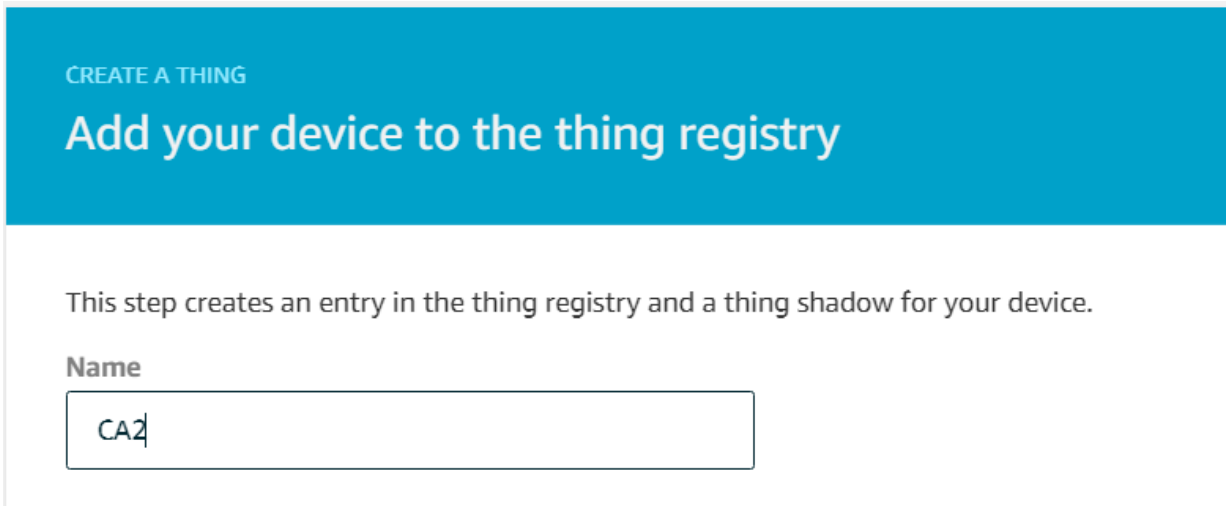


b) On the page that says "You don't have any things yet", choose "Register a thing". If you have created a thing before, choose Create.



c) 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 "EcoPlant" for the name.

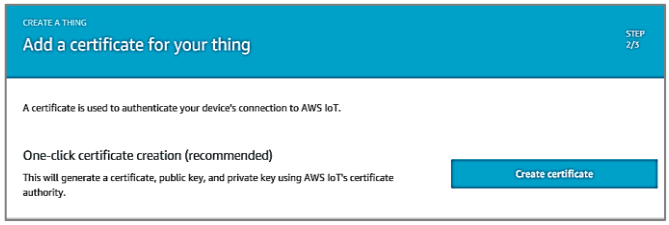
Click “Create a single thing”



B. Create Certificates

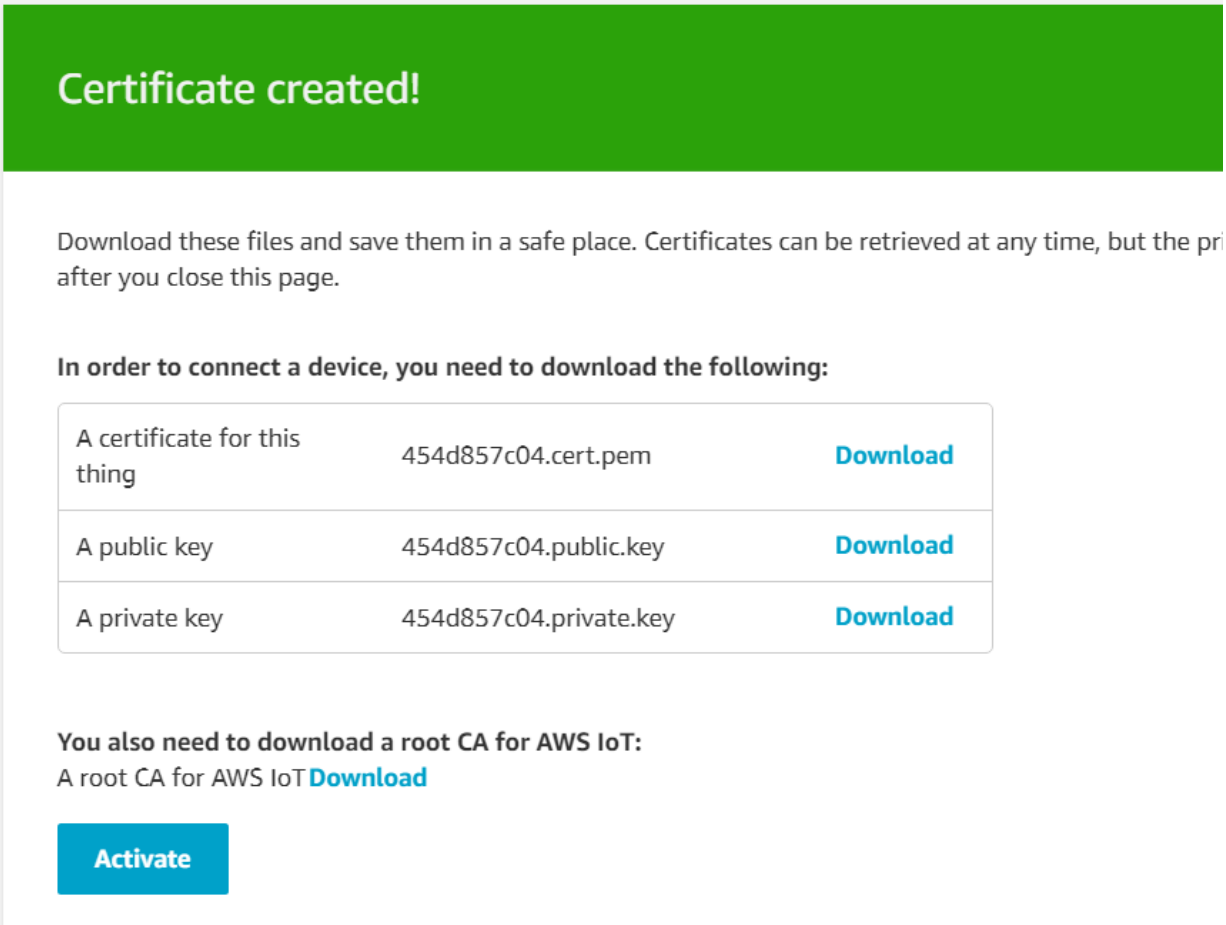
Task

a) After you have completed the previous section, you might be presented with this screen.



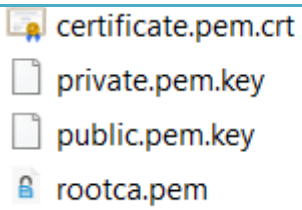
b) Choose “One-click certificate creation” to generate an X.509 certificate and key pair.

c) After a while, you should see the following screen, where there are a total of **four** download links.

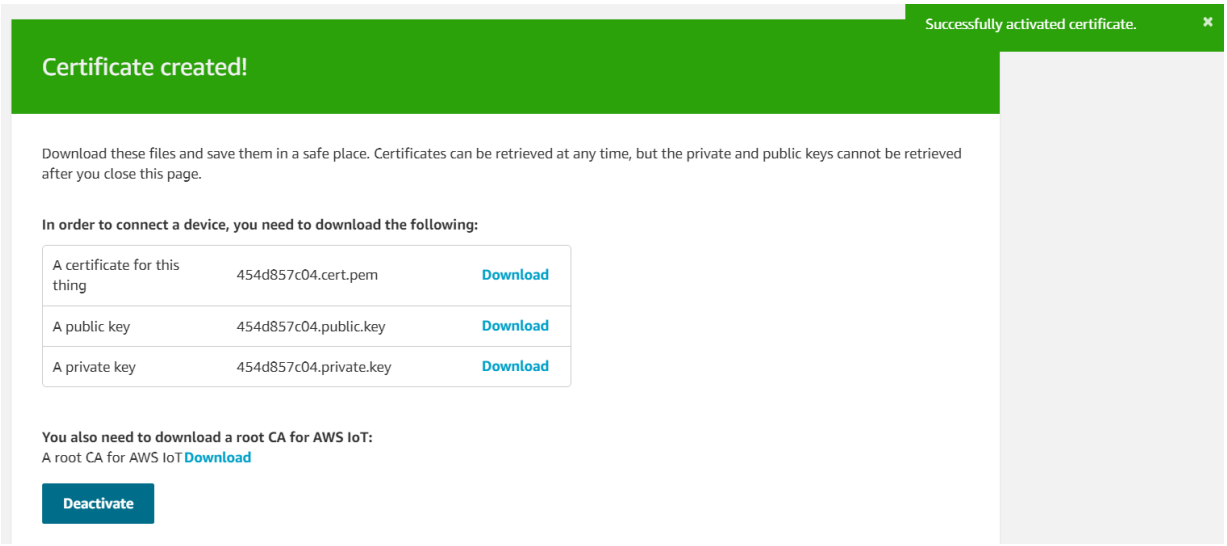


d) Direct link for root CA1 can be found at the link below.For the root CA, you can choose CA 1. Do a right-click, “Save As” when clicking on the link. Do not click directly on the link.
https://docs.aws.amazon.com/iot/latest/developerguide/server-authentication.html#server-authentication-certs

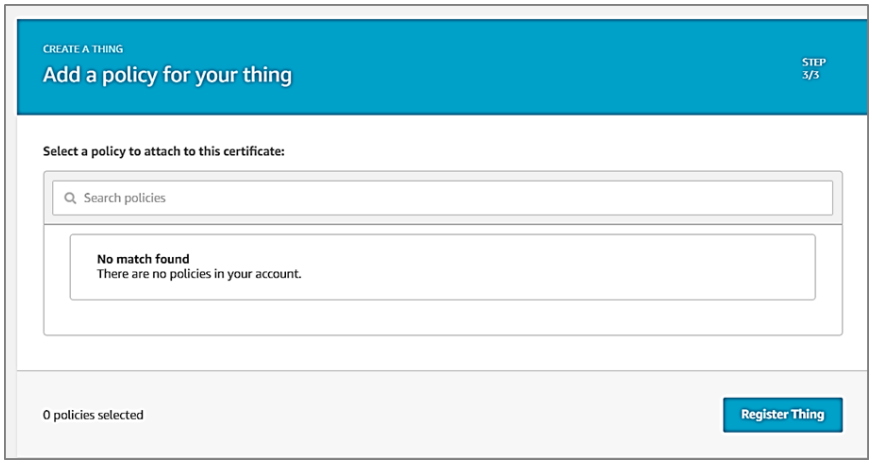
e) Create a working directory called ca2 and download all the 4 files above in this ca2 directory, renaming them.



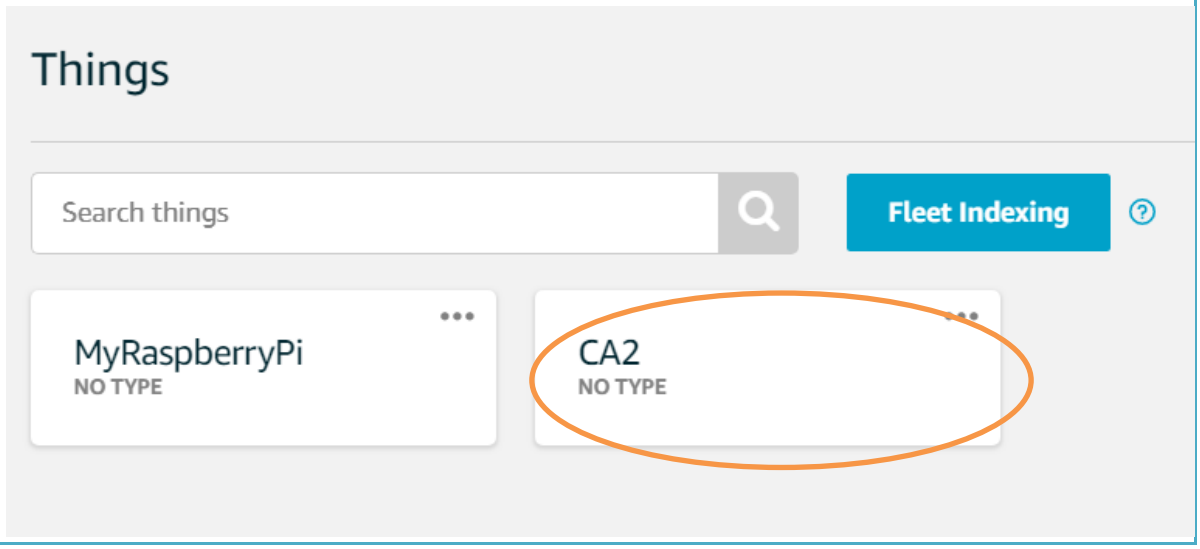
f) Next, click the “Activate” button. Almost immediately, you should see “Successfully activated certificate” and the Activate button changes to “Deactivate”



g) Click to the next page, and click “Register Thing”



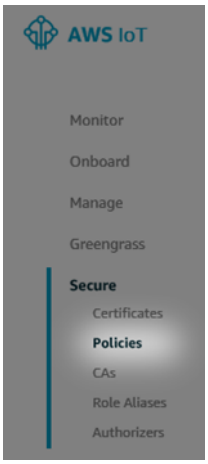
h) You are brought to this page. Continue with the next section to attach a security policy to your Thing.



C. Create a Security Policy for your RPi

Task

a) On the left IOT Core dashboard, select Policies under the Secure sub-menu



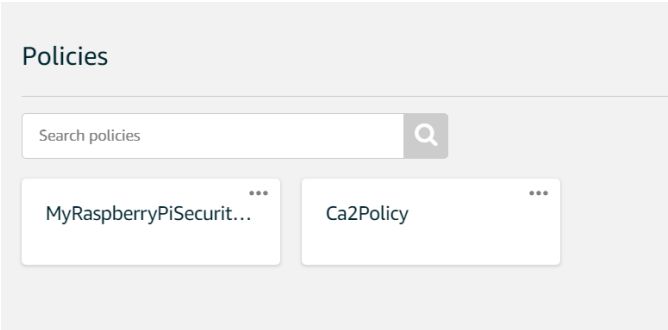
b) On the next page, choose "Create new policy"

c) On the Create a policy page, key in the following configuration

Field	Type this in
Name	Ca2Policy
Action	iot:*
Resource ARN	*
Allow	Checked

d) Click "**Create**".

You now have a Security Policy that allows all access to IOT Core services

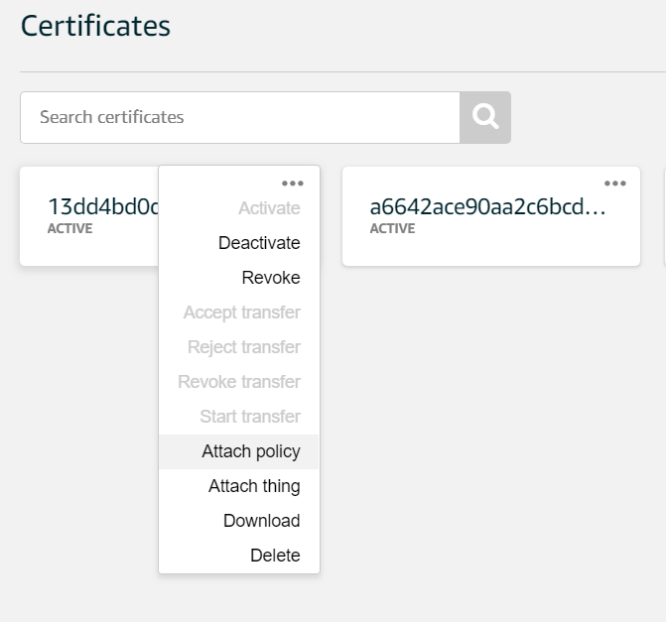


D. Attach Security Policy and Thing to your Cert

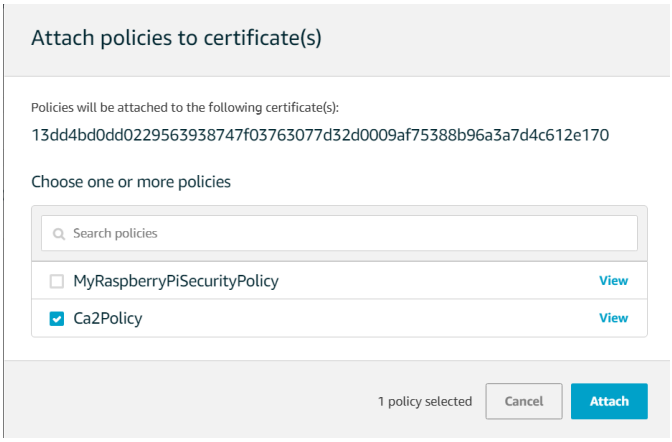
Task

a) On the left nav bar, click "Security, Certificates"

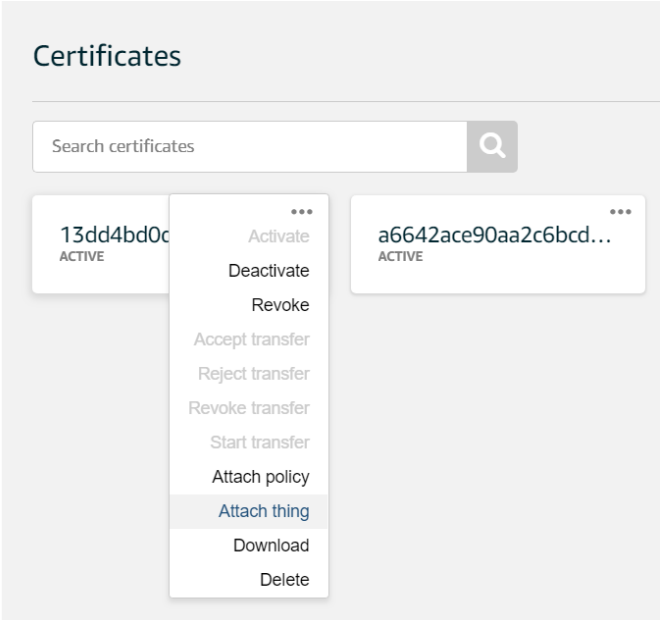
b) The X.509 certificate you created earlier is shown. Click the checkbox beside it, then click "Actions" button and choose "Attach Policy"



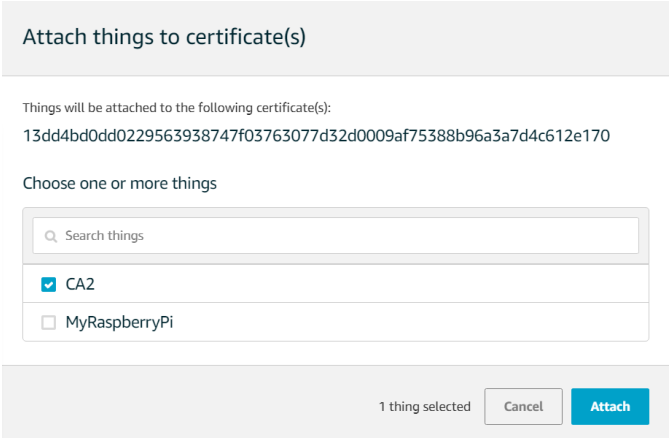
c) Check the “Ca2Policy” you created earlier and click “Attach” button.



d) Let’s attach the “Thing” to this certificate. Click “Actions” button and choose “Attach Thing”



e) 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

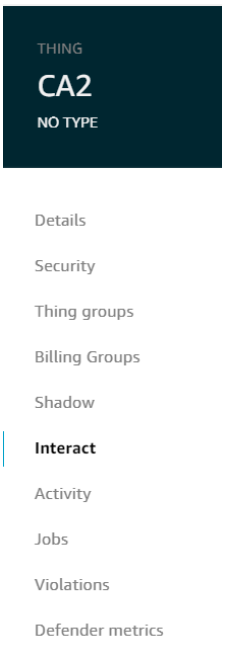


E. Copy REST API endpoint of your "Thing"

Task

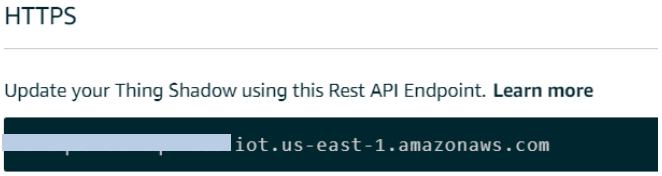
a) Click "Manage -> Things" and choose "CA2"

On the next screen, choose "Interact"



b) Copy and paste the REST API endpoint into a Notepad.

You will need this value later.

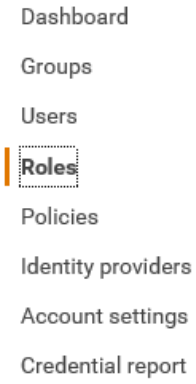


A. Create AWS Role

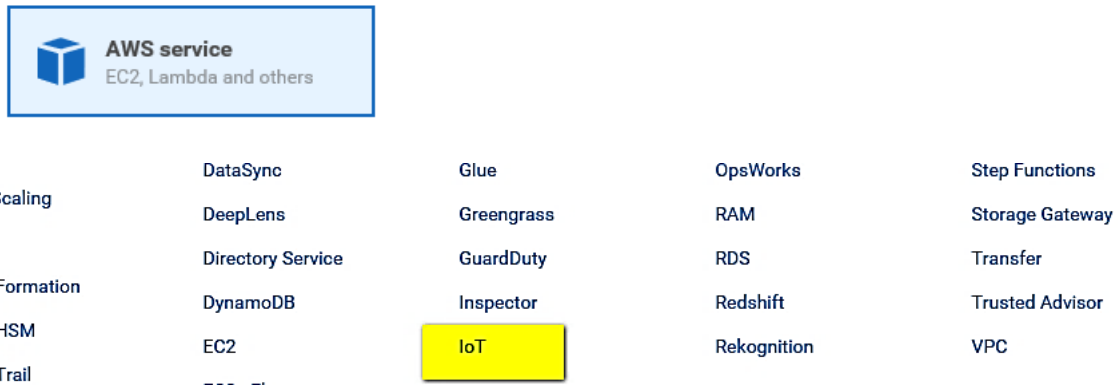
In this section, you will learn how to use a special feature known as **AWS IoT Rules** to take information from an incoming MQTT message and write it to a AWS Simple Notification Service (SNS) topic which thens sends email and SMS to you.

Task

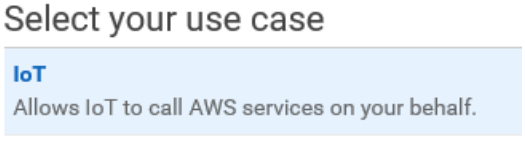
a) Search for the IAM service on AWS Console and choose “Roles”



b) Click “Create Role” and on the next page, choose “AWS service”, then “IOT”

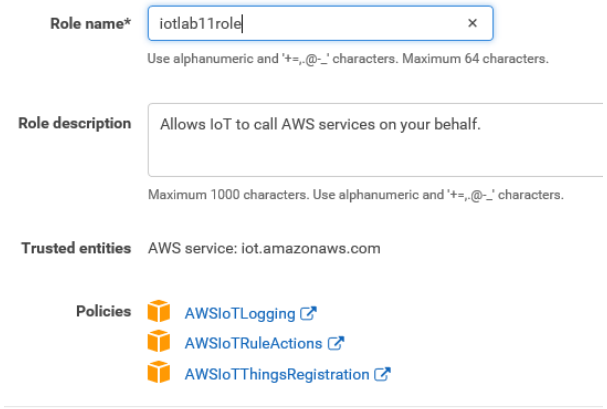


c) Under “Select your use case”, select IoT



d) Click “Next->Permissions”.
You will be brought to another page.Do not do anything on the new page, but just click “Next->Tags” You will be brought to another new page.Do not do anything. Just click “Next->Review”

e) You will see a page that requires you to input a name for your Role.

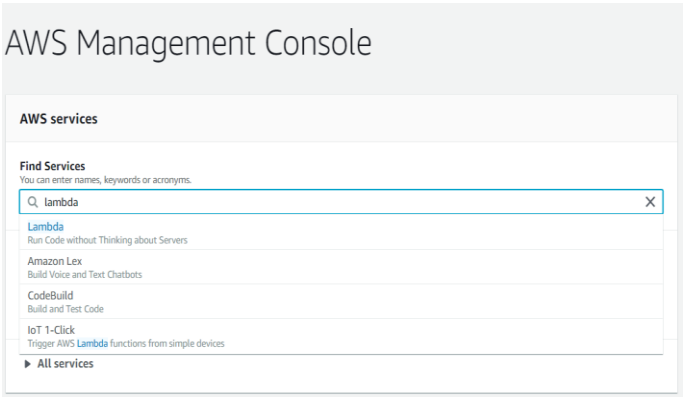


2.Create Lambda Function for Telegram Bot

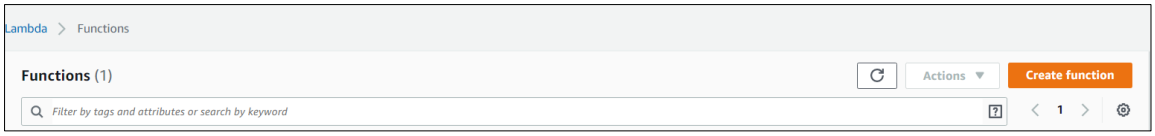
A. Create an Lambda Function

Task

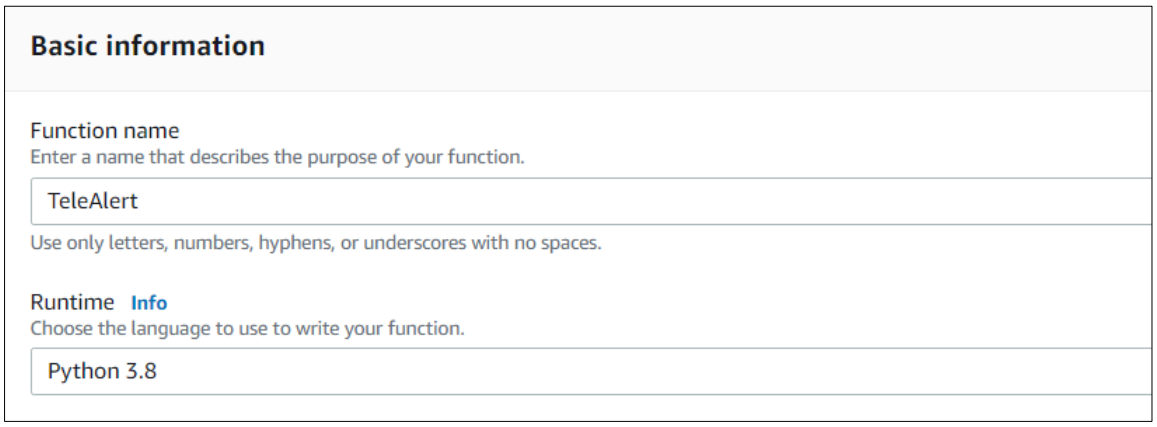
a) AWS Management Console. Select "Lambda"



b) Choose Create new Function

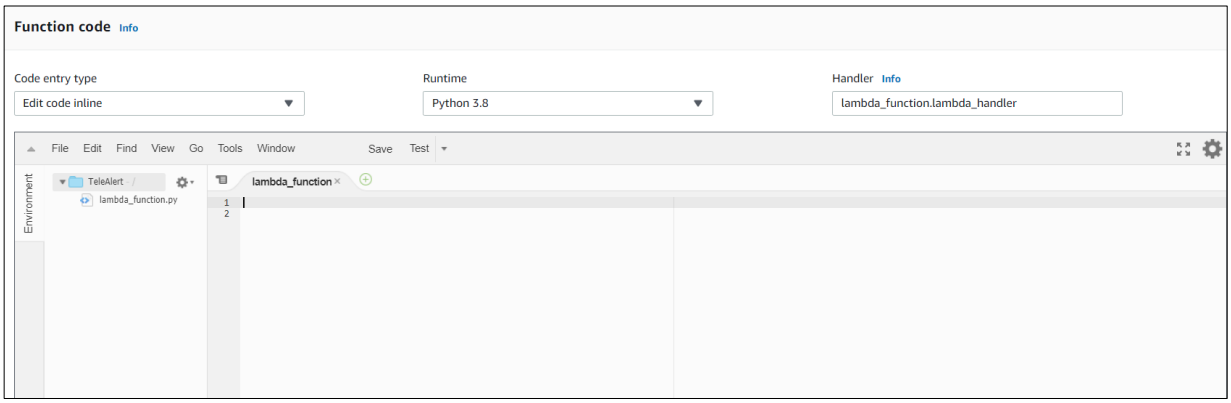


c) Type a Function name and choose "Python 3.8" for Runtime info.



d) This function executes the sendMessage method of the Telegram Bot API to forward SNS messages (notifications) to a Telegram chat.

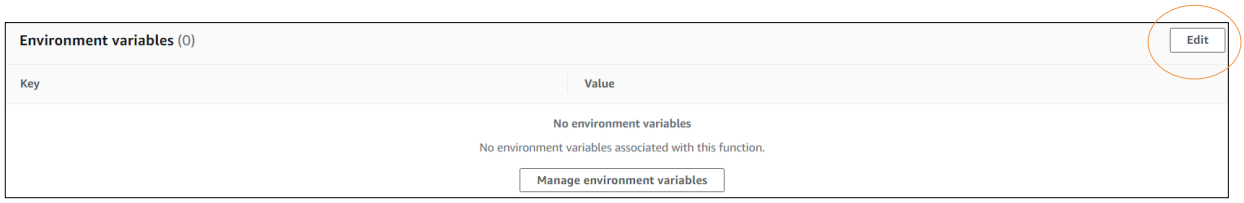
Input the following codes in the lambda function codes.



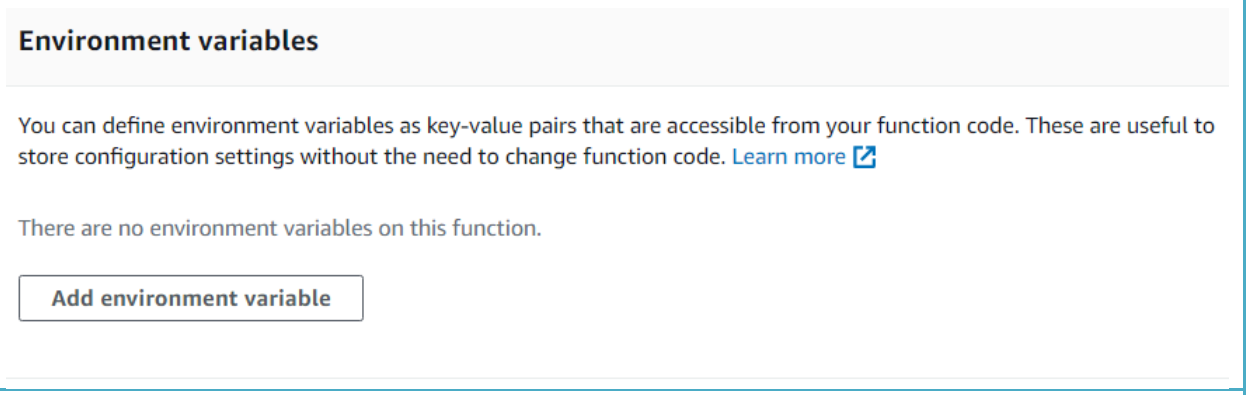
lambda_function Codes

```
import os
import json
from urllib import request, parse, error

def lambda_handler(event, context):
    url = 'https://api.telegram.org/bot%s/sendMessage' %
os.environ['TOKEN']
    message = event['Records'][0]['Sns']['Message']
    data = parse.urlencode({'chat_id': os.environ['CHAT_ID'], 'text':
message})

    try:
        # Send the SNS message (notification) to Telegram
        request.urlopen(url, data.encode('utf-8'))
    except error.HTTPError as e:
        print('Failed to send the SNS message below:\n%s' % message)
        response = json.load(e)
        if 'description' in response:
            print(response['description'])
        raise e
```

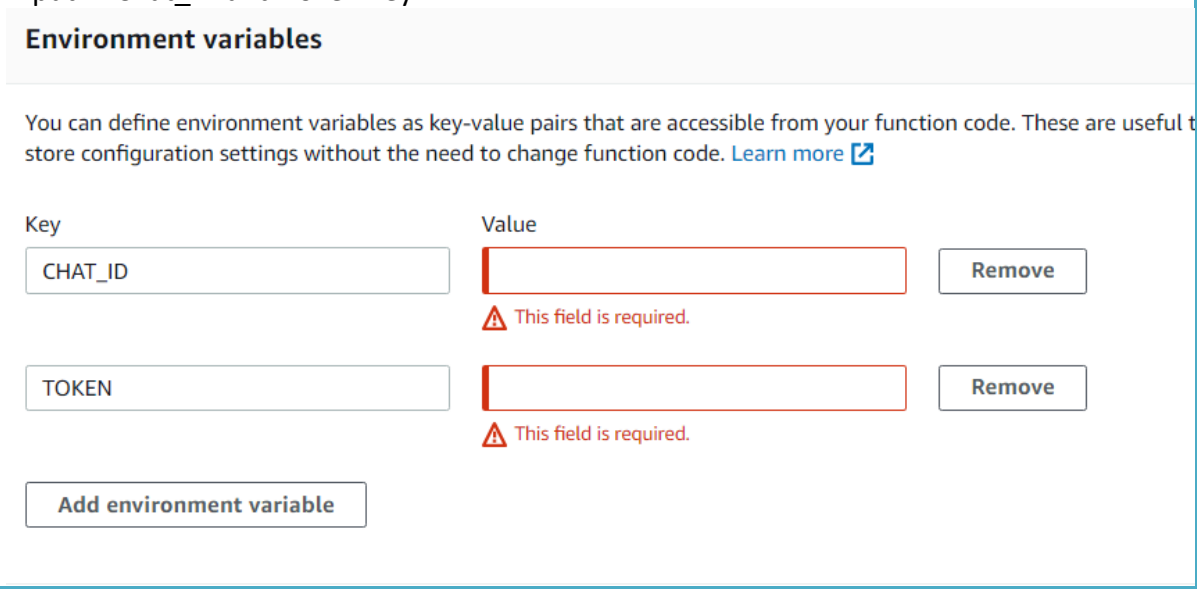
e) Under Environment variables select on “Edit”



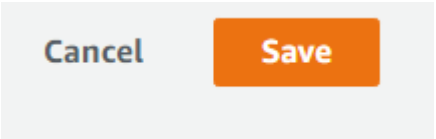
f) Select on “Add environment variable”



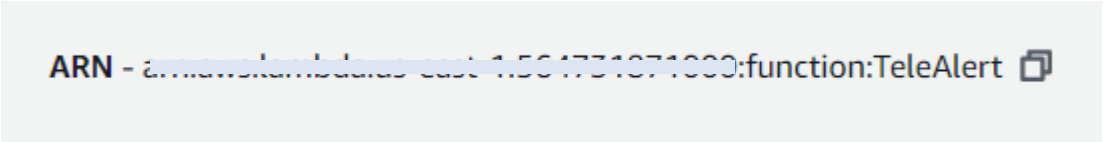
g) Input in Chat_ID and Token key.



h) Click on “Save”



i) Copy the ARN Topic which is at the top right . It will be need to create SNS subscription.

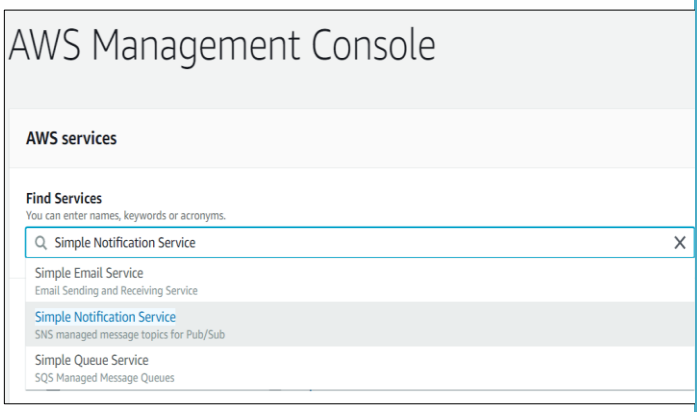


3.Set up SNS

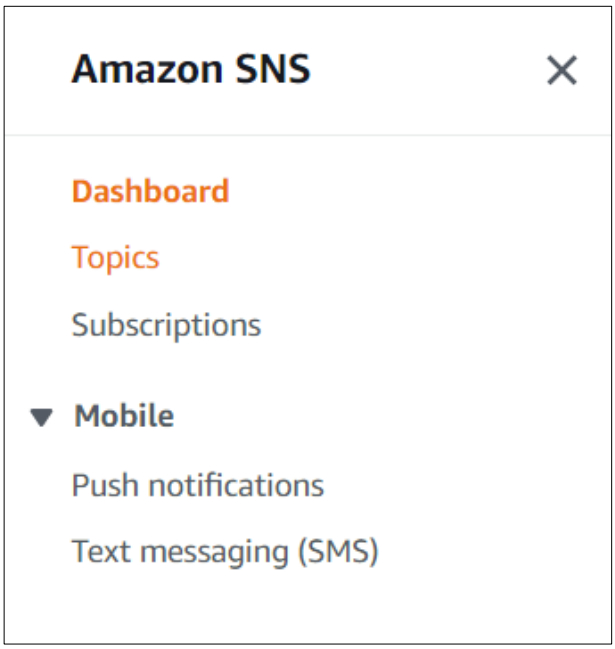
B. Create an AWS SNS Topic for EMAIL

Task

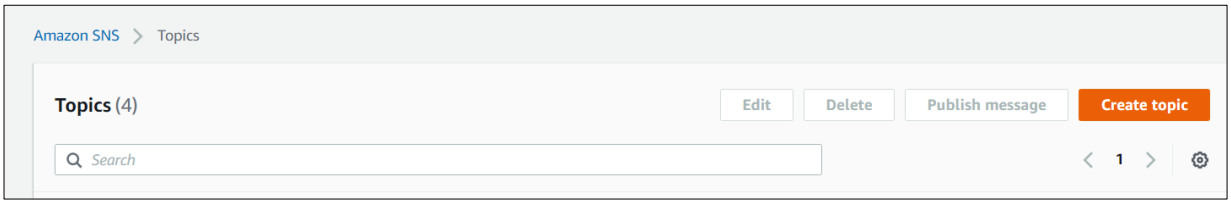
a) AWS Management Console. Select "Simple Notification Service"



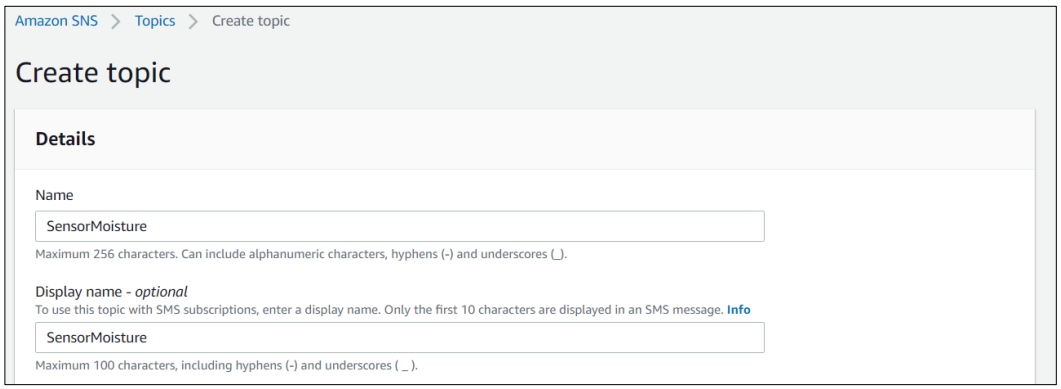
b) Choose Topics



c) Choose Create new topic



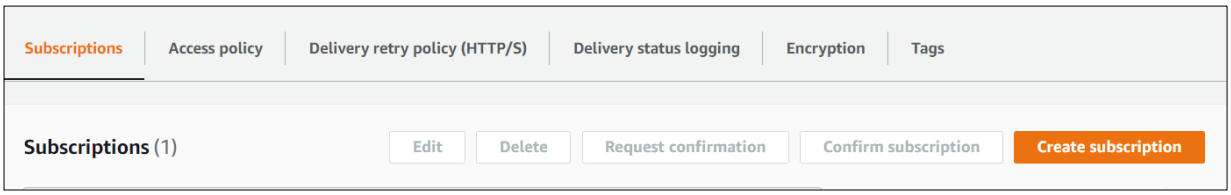
d) Type a topic name and a display name, and then choose Create topic.



C. Subscribe to an Amazon SNS Topic for EMAIL

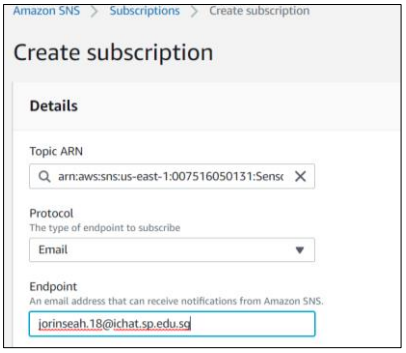
Task

a) In the Amazon SNS console, select the check box next to the topic you just created. From the Actions menu, choose Subscribe to topic.



b) On Create subscription, from the Protocol drop-down list, choose EMAIL.

In the Endpoint field, type your EMAIL ADDRESS and then choose Create subscription

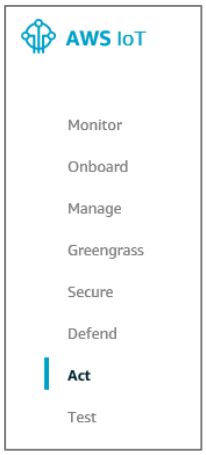


D. Create IoT Rule to send EMAIL

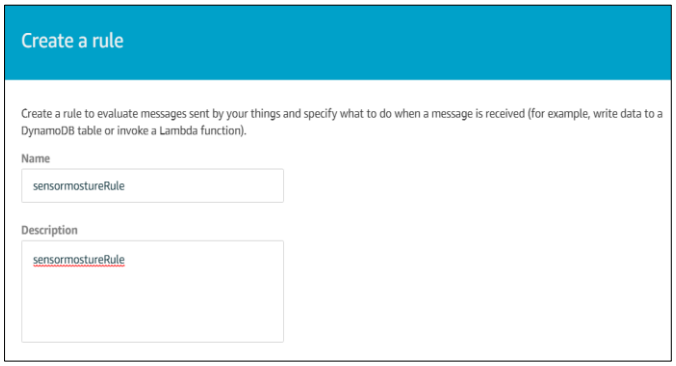
The AWS IoT rules engine listens for incoming MQTT messages that match a rule. When a matching message is received, the rule takes some action with the data in the MQTT message. In this step, you will create and configure a rule to send the data received from a device to an Amazon SNS topic.

Task

a) In the AWS IoT console, in the left navigation pane, choose “Act”, then “Create a rule”

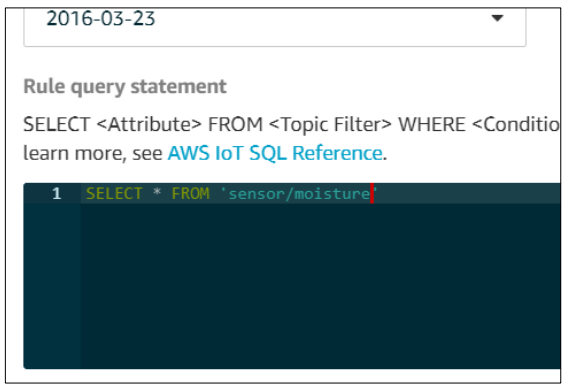


b) On the Create a rule page, in the Name field, type a name for your rule. In the Description field, type a description for the rule.

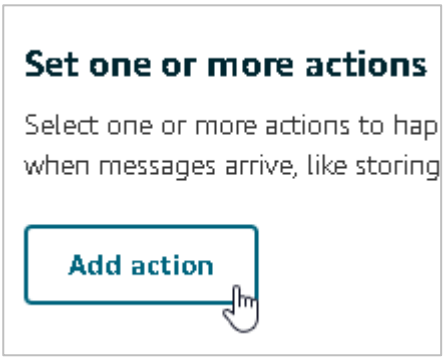


c) Scroll down to Rule Query statement.

Type
SELECT * FROM 'sensors/moisture'

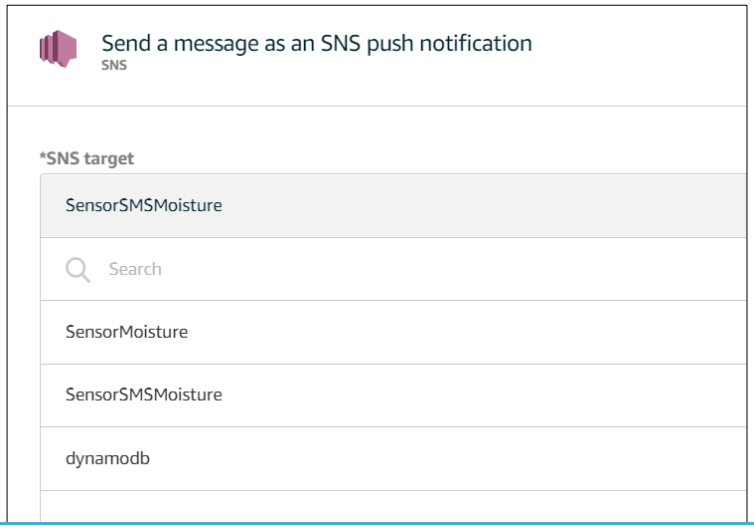


d) In Set one or more actions, choose Add action.

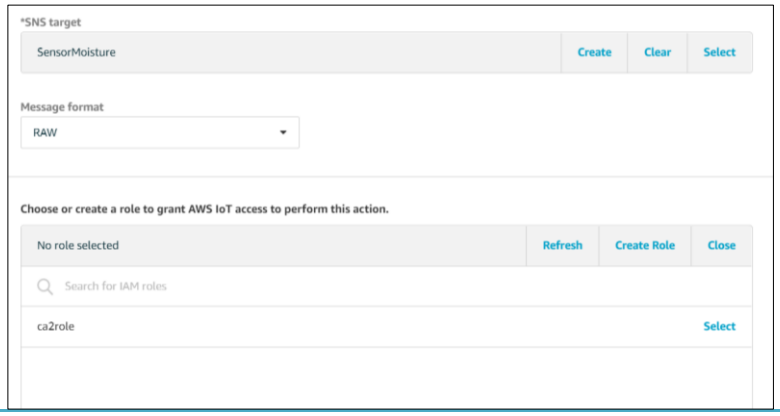


e) On the Select an action page, select Send a message as an SNS push notification, and then choose Configure action.

f) On the Configure action page, from the SNS target drop-down list, choose the Amazon SNS topic you created earlier.

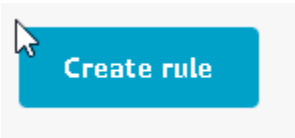


g) Choose ca2role to grant AWS IoT access to perform this action.



h) Choose Add action.

i) On the Create a Rule page, choose Create rule.

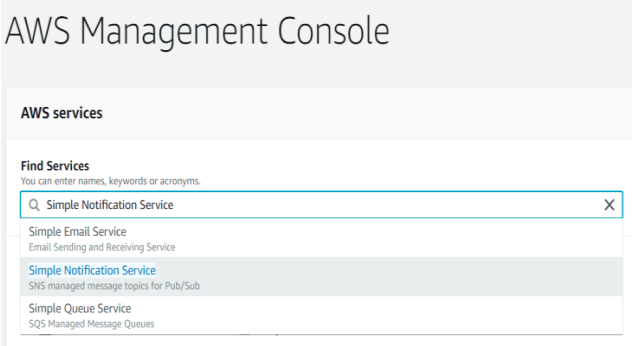


j) On the Overview page for the rule, choose the left arrow to return to the AWS IoT dashboard.

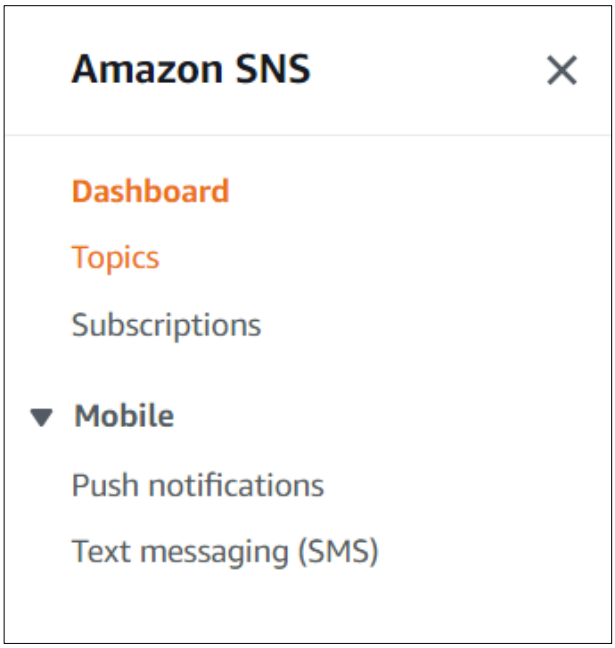
E. Create an AWS SNS Topic for Lambda Function

Task

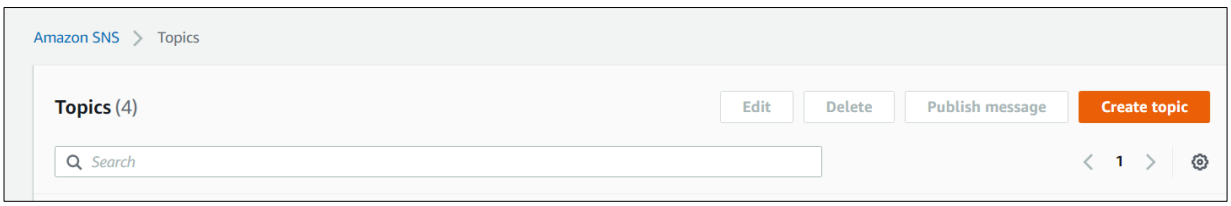
b) AWS Management Console. Select "Simple Notification Service"



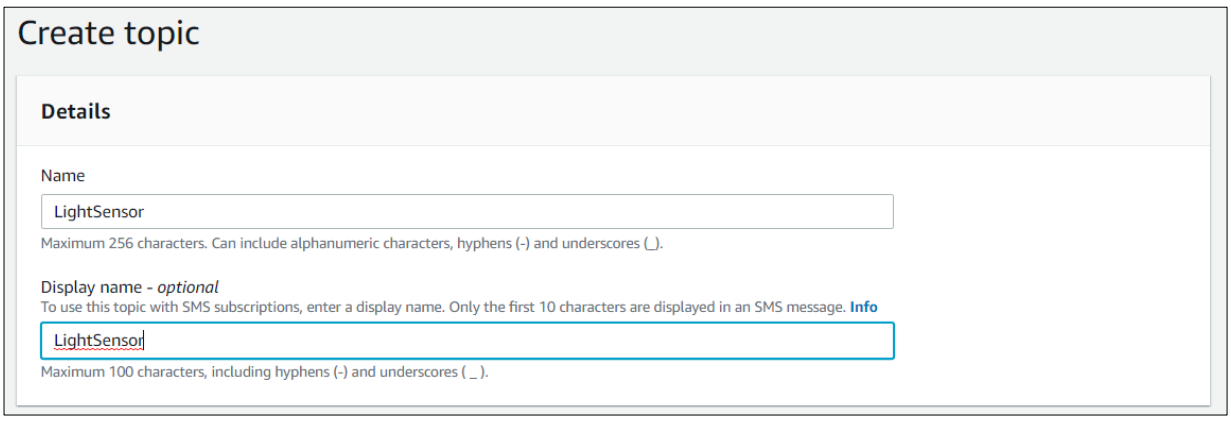
e) Choose Topics



f) Choose Create new topic



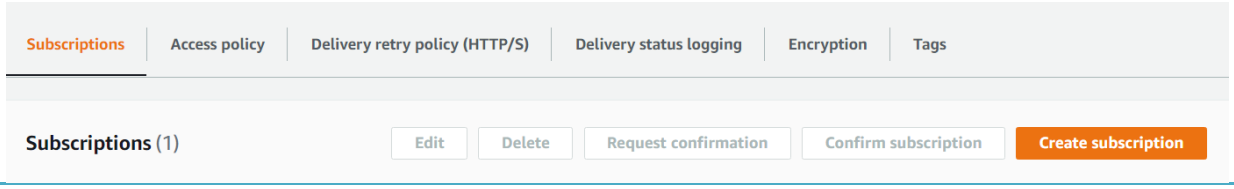
g) Type a topic name and a display name, and then choose Create topic.



F. Subscribe to an Amazon SNS Topic for Lambda Function

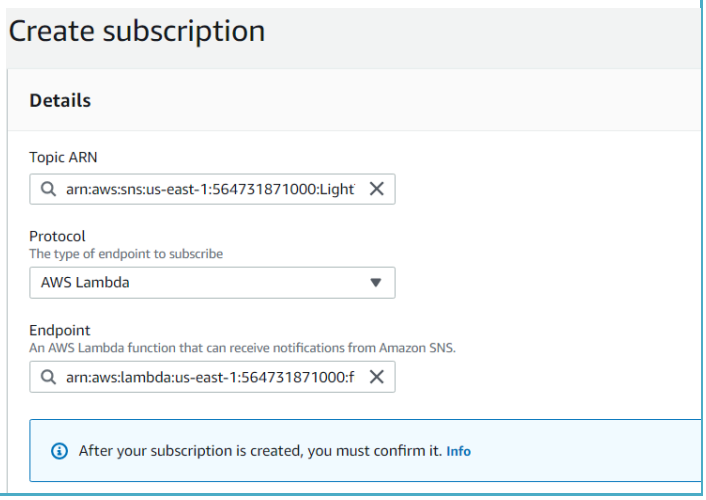
Task

a) In the Amazon SNS console, select “Create subscription”.



b) On Create subscription, from the Protocol drop-down list, choose AWS Lambda.

In the Endpoint field, paste in the ARN topic that you copy earlier in Lambda function.

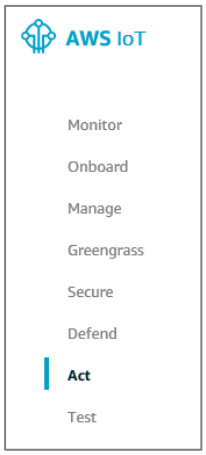


G. Create IoT Rule for Lambda to send Telegram alert

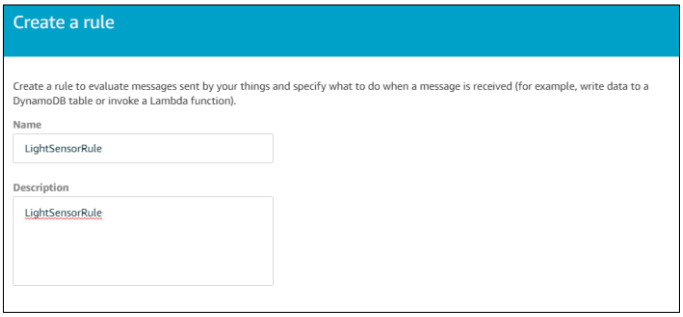
The AWS IoT rules engine listens for incoming MQTT messages that match a rule. When a matching message is received, the rule takes some action with the data in the MQTT message. In this step, you will create and configure a rule to send the data received from a device to an Amazon SNS topic.

Task

a) In the AWS IoT console, in the left navigation pane, choose “Act”, then “Create a rule”

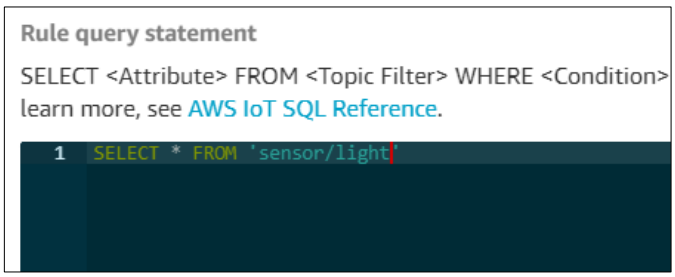


j) On the Create a rule page, in the Name field, type a name for your rule. In the Description field, type a description for the rule.

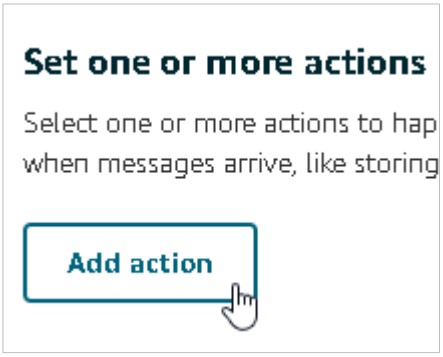


k) Scroll down to Rule Query statement.

Type
SELECT * FROM 'sensor/light'

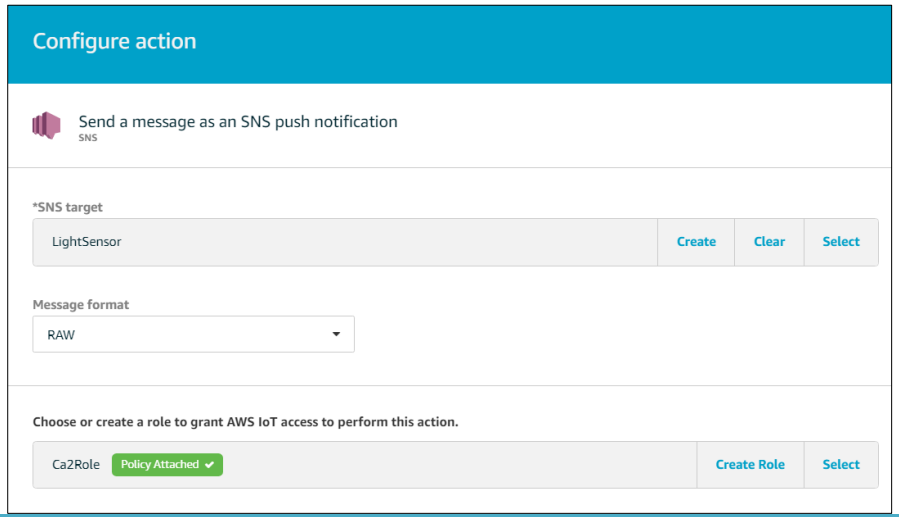


l) In Set one or more actions, choose Add action.

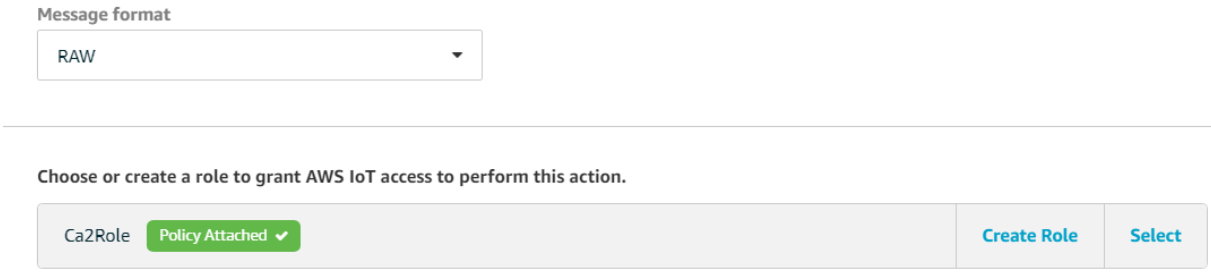


m) On the Select an action page, select Send a message as an SNS push notification, and then choose Configure action.

n) On the Configure action page, from the SNS target drop-down list, choose the Amazon SNS topic you created earlier.



a) Choose ca2role to grant AWS IoT access to perform this action.



o) Choose Add action.

p) On the Create a Rule page, choose Create rule.

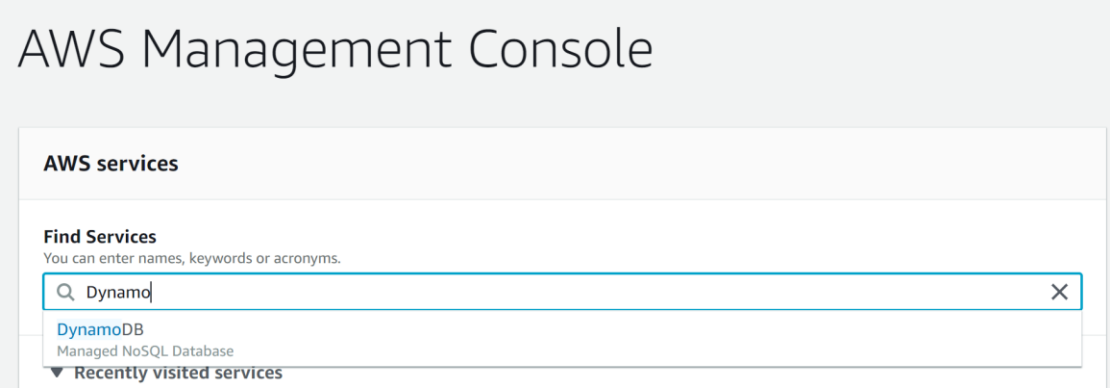
q) On the Overview page for the rule, choose the left arrow to return to the AWS IoT dashboard.

4.Setting up DynamoDB

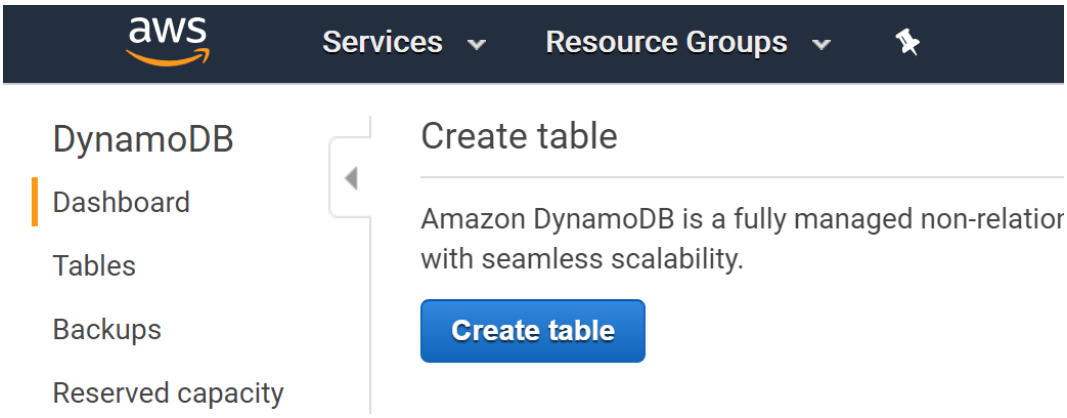
A. Create a DynamoDB table

Task

a) Go back to the AWS Management Console and search for "DynamoDB"



b) Click on the "Create table" button to create a new table.

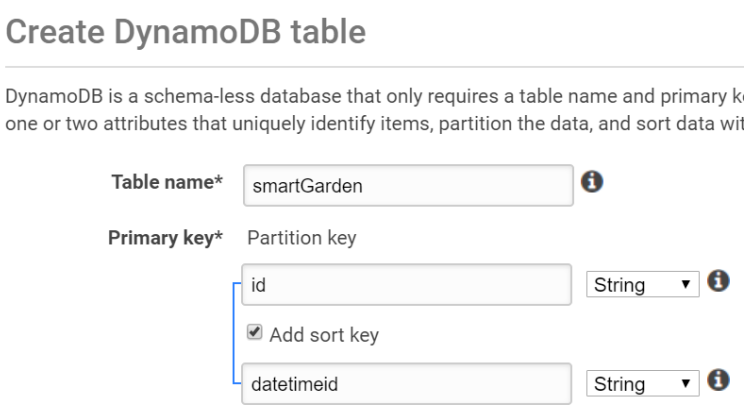


c) Create 3 tables using the attributes as shown below:

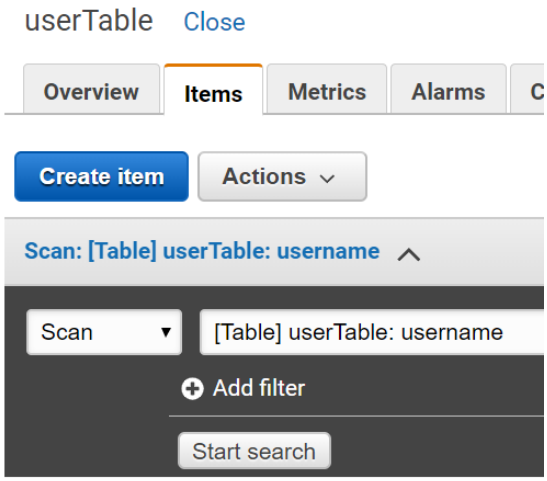
Table name	Partition key	Sort key
smartGarden	id	datetimeid
userTable	username	
statusTable	id	datetimeid
picUrlTable	id	datetimeid

Task

d) It will look like this:

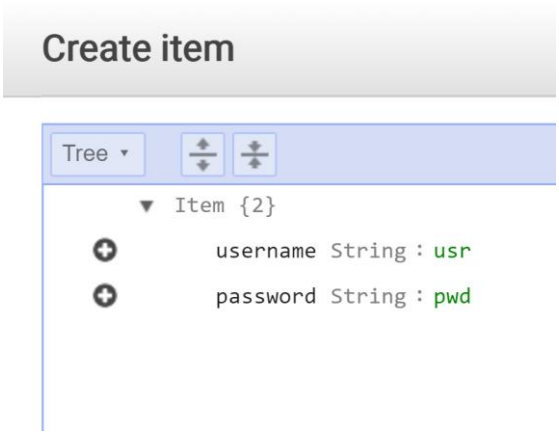


e) After creating the userTable table, click on "Create item" to create a new user for the web page.



f) Create a new user by typing in your desired username and password. For this example we will create a user with the following credentials:

username: usr
password: pwd



g) The item is then created and will be shown in the table:

Task

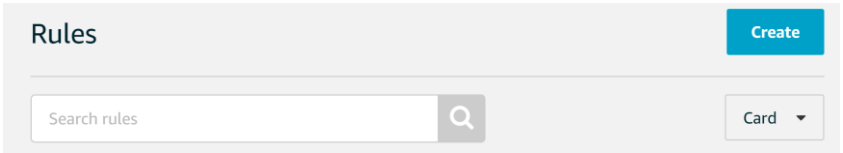
username	password
usr	pwd

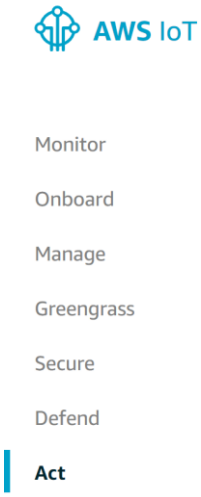
B. Create rule to publish MQTT message to DB

In this step, you will create and configure a rule to send the data received from a device to the AWS DynamoDB table you have created in Step A of this section.

Task

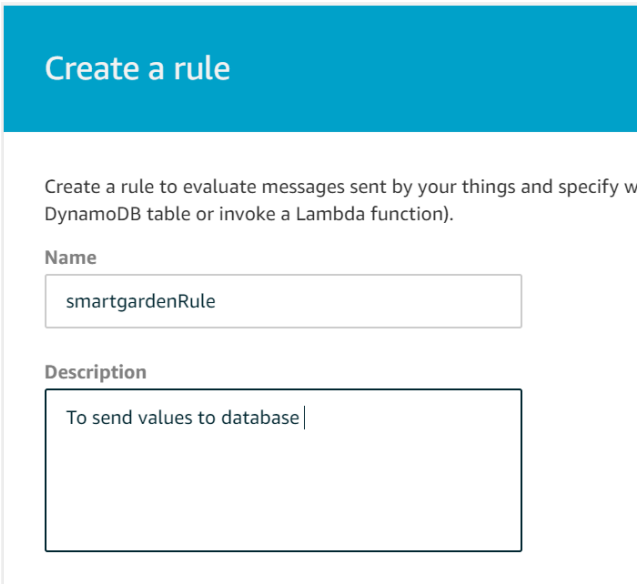
a) Going back to the AWS IoT console, in the left navigation bar, click on “Act”. Then click “Create”.



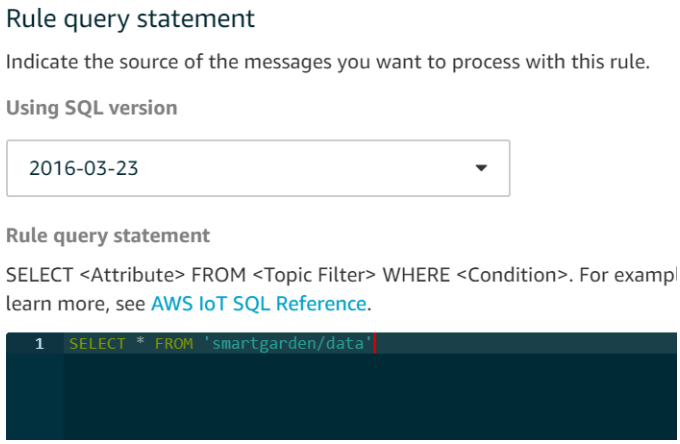


Task

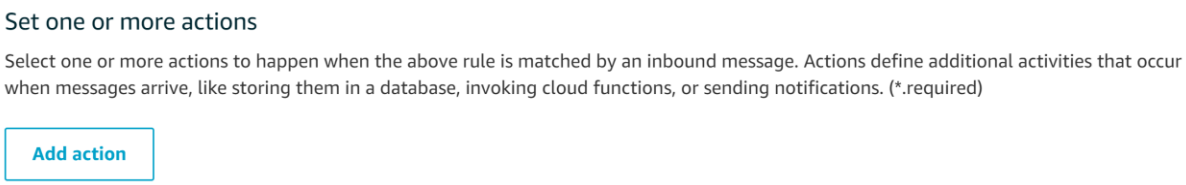
b) Type in a name and short description for your rule. Over here, we named it "smartgardenRule", this will be for the "smartGarden" database.



c) For the Rule query statement, you should select the latest SQL version and type in "SELECT * FROM 'smartgarden/data'" in the query statement box.



d) In Set one or more actions, choose Add action.



Task
e) Select the action to “Split message into multiple columns of a DynamoDB table (DynamoDBv2)” and click “Configure action”.   
f) On the Configure action page, choose the DynamoDB table you created earlier.   
g) If you are using a **AWS Paid account**, click “Create a new role”. Then select the newly created role and click “Update role”. If you are using a **AWS Educate account**, you will not be able to create a new role. Instead, just choose the one you created in part A (ca2role) from the drop-down list and click “Update Role” For example, we will be using the “ca2role” role.      Then click “Add action”.
h) This brings you back to the Create a Rule page. Click “Create rule” at the bottom right hand side of the screen to create the rule.
i) Create rules for the other tables as well, with the following fields:

Task

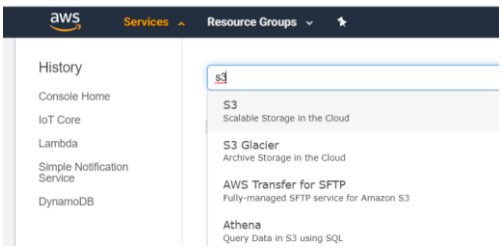
Name	Description	SQL Query Statement	For table
userTableRule	Rule to login	SELECT * FROM 'smartgarden/user'	userTable
statusTableRule	Rule to send motor status from DynamoDB	SELECT * FROM 'smartgarden/statusTable	statusTable
StorePlantImgUrl	Store the plant image url of s3 storage, to DynamoDB	SELECT * FROM 'smartgarden/imgurl/plant'	picURLTable

5.Setting up S3

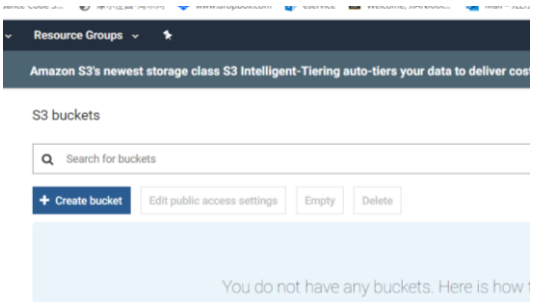
A. Create a bucket on Amazon S3

Task

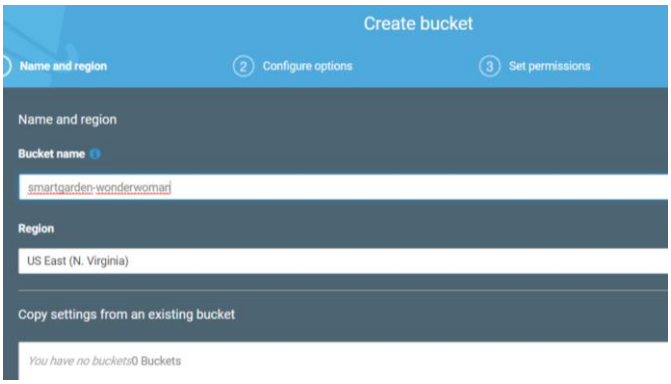
a) Log in your AWS console and search for S3



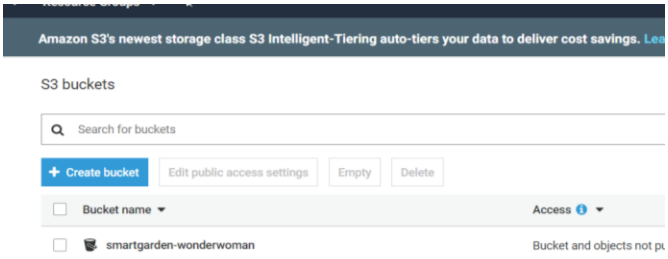
b) Click “Create bucket”



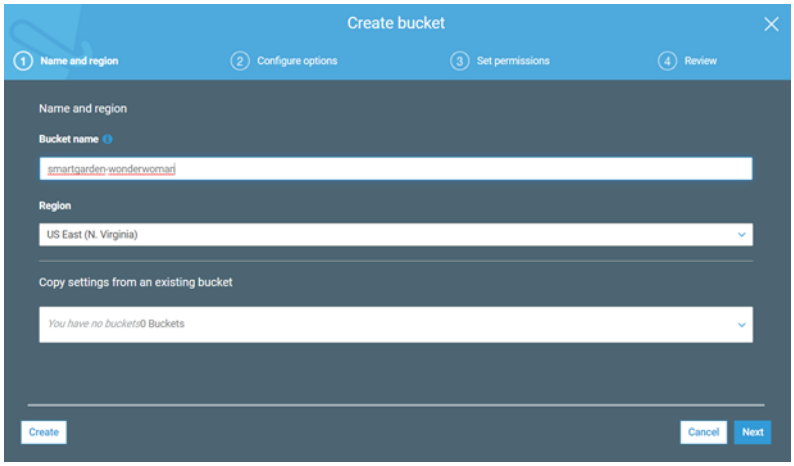
c) Type in a unique name for your bucket and choose Region as “US East (Virginia)” which is us-east-1 Click “Create” button



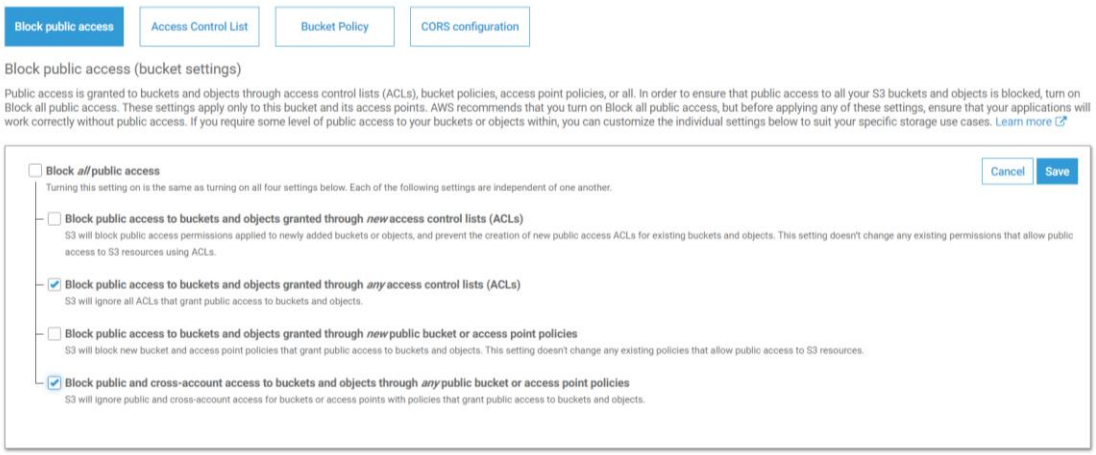
d) Your newly created bucket should appear in the list.



e) After creating the bucket, select " Set permission" .



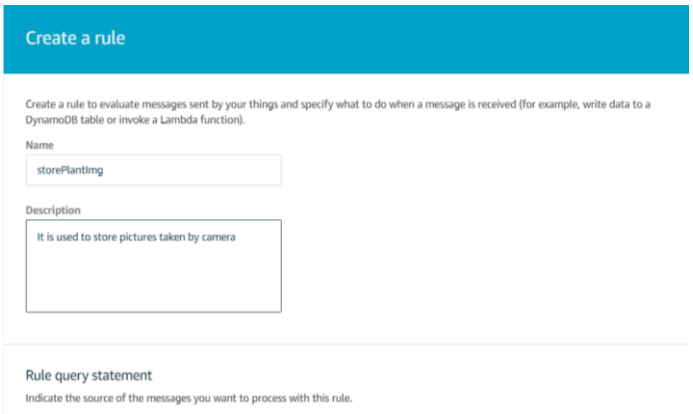
f) Only tick "Block public access to buckets and objects granted through any access control lists(ACLs) and "lock public and cross-account access to bucket and obkect through any public bucket or access point policies"



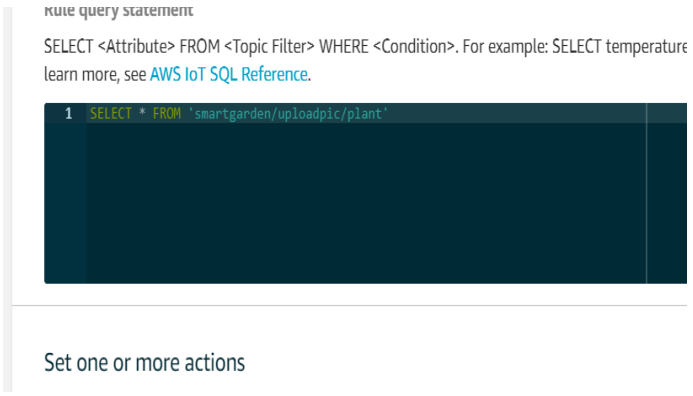
B. Create Rule for Amazon S3

Task

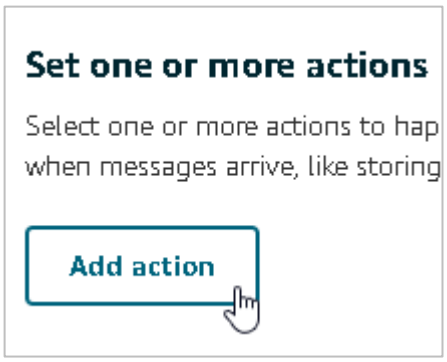
a) On the Create a rule page, in the Name field, type a name for your rule. In the Description field, type a description for the rule.



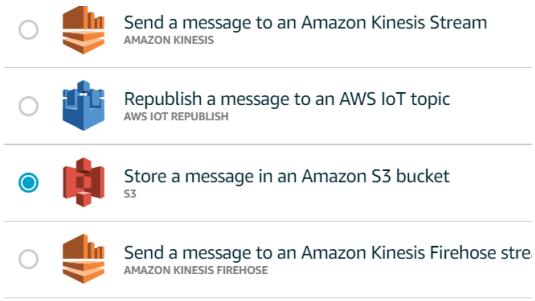
b) Scroll down to Rule Query statement.Type SELECT * FROM 'smartgarden/uploadpic/plant'



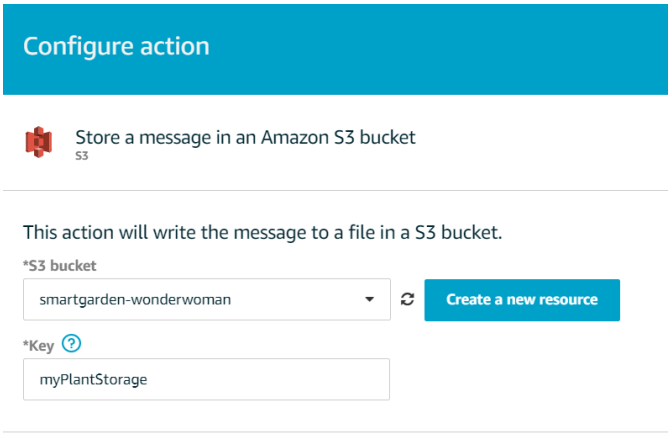
c) In Set one or more actions, choose Add action.



d) On the Select an action page, select Send a message as an S3 push notification, and then choose Configure action.



e) On the Configure action , in the S3 bucket, type a name. In the Key field, type a description for the key.



Section 5
Source codes

The highlighted parts of the code are the ones you have to change according to what you have created.

A. Certifications

	Task
a)	Transfer the certifications you saved earlier (Section 4.1 B) to the folder ¬/smartgarden in your RPi by using Filezilla.

B. Arduino Scripts

We will create an Arduino program that reads the values of the DHT11 sensor (temperature and humidity), LDR sensor (light values), and the soil moisture sensor.

Also the program will light up the red LED if the soil moisture level is too high (the higher it is , the drier the soil) and the yellow LED if the room is too dark. The buzzer will have a sound when the user taps valid card on the NFC card reader. The program will control the motor and automate the watering system.

It will send the values read to the RPi through serial communication to store it in the database, and retrieve back data from the RPi that will be used to control the motor.

	Task
a)	Open the **Arduino IDE** on the RPi and save the new file as **smartgarden.ino**. The file will be saved in the ~/sketchbook/smartgarden folder of your RPi.

smartGarden.ino

```
#include <dht.h>    // dht lib

dht DHT;    // initialise dht sensor
```

```
#define DHT11_PIN 7

int soilValue = 0;    // set soil moisture value to 0
int soilPin = A0;        // set soil sensor to A0
int chk;
float temp;
float hum;
int ldrValue;
int yellowLEDPin = 2;  // set yellow led to pin 12 (ldr)
int redLEDPin = 8;     // set red led to pin 13 (water)
int motorPin = 3;       // set motor to pin 3
int ldrPin = A5;
/* 'A': auto
   'M': manual
   'O': on
   'F': off
*/
char status;

/*NFC Card Reader */
#include <SPI.h>
#include <MFRC522.h>

#define SDAPIN 10
#define RESETPIN 9

byte FoundTag;
byte ReadTag;
byte TagData[MAX_LEN];
byte TagSerialNumber[5];
byte GoodTagSerialNumber[5] = {136, 4 ,239, 94, 61};

MFRC522 nfc(SDAPIN, RESETPIN);

//buzzer
#define buzzerPin 5

#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
```

```
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
```

```
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978

//Mario main theme melody
int melody[] = {
  NOTE_E7, NOTE_E7, 0, NOTE_E7,
  0, NOTE_C7, NOTE_E7, 0,
  NOTE_G7, 0, 0, 0,
  NOTE_G6, 0, 0, 0,

  NOTE_C7, 0, 0, NOTE_G6,
```

```
  0, 0, NOTE_E6, 0,
  0, NOTE_A6, 0, NOTE_B6,
  0, NOTE_AS6, NOTE_A6, 0,

  NOTE_G6, NOTE_E7, NOTE_G7,
  NOTE_A7, 0, NOTE_F7, NOTE_G7,
  0, NOTE_E7, 0, NOTE_C7,
  NOTE_D7, NOTE_B6, 0, 0,

  NOTE_C7, 0, 0, NOTE_G6,
  0, 0, NOTE_E6, 0,
  0, NOTE_A6, 0, NOTE_B6,
  0, NOTE_AS6, NOTE_A6, 0,

  NOTE_G6, NOTE_E7, NOTE_G7,
  NOTE_A7, 0, NOTE_F7, NOTE_G7,
  0, NOTE_E7, 0, NOTE_C7,
  NOTE_D7, NOTE_B6, 0, 0
};

int tempo[] = {
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  9, 9, 9,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,

  9, 9, 9,
  12, 12, 12, 12,
  12, 12, 12, 12,
  12, 12, 12, 12,
```

```
};
//end buzzer


void setup() {
  Serial.begin(9600);
  SPI.begin();
  nfc.begin();
  pinMode(redLEDPin, OUTPUT);
  pinMode(motorPin, OUTPUT);
  pinMode(ldrPin, INPUT);
  pinMode(yellowLEDPin, OUTPUT);
  delay (1000);
  }

void loop() {
  // Receive data from server
  if (Serial.available() ) {
    status = Serial.read();
  }

  chk = DHT.read11(DHT11_PIN);
  temp = DHT.temperature;
  hum = DHT.humidity;
  soilValue = analogRead(soilPin);
  ldrValue = analogRead(ldrPin);

  Serial.println(temp);
  Serial.println(hum);
  Serial.println(soilValue);
  Serial.println(ldrValue);

  if (status == 'A') {
    if (soilValue > 500) {
      analogWrite(motorPin, 220);
      digitalWrite(redLEDPin, HIGH);
    } else {
      digitalWrite(redLEDPin, LOW);
      analogWrite(motorPin, LOW);
    }
  } else if (status == 'M' || status == 'F') {
    if (soilValue > 500) {
      analogWrite(motorPin, LOW);
      digitalWrite(redLEDPin, HIGH);
    } else {
```

```
      digitalWrite(redLEDPin, LOW);
      analogWrite(motorPin, LOW);
    }
  } else if (status == 'O') {
    if (soilValue > 500) {
      digitalWrite(redLEDPin, HIGH);
    } else {
      digitalWrite(redLEDPin, LOW);
    }
    analogWrite(motorPin, 220);
  } else {
    if (soilValue > 500) {
      digitalWrite(redLEDPin, HIGH);
    } else {
      digitalWrite(redLEDPin, LOW);
    }
    analogWrite(motorPin, LOW);

    if (ldrValue<=700) {
      digitalWrite(yellowLEDPin, HIGH);
    } else {
      digitalWrite(yellowLEDPin, LOW);
    }

  }

  String GoodTag="False";
  FoundTag = nfc.requestTag(MF1_REQIDL, TagData);

  if (FoundTag == MI_OK) {
    ReadTag = nfc.antiCollision(TagData);
    memcpy(TagSerialNumber, TagData, 4);

    for(int i=0; i < 4; i++){
      if (GoodTagSerialNumber[i] != TagSerialNumber[i]) {
        break;
      }
      if (i == 3) {
        GoodTag="TRUE";
      }
    }
    if (GoodTag == "TRUE"){
        Serial.println("Wonderwoman");
        tone(5, 2000, 100);
    }
```

```
    else {
        Serial.println("Guest");
        tone(5, 100, 50);
    }

  }
  else {
  Serial.println("NoCard");
  }
  delay(6200);
}
```

C. Aws_pubsub scripts

Next, create aws_pubsub scripts which are aws_pubsub_data.py and aws_pubsub_status.py which will be used to send the data from the sensors to the database, and receive the status of the motor controlled by the web server from the database.

Aws_pubsub_data.py

```
# Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import serial
from rpi_lcd import LCD
from time import sleep
import datetime as datetime
import json
from gpiozero import MCP3008
import sys
import boto3
from boto3.dynamodb.conditions import Key, Attr
import datetime as dt
from twilio.rest import Client

# Get serial to fetch data from arduino
ser = serial.Serial('/dev/ttyUSB0', 9600)
lcd = LCD()

# Your Account Sid and Auth Token from twilio.com/console
account_sid = 'Your Account SID Token'
auth_token = 'Your Auth Token'
client = Client(account_sid, auth_token)

def customCallback(client, userdata, message):
  print("Received a new message: ")
  print(message.payload)
  print("from topic: ")
```

```
  print(message.topic)
  print("--------------\n\n")

host = " YOUR REST API ENDPOINT"
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

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

# Connect and subscribe to AWS IoT
my_rpi.connect()
my_rpi.subscribe("smartgarden/user", 1, customCallback)
my_rpi.subscribe("smartgarden/data", 1, customCallback)
my_rpi.subscribe("sensor/moisture", 1, customCallback)
my_rpi.subscribe("sensor/light", 1, customCallback)
sleep(2)

# Publish to the same topic in a loop forever
loopCount = 0
update = True
while update:
  try:
    temp = ser.readline().strip('\r\n')
    hum = ser.readline().strip('\r\n')
    soil = ser.readline().strip('\r\n')
    light = ser.readline().strip('\r\n')
    cardUser = ser.readline().strip('\r\n')


    temp = float(temp)
    hum = float(hum)
    soil = float(soil)
    light = float(light)

    print("Temp: {}" .format(temp))
    print("Humidity: {}" .format(hum))
    print("Soil: {}" .format(soil))
    print("light: {}" .format(light))
    print("card user is {}" .format(cardUser))

    # When user used valid card, it will display a welcome message
    if cardUser != "NoCard":
       lcd.text('Welcome', 1)
       lcd.text('{}' .format(cardUser), 2)
       messageTosend="Hi {}, now the temperature is {}, humidity is {}, soil moistur
e is {}, light level is {}".format(cardUser, temp, hum, soil, light)
       message = client.messages.create(body=messageTosend,from_='whatsapp:+14155238
886',to='whatsapp:+6594235289')
       sleep(2)
       lcd.clear()

    lcd.text('Temp: {}'.format(temp), 1)
```

```
    lcd.text('Humidity: {}'.format(hum), 2)
    sleep(1)
    lcd.clear()

    lcd.text('Soil: {}'.format(soil), 1)
    lcd.text('Light: {}'.format(light), 2)
    sleep(2)
    lcd.clear()

    loopCount = loopCount+1
    sleep(1)
    print('This is {} record'.format(loopCount))
    print()

    message = {}
    message["id"] = "id_smartgarden"
    now = datetime.datetime.now()
    message["datetimeid"] = now.isoformat()
    message["temperature"] = temp
    message["humidity"] = hum
    message["soil"] = soil
    message["light"] = light
    my_rpi.publish("smartgarden/data", json.dumps(message), 1)

    if soil >= 700:
      msg = "Your plant is too dry now, the soil moisture value is " + str(soil) + "
. Please on the water pump!"
      my_rpi.publish("sensor/moisture", msg, 1)

    if light < 600:
      msg = "There is not enough light for the plant, the light level value is " + s
tr(light) + "."
      my_rpi.publish("sensor/light", msg, 1)

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

Aws_pubsub_status.py

```
# Import SDK packages
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import boto3
from boto3.dynamodb.conditions import Key, Attr
import serial
from time import sleep
import datetime as dt
from datetime import date

# Get serial to fetch data from arduino
ser = serial.Serial('/dev/ttyUSB0', 9600)

def customCallback(client, userdata, message):
  print("Received a new message: ")
  print(message.payload)
  print("from topic: ")
```

```
  print(message.topic)
  print("--------------\n\n")

host = " YOUR REST API ENDPOINT"
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

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

# Connect and subscribe to AWS IoT
my_rpi.connect()
my_rpi.subscribe("smartgarden/statusTable", 1, customCallback)
sleep(1)

# Publish to the same topic in a loop forever
loopCount = 0
while True:
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('statusTable')
    response = table.query(KeyConditionExpression=Key('id').eq('id_status'),ScanInde
xForward=False)
    items = response['Items']

    n=1
    data = items[:n]
    uStatus = data[0]['status']
    status = uStatus.encode('latin-1')
    print(status)
    ser.write(status)
    sleep(2)
```

scripts.py

This script will allow you to run the other two scripts at the same time in one script.

```
from multiprocessing import Process

def script1():
  while True:
    import aws_pubsub_data

def script2():
  while True:
    import aws_pubsub_status

if __name__ == '__main__':
  print ('Running scripts...')
```

```
  proc1 = Process(target = script1)
  proc1.start()
  print ('Reading script running...')

  proc2 = Process(target = script2)
  proc2.start()
  print ('Status script running...')

  print ('Scripts running')
```

D. Telegram scripts

telegram_scripts.py

```
import boto3
import botocore
from time import sleep
import time,sys,picamera
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from time import sleep
import telepot
import dynamodb
import tweepy

bot_token = 'Your bot token'

CONSUMER_KEY = 'Your API Key'
CONSUMER_SECRET = 'Your API secret Key'
ACCESS_KEY = 'Your access token'
ACCESS_SECRET = 'Your access token secret'


auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.secure = True
auth.set_access_token(ACCESS_KEY, ACCESS_SECRET)
api = tweepy.API(auth)

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

host = " YOUR REST API ENDPOINT"
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

my_rpi = AWSIoTMQTTClient("leiPubSub")
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("smartgarden/uploadpic/plant", 1, customCallback)
my_rpi.subscribe("telegram/takepicCommand", 1, customCallback)

camera = picamera.PiCamera()

s3 = boto3.client('s3')
bucket_name = 'smartgarden-wonderwoman'
plantfolder = 'plant/'
flowerfolder = 'flower/'


def plant_pic():
    timestring = time.strftime("%Y-%m-%dT%H_%M_%S", time.gmtime())
    filename = 'plant'+timestring+'.jpg'
    camera.capture('/home/pi/Desktop/plant/'+filename)
    return filename



def upload_plant_to_s3(filename):
    try:
        response = s3.upload_file('/home/pi/Desktop/plant/'+filename, bucket_name,
plantfolder + filename, ExtraArgs={'ACL':'public-read'})
        print("Plant File {} uploaded" .format(filename))
        return filename
    except botocore.exceptions.ClientError as e:
        print(e)
        return False



def detect_labels(bucket, key, max_labels=10, min_confidence=60, region="us-east-
1"):
    print("Now is detecting ... {} in {}" .format(key, bucket))
    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 plant_detect(file_name, teleUser):
      upload_plant_to_s3(file_name)
      try:
        highestconfidence = 0
        best_bet_item = "Unknown"
        bug_list = ["Insect", "Honey Bee", "Invertebrate", "Bee", "Arachnid",
"Spider", "Tick", "Garden Spider", "Animal"]
```

```
        for label in detect_labels(bucket_name, plantfolder + file_name):
          print("{Name} - {Confidence}%".format(**label))
          if label["Name"] in bug_list:
             dynamodb.send_url("plant", file_name, teleUser)
             return True
          else:
             s3.delete_object(Bucket=bucket_name, Key=plantfolder+file_name)
        sleep(1)
      except botocore.exceptions.ClientError as e:
        print(e)
      return False




def flower_detect(teleUser):
      try:
        highestconfidence = 0
        best_bet_item = "Unknown"
        bug_list = ["Insect", "Honey Bee", "Invertebrate", "Bee", "Arachnid",
"Spider", "Tick", "Garden Spider", "Animal"]
        for label in detect_labels(bucket_name, flowerfolder + 'tempPic.jpg'):
          print("{Name} - {Confidence}%".format(**label))
          if label["Name"] in bug_list:
            timestring = time.strftime("%Y-%m-%dT%H_%M_%S", time.gmtime())
            filename = 'flower'+timestring+'.jpg'
            s3.download_file(bucket_name, flowerfolder + 'tempPic.jpg',
'/home/pi/Desktop/flower/flower.jpg')
            response = s3.upload_file('/home/pi/Desktop/flower/flower.jpg',
bucket_name, flowerfolder + filename, ExtraArgs={'ACL':'public-read'})
            sleep(1)
            dynamodb.send_url("flower", filename, teleUser)
            print("Flower File {} uploaded" .format(filename))
            return True
          else:
            s3.delete_object(Bucket=bucket_name, Key=flowerfolder + 'tempPic.jpg')
        sleep(1)
      except botocore.exceptions.ClientError as e:
        print(e)
      return False


def respondToMsg(msg):
    chat_id = str(msg['chat']['id'])
    command = msg['text']
    teleUser = 'guest'
    if chat_id == 'Your chat id':
        teleUser = 'Your name'
    elif chat_id == 'Another chat id':
        teleUser = 'Another name'
    else:
        print(chat_id)


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

    if command == 'plant':
        #my_rpi.publish("telegram/takepicCommand", "plant", 1)
        plant_file = plant_pic()
```

```
        bot.sendPhoto(chat_id, photo=open('/home/pi/Desktop/plant/'+plant_file,
'rb'))
        msg = "This is the plant uploaded by {}" .format(teleUser)
        api.update_with_media('/home/pi/Desktop/plant/'+plant_file, status=msg)
    if command == 'flower':
        my_rpi.publish("telegram/takepicCommand", "flower", 1)
        filename = 'flower.jpg'
        sleep(3)
        bot.sendMessage(chat_id, "Taking picture of flower now!")
        s3.download_file(bucket_name, flowerfolder + 'tempPic.jpg',
'/home/pi/Desktop/flower/'+filename)
        bot.sendPhoto(chat_id, photo=open('/home/pi/Desktop/flower/'+filename,
'rb'))
        msg = "This is the flower uploaded by {}" .format(teleUser)
        api.update_with_media('/home/pi/Desktop/flower/'+filename, status=msg)
    if command == 'detect':
        loop = 1
        noBug = True
        plantBug = False
        flowerBug = False
        while loop < 6:
          #detect plant bug
          if not plantBug:
            bot.sendMessage(chat_id, "Detecting plant now...({}/5)" .format(loop))
            plant_file = plant_pic()
            plant_bug = plant_detect(plant_file, teleUser)
            if plant_bug:
               noBug = False
               plantBug = True
               bot.sendMessage(chat_id, "Your plant is detected with a bug! Please
be alert!")
               bot.sendPhoto(chat_id,
photo=open('/home/pi/Desktop/plant/'+plant_file, 'rb'))

          sleep(3)

          #detect flower bug
          if not flowerBug:
            my_rpi.publish("telegram/takepicCommand", "flower", 1)
            sleep(5)
            bot.sendMessage(chat_id, "Detecting flower now...({}/5)" .format(loop))
            flower_bug = flower_detect(teleUser)
            if flower_bug:
               noBug = False
               flowerBug = True
               sleep(1)
               bot.sendMessage(chat_id, "Your flower is detected with a bug! Please
be alert!")
               bot.sendPhoto(chat_id,
photo=open('/home/pi/Desktop/flower/flower.jpg', 'rb'))

          loop = loop + 1

        if noBug:
          bot.sendMessage(chat_id, "There is no bug bothering your plant and flower!
Don't worry!")
        else:
          bot.sendMessage(chat_id, "Detection is completed!")

bot = telepot.Bot(bot_token)
```

```
bot.message_loop(respondToMsg)

while True:
    sleep(1)
```

Now, use another raspberry pi to control the telegram

tele_flower.py

```
import boto3
import botocore
from time import sleep
import time,sys,picamera
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from time import sleep


def customCallback(client, userdata, message):
  print("Received a new message: ")
  print(message.payload)
  print("from topic: ")
  print(message.topic)
  print("--------------\n\n")
  if message.payload == 'flower':
    upload_flower_to_s3()


host = " YOUR REST API ENDPOINT "
rootCAPath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

my_rpi = AWSIoTMQTTClient("jorinPubSub")
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("smartgarden/uploadpic/plant", 1, customCallback)
my_rpi.subscribe("telegram/takepicCommand", 1, customCallback)

camera = picamera.PiCamera()

s3 = boto3.client('s3')
bucket_name = 'smartgarden-wonderwoman'
flowerfolder = 'flower/'


def flower_pic():
    timestring = time.strftime("%Y-%m-%dT%H_%M_%S", time.gmtime())
    filename = 'flower.jpg'
    camera.capture('/home/pi/Desktop/flower/'+filename)
```

```
    return filename


def upload_flower_to_s3():
    try:
        filename = flower_pic()
        response = s3.upload_file('/home/pi/Desktop/flower/'+filename, bucket_name,
flowerfolder + 'tempPic', ExtraArgs={'ACL':'public-read'})
        print("File {} uploaded" .format(filename))
        return filename
    except botocore.exceptions.ClientError as e:
        print(e)
        return False


while True:
    sleep(1)
```

E. DynamoDB

dynamodb.py

```
import boto3
from boto3.dynamodb.conditions import Key, Attr
import datetime as dt
from datetime import date, timedelta


def login():
  try:
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('userTable')
    response = table.scan()
    items = response['Items']
    print(items)
    return items
  except:
    import sys
    print(sys.exc_info()[0])
    print(sys.exc_info()[1])
```

```
def get_data():
  try:
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('smartGarden')

    startdate = date.today().isoformat()
    response = table.query(KeyConditionExpression=Key('id').eq('id_smartgarden') & K
ey('datetimeid').begins_with(startdate),
        ScanIndexForward=False
    )

    items = response['Items']

    n=1 # get latest data
    data = items[:n]
    print(data)
    return data
  except:
    import sys
    print(sys.exc_info()[0])
    print(sys.exc_info()[1])



def get_chart_data():
  try:
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('smartGarden')

    startdate = date.today().isoformat()
    response = table.query(KeyConditionExpression=Key('id').eq('id_smartgarden') & K
ey('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])



def get_status():
  try:
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('statusTable')

    startdate = date.today().isoformat()
    response = table.query(KeyConditionExpression=Key('id').eq('id_status') & Key('d
atetimeid').begins_with(startdate),
        ScanIndexForward=False
    )

    items = response['Items']

    n=1
    data = items[:n]
    return data
  except:
    import sys
    print(sys.exc_info()[0])
    print(sys.exc_info()[1])


def get_all_status():
  try:
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('statusTable')

    startdate = date.today().isoformat()
    response = table.query(KeyConditionExpression=Key('id').eq('id_status') & Key('d
atetimeid').begins_with(startdate),
        ScanIndexForward=False
    )

    items = response['Items']

    n=10
    data = items[:n]
    return data
  except:
    import sys
    print(sys.exc_info()[0])
```

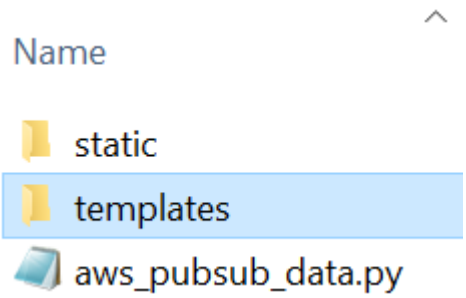
```
    print(sys.exc_info()[1])


def send_status(status):
  try:
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('statusTable')

    now = dt.datetime.now()
    new_item = {
      "id": "id_status",
      'datetimeid': now.isoformat(),
      'status': status
    }
    table.put_item(Item = new_item)

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




def send_url(type, fileName, user):
  try:
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('picUrlTable')

    now = dt.datetime.now()
    new_item = {
      "id": type,
      'datetimeid': now.isoformat(),
    'filename': fileName,
      'updatedBy': user
    }
    table.put_item(Item = new_item)

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


def get_Img_url(type):
  try:
```

```
    dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
    table = dynamodb.Table('picUrlTable')
    response = table.query(KeyConditionExpression=Key('id').eq(type)& Key('datetimei
d').begins_with("2020"),ScanIndexForward=False)
    items = response['Items']

    n=9 # limit to last 9 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__":
  query_data_from_dynamodb()
```

F. Creating html scripts

Task

1. Download the template here https://startbootstrap.com/themes/sb-admin-2/
2. Create 2 new folders called **templates** and **static** in your laptop inside the **~/smartgarden** folder.
3. Create the following html pages in the templates folder.
   - **login.html**
   - **index.html**
   - **charts.html**
   - **tables.html**
   - **gallery.html**



4. In the template you have downloaded, you will see that the CSS, Javascript etc codes are stored in 5 different folders as shown.
5. Copy all the 5 folders and paste it into your **static** folder.

login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Login Smart-Garden</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/boo
tstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></sc
ript>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></
script>
  <!-- Custom fonts for this template-->
  <link href="static/vendor/fontawesome-
free/css/all.min.css" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,4
00i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">

  <!-- Custom styles for this template-->
  <link href="static/css/sb-admin-2.min.css" rel="stylesheet">
 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></scr
ipt>
  <!-- Bootstrap core JavaScript-->
  <script src="static/vendor/jquery/jquery.min.js"></script>
  <script src="static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

  <!-- Core plugin JavaScript-->
  <script src="static/vendor/jquery-easing/jquery.easing.min.js"></script>

  <!-- Custom scripts for all pages-->
  <script src="static/js/sb-admin-2.min.js"></script>

  <!-- Page level plugins -->
  <script src="static/vendor/chart.js/Chart.min.js"></script>
```

```
  <!-- Page level custom scripts -->
  <script src="static/js/demo/chart-area-demo.js"></script>
  <script src="static/js/demo/chart-pie-demo.js"></script>

<style type="text/css">
  .login-form {
    width: 340px;
      margin: 50px auto;
  }
    .login-form form {
      margin-bottom: 15px;
        background: #f7f7f7;
        box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);
        padding: 30px;
    }
    .login-form h2 {
        margin: 0 0 15px;
    }
    .form-control, .btn {
        min-height: 38px;
        border-radius: 2px;
    }
    .btn {
        font-size: 15px;
        font-weight: bold;
    }


@font-face {
  font-family: Poppins-Regular;
  src: url('../fonts/poppins/Poppins-Regular.ttf');
}

@font-face {
  font-family: Poppins-Medium;
  src: url('../fonts/poppins/Poppins-Medium.ttf');
}

@font-face {
  font-family: Poppins-Bold;
  src: url('../fonts/poppins/Poppins-Bold.ttf');
}

@font-face {
  font-family: Poppins-SemiBold;
```

```
  src: url('../fonts/poppins/Poppins-SemiBold.ttf');
}

@font-face {
  font-family: Montserrat-Bold;
  src: url('../fonts/montserrat/Montserrat-Bold.ttf');
}

@font-face {
  font-family: Montserrat-SemiBold;
  src: url('../fonts/montserrat/Montserrat-SemiBold.ttf');
}

@font-face {
  font-family: Montserrat-Regular;
  src: url('../fonts/montserrat/Montserrat-Regular.ttf');
}


* {
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}

body, html {
  height: 100%;
}

a {
  font-family: Poppins-Regular;
  font-size: 14px;
  line-height: 1.7;
  color: #666666;
  margin: 0px;
  transition: all 0.4s;
  -webkit-transition: all 0.4s;
  -o-transition: all 0.4s;
  -moz-transition: all 0.4s;
}

a:focus {
  outline: none !important;
}
```

```
a:hover {
  text-decoration: none;
  color: #6675df;
}

h1,h2,h3,h4,h5,h6 {
  margin: 0px;
}

p {
  font-family: Poppins-Regular;
  font-size: 14px;
  line-height: 1.7;
  color: #666666;
  margin: 0px;
}

ul, li {
  margin: 0px;
  list-style-type: none;
}


input {
  outline: none;
  border: none;
}

textarea {
  outline: none;
  border: none;
}

textarea:focus, input:focus {
  border-color: transparent !important;
}

input:focus::-webkit-input-placeholder { color:transparent; }
input:focus:-moz-placeholder { color:transparent; }
input:focus::-moz-placeholder { color:transparent; }
input:focus:-ms-input-placeholder { color:transparent; }

textarea:focus::-webkit-input-placeholder { color:transparent; }
textarea:focus:-moz-placeholder { color:transparent; }
textarea:focus::-moz-placeholder { color:transparent; }
```

```
textarea:focus:-ms-input-placeholder { color:transparent; }

input::-webkit-input-placeholder { color: #999999;}
input:-moz-placeholder { color: #999999;}
input::-moz-placeholder { color: #999999;}
input:-ms-input-placeholder { color: #999999;}

textarea::-webkit-input-placeholder { color: #999999;}
textarea:-moz-placeholder { color: #999999;}
textarea::-moz-placeholder { color: #999999;}
textarea:-ms-input-placeholder { color: #999999;}


label {
  display: block;
  margin: 0;
}

button {
  outline: none !important;
  border: none;
  background: transparent;
}

button:hover {
  cursor: pointer;
}

iframe {
  border: none !important;
}


.txt1 {
  font-size: 13px;
  line-height: 1.4;
  color: #555555;
}

.txt2 {
  font-size: 13px;
  line-height: 1.4;
  color: #999999;
}
```

```
.size1 {
  width: 355px;
  max-width: 100%;
}

.size2 {
  width: calc(100% - 43px);
}

.bg1 {background: #3b5998;}
.bg2 {background: #1da1f2;}
.bg3 {background: #cd201f;}


.limiter {
  width: 100%;
  margin: 0 auto;
}

.container-login100 {
  width: 100%;
  min-height: 100vh;
  display: -webkit-box;
  display: -webkit-flex;
  display: -moz-box;
  display: -ms-flexbox;
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  align-items: center;
  background: #f2f2f2;
}


.wrap-login100 {
  width: 100%;
  background: #fff;
  overflow: hidden;
  display: -webkit-box;
  display: -webkit-flex;
  display: -moz-box;
  display: -ms-flexbox;
  display: flex;
  flex-wrap: wrap;
  align-items: stretch;
  flex-direction: row-reverse;
```

```
}

.login100-more {
  width: calc(100% - 560px);
  background-repeat: no-repeat;
  background-size: cover;
  background-position: center;
  position: relative;
  z-index: 1;
}

.login100-more::before {
  content: "";
  display: block;
  position: absolute;
  z-index: -1;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
  background: rgba(0,0,0,0.1);
}




.login100-form {
  width: 560px;
  min-height: 100vh;
  display: block;
  background-color: #f7f7f7;
  padding: 173px 55px 55px 55px;
}

.login100-form-title {
  width: 100%;
  display: block;
  font-size: 30px;
  color: #333333;
  line-height: 1.2;
  text-align: center;
}
```

```
.wrap-input100 {
  display: -webkit-box;
  display: -webkit-flex;
  display: -moz-box;
  display: -ms-flexbox;
  display: flex;
  flex-wrap: wrap;
  align-items: flex-end;
  width: 100%;
  height: 80px;
  position: relative;
  border: 1px solid #e6e6e6;
  border-radius: 10px;
  margin-bottom: 10px;
  margin-top: 30px;
}

.label-input100 {
  font-family: Montserrat-Regular;
  font-size: 18px;
  color: #999999;
  line-height: 1.2;

  display: block;
  position: absolute;
  pointer-events: none;
  width: 100%;
  padding-left: 24px;
  left: 0;
  top: 30px;

  -webkit-transition: all 0.4s;
  -o-transition: all 0.4s;
  -moz-transition: all 0.4s;
  transition: all 0.4s;
}

.input100 {
  display: block;
  width: 100%;
  background: transparent;
  font-size: 18px;
  color: #555555;
  line-height: 1.2;
  padding: 0 26px;
```

```
}

input.input100 {
  height: 100%;
  -webkit-transition: all 0.4s;
  -o-transition: all 0.4s;
  -moz-transition: all 0.4s;
  transition: all 0.4s;
}

.focus-input100 {
  position: absolute;
  display: block;
  width: calc(100% + 2px);
  height: calc(100% + 2px);
  top: -1px;
  left: -1px;
  pointer-events: none;
  border: 1px solid #6675df;
  border-radius: 10px;

  visibility: hidden;
  opacity: 0;

  -webkit-transition: all 0.4s;
  -o-transition: all 0.4s;
  -moz-transition: all 0.4s;
  transition: all 0.4s;

  -webkit-transform: scaleX(1.1) scaleY(1.3);
  -moz-transform: scaleX(1.1) scaleY(1.3);
  -ms-transform: scaleX(1.1) scaleY(1.3);
  -o-transform: scaleX(1.1) scaleY(1.3);
  transform: scaleX(1.1) scaleY(1.3);
}

.input100:focus + .focus-input100 {
  visibility: visible;
  opacity: 1;

  -webkit-transform: scale(1);
  -moz-transform: scale(1);
  -ms-transform: scale(1);
  -o-transform: scale(1);
  transform: scale(1);
```

```
}

.eff-focus-selection {
  visibility: visible;
  opacity: 1;

  -webkit-transform: scale(1);
  -moz-transform: scale(1);
  -ms-transform: scale(1);
  -o-transform: scale(1);
  transform: scale(1);
}

.input100:focus {
  height: 48px;
}

.input100:focus + .focus-input100 + .label-input100 {
  top: 14px;
  font-size: 13px;
}

.has-val {
  height: 48px !important;
}

.has-val + .focus-input100 + .label-input100 {
  top: 14px;
  font-size: 13px;
}




.container-login100-form-btn {
  width: 100%;
  display: -webkit-box;
  display: -webkit-flex;
  display: -moz-box;
  display: -ms-flexbox;
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
}
```

```
.login100-form-btn {
  display: -webkit-box;
  display: -webkit-flex;
  display: -moz-box;
  display: -ms-flexbox;
  display: flex;
  justify-content: center;
  align-items: center;
  padding: 0 20px;
  width: 100%;
  height: 50px;
  border-radius: 10px;
  background: #6675df;

  font-size: 12px;
  color: #fff;
  line-height: 1.2;
  text-transform: uppercase;
  letter-spacing: 1px;

  -webkit-transition: all 0.4s;
  -o-transition: all 0.4s;
  -moz-transition: all 0.4s;
  transition: all 0.4s;
}

.login100-form-btn:hover {
  background: #333333;
}



@media (max-width: 992px) {
  .login100-form {
    width: 50%;
    padding-left: 30px;
    padding-right: 30px;
  }

  .login100-more {
    width: 50%;
  }
}

@media (max-width: 768px) {
```

```
  .login100-form {
    width: 100%;
  }

  .login100-more {
    display: none;
  }
}

@media (max-width: 576px) {
  .login100-form {
    padding-left: 15px;
    padding-right: 15px;
    padding-top: 70px;
  }
}



.validate-input {
  position: relative;
}

.alert-validate::before {
  content: attr(data-validate);
  position: absolute;
  z-index: 100;
  max-width: 70%;
  background-color: #fff;
  border: 1px solid #c80000;
  border-radius: 2px;
  padding: 4px 25px 4px 10px;
  top: 50%;
  -webkit-transform: translateY(-50%);
  -moz-transform: translateY(-50%);
  -ms-transform: translateY(-50%);
  -o-transform: translateY(-50%);
  transform: translateY(-50%);
  right: 12px;
  pointer-events: none;

  font-family: Poppins-Regular;
  color: #c80000;
  font-size: 13px;
  line-height: 1.4;
```

```
  text-align: left;

  visibility: hidden;
  opacity: 0;

  -webkit-transition: opacity 0.4s;
  -o-transition: opacity 0.4s;
  -moz-transition: opacity 0.4s;
  transition: opacity 0.4s;
}

.alert-validate::after {
  content: "\f12a";
  font-family: FontAwesome;
  display: block;
  position: absolute;
  z-index: 110;
  color: #c80000;
  font-size: 16px;
  top: 50%;
  -webkit-transform: translateY(-50%);
  -moz-transform: translateY(-50%);
  -ms-transform: translateY(-50%);
  -o-transform: translateY(-50%);
  transform: translateY(-50%);
  right: 18px;
}

.alert-validate:hover:before {
  visibility: visible;
  opacity: 1;
}

#submitBtn{
  margin-top: 30px;
}


@media (max-width: 992px) {
  .alert-validate::before {
    visibility: visible;
    opacity: 1;
  }
}
```

```
.login100-form-social-item {
  width: 36px;
  height: 36px;
  font-size: 18px;
  color: #fff;
  border-radius: 50%;
}

.login100-form-social-item:hover {
  background: #333333;
  color: #fff;
}


</style>


</head>
<body style="background-color: #666666;">

  <div class="limiter">
    <div class="container-login100">
      <div class="wrap-login100">
        <form class="login100-form validate-
form" id="loginForm" onsubmit="return false">
          <span class="login100-form-title p-b-43 mb-8">
            Login to Smart Garden
          </span>


          <div class="mt-8 wrap-input100 validate-input">
            <input class="input100" type="text" id="userName"  required="required">
            <span class="focus-input100"></span>
            <span class="label-input100">Username</span>
          </div>


          <div class="wrap-input100 validate-input">
            <input class="input100" type="password" id="password" required="required
">
            <span class="focus-input100"></span>
            <span class="label-input100">Password</span>
          </div>
```

```
        <div class="form-group" id="submitBtn">
            <button type="submit" class="btn btn-primary btn-block login100-form-
btn">Log in</button>
        </div>


        </form>

        <div class="login100-more" style="background-image: url('../static/img/bg-
01.jpg');">
        </div>
      </div>
    </div>
  </div>




</body>

<script type="text/javascript">
    $('.input100').each(function(){
        $(this).on('blur', function(){
            if($(this).val().trim() != "") {
                $(this).addClass('has-val');
            }
            else {
                $(this).removeClass('has-val');
            }
        })
    })

$('#loginForm').on('submit',function (e) {
  console.log("call function");
  let user = $("#userName").val();
  let pwd = $("#password").val();
  console.log(user);
  console.log(pwd);
  jQuery.ajax({
                url: '/api/verify/'+user+'/'+pwd,
                type: 'POST',
                success: function(ndata){
                    console.log(ndata);
```

```
          if(ndata == "success"){
             window.location.replace("index.html");
          }
          else if(ndata == "fail"){
             window.alert("Please provide valid credentials!");
             location.reload();
          }
          }
            });//end ajax

});

</script>
</html>
```

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>Smart Garden</title>

  <!-- Custom fonts for this template-->
  <link href="static/vendor/fontawesome-
free/css/all.min.css" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,4
00i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">

  <!-- Custom styles for this template-->
  <link href="static/css/sb-admin-2.min.css" rel="stylesheet">
 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></scr
ipt>



</head>
```

```
<body id="page-top">

  <!-- Page Wrapper -->
  <div id="wrapper">

    <!-- Sidebar -->
    <ul class="navbar-nav bg-gradient-primary sidebar sidebar-
dark accordion" id="accordionSidebar">

      <!-- Sidebar - Brand -->
      <a class="sidebar-brand d-flex align-items-center justify-content-
center" href="index.html">
        <div class="sidebar-brand-icon rotate-n-5 pl-3">
    <i class="fas fa-seedling"></i>

        </div>
        <div class="sidebar-brand-text mx-3">Smart Garden<sup>1</sup></div>
      </a>

      <!-- Divider -->
      <hr class="sidebar-divider my-0">

      <!-- Nav Item - Dashboard -->
      <li class="nav-item active">
        <a class="nav-link" href="index.html">
          <i class="fas fa-fw fa-tachometer-alt"></i>
          <span>Current Data</span></a>
      </li>


      <!-- Divider -->
      <hr class="sidebar-divider">

      <!-- Heading -->
      <div class="sidebar-heading">
        Addons
      </div>


      <!-- Nav Item - Charts -->
      <li class="nav-item">
        <a class="nav-link" href="charts.html">
          <i class="fas fa-fw fa-chart-area"></i>
          <span>Charts</span></a>
      </li>
```

```
      <!-- Nav Item - Tables -->
      <li class="nav-item">
        <a class="nav-link" href="tables.html">
          <i class="fas fa-fw fa-table"></i>
          <span>Tables</span></a>
      </li>

          <!-- Nav Item - Gallery -->
      <li class="nav-item">
        <a class="nav-link" href="gallery.html">
      <i class="far fa-image"></i>
          <span>Gallery</span></a>
      </li>

      <!-- Divider -->
      <hr class="sidebar-divider d-none d-md-block">

      <!-- Sidebar Toggler (Sidebar) -->
      <div class="text-center d-none d-md-inline">
        <button class="rounded-circle border-0" id="sidebarToggle"></button>
      </div>

    </ul>
    <!-- End of Sidebar -->

    <!-- Content Wrapper -->
    <div id="content-wrapper" class="d-flex flex-column">

      <!-- Main Content -->
      <div id="content">

        <!-- Topbar -->
        <nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-
top shadow">

          <!-- Sidebar Toggle (Topbar) -->
          <button id="sidebarToggleTop" class="btn btn-link d-md-none rounded-
circle mr-3">
            <i class="fa fa-bars"></i>
          </button>


          <!-- Topbar Navbar -->
          <ul class="navbar-nav ml-auto">
```

```
            <!-- Nav Item - User Information -->
            <li class="nav-item dropdown no-arrow">
              <a class="nav-link dropdown-toggle" id="logout" role="button" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false"><i class="fas fa-sign-
out-alt fa-sm fa-fw mr-2 text-gray-400"></i>
                  Logout
              </a>
            </li>

          </ul>

        </nav>
        <!-- End of Topbar -->


        <!-- Begin Page Content -->
        <div class="container-fluid">

          <!-- Page Heading -->
          <div class="d-sm-flex align-items-center justify-content-between mb-4">
            <h1 class="h3 mb-0 text-gray-800">Current Data</h1>
          </div>

          <!-- Content Row -->
          <div class="row">

            <!-- Current Temperature -->
            <div class="col-xl-3 col-md-6 mb-4">
              <div class="card border-left-primary shadow h-100 py-2">
                <div class="card-body">
                  <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
                      <div class="text-xs font-weight-bold text-primary text-
uppercase mb-1">Temperature</div>
                      <div class="h5 mb-0 font-weight-bold text-gray-
800" id="currentTemp"></div>
                    </div>
                    <div class="col-auto">
                      <i class="fas fa-thermometer-three-quarters fa-2x text-gray-
300"></i>
                    </div>
                  </div>
                </div>
              </div>
            </div>
```

```
            <!-- Current Humidity -->
            <div class="col-xl-3 col-md-6 mb-4">
              <div class="card border-left-success shadow h-100 py-2">
                <div class="card-body">
                  <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
                      <div class="text-xs font-weight-bold text-success text-
uppercase mb-1">Humidity</div>
                      <div class="h5 mb-0 font-weight-bold text-gray-
800" id="currentHum"></div>
                    </div>
                    <div class="col-auto">
                      <i class="fas fa-tint fa-2x text-gray-300"></i>
                    </div>
                  </div>
                </div>
              </div>
            </div>

            <!-- Current Soil Moisture -->
            <div class="col-xl-3 col-md-6 mb-4">
              <div class="card border-left-info shadow h-100 py-2">
                <div class="card-body">
                  <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
                      <div class="text-xs font-weight-bold text-success text-
uppercase mb-1">Soil Moisture</div>
                      <div class="h5 mb-0 font-weight-bold text-gray-
800" id="currentSoil"></div>
                    </div>
                    <div class="col-auto">
                      <i class="fas fa-seedling fa-2x text-gray-300"></i>
                    </div>
                  </div>
                </div>
              </div>
            </div>

            <!-- Current Light -->
            <div class="col-xl-3 col-md-6 mb-4">
              <div class="card border-left-warning shadow h-100 py-2">
                <div class="card-body">
                  <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
```

```
                    <div class="text-xs font-weight-bold text-warning text-
uppercase mb-1">Light Level</div>
                    <div class="h5 mb-0 font-weight-bold text-gray-
800" id="currentLight"></div>
                  </div>
                  <div class="col-auto">
                    <i class="fas fa-lightbulb fa-2x text-gray-300"></i>
                  </div>
                </div>
              </div>
            </div>
          </div>


          <!-- Page Heading -->
          <div class="d-sm-flex align-items-center justify-content-between mb-2 mt-
4">
            <h1 class="h3 mb-2 text-gray-800">Watering Options</h1>
          </div>


<div class="row">
            <div class="col-xl-3 col-md-6 mb-4">
              <div class="card border-left-danger shadow h-100 py-2">
        <div class="card-body">
                <div class="row no-gutters align-items-center">
                  <div class="col mr-2">
                    <div class="text-xs font-weight-bold text-warning text-
uppercase mb-1">Automated Watering</div>
                    <div class="h5 mb-0 font-weight-bold text-gray-
800" id="currentLight"></div>
                  </div>
                  <div class="col-auto">
                    <label class="toggleBtn">
                <input class="switch-
input" id="autoSwitch" type="checkbox" onclick="auto()" />
                <span class="switch-label" data-on="on" data-off="off"></span>
                <span class="switch-handle"></span>
              </label>
                  </div>
                </div>
              </div>
        </div>
        </div>
```

```
            <div class="col-xl-3 col-md-6 mb-4" id="manual">
              <div class="card border-left-danger shadow h-100 py-2">
                <div class="card-body">
                  <div class="row no-gutters align-items-center">
                    <div class="col mr-2">
                      <div class="text-xs font-weight-bold text-warning text-
uppercase mb-1">Manual On/Off</div>
                      <div class="h5 mb-0 font-weight-bold text-gray-
800" id="currentLight"></div>
                    </div>
                    <div class="col-auto">
                      <label class="toggleBtn">
                <input class="switch-input switch2-
input" id="manualSwitch" type="checkbox" onclick="manual()" />
                <span class="switch-label switch2-label" data-on="on" data-
off="off"></span>
                <span class="switch-handle switch2-handle"></span>
              </label>
                    </div>
                  </div>
                </div>
        </div>
        </div>

        </div>
    <div class="col-lg-6 mt-2">


              <!-- Approach -->
              <div class="card shadow mb-4">
                <div class="card-header py-3">
                  <h6 class="m-0 font-weight-bold text-primary">Standard</h6>
                </div>
                <div class="card-body">
                  When the soil moisture level goes above 700 (for our soil moisture
 sensor, the higher it is the drier the soil), the red LED will light up as a warnin
g to show that the plant needs water. In the automated system, the water pump will s
tart to run and pump water into the soil automatically. The user can also switch to
manual system that he can on or off the pump manually.
                </div>
              </div>

            </div>
```

```
      </div>
      <!-- End of Main Content -->

      <!-- Footer -->
      <footer class="sticky-footer bg-white">
        <div class="container my-auto">
          <div class="copyright text-center my-auto">
            <span>Copyright &copy; Your Website 2019</span>
          </div>
        </div>
      </footer>
      <!-- End of Footer -->

    </div>
    <!-- End of Content Wrapper -->

  </div>
  <!-- End of Page Wrapper -->

  <!-- Scroll to Top Button-->
  <a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
  </a>

  <!-- Logout Modal-->
  <div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
          <button class="close" type="button" data-dismiss="modal" aria-
label="Close">
            <span aria-hidden="true">×</span>
          </button>
        </div>
        <div class="modal-
body">Select "Logout" below if you are ready to end your current session.</div>
        <div class="modal-footer">
          <button class="btn btn-secondary" type="button" data-
dismiss="modal">Cancel</button>
          <a class="btn btn-primary" href="login.html">Logout</a>
        </div>
      </div>
```

```
  </div>
</div>

<!-- Bootstrap core JavaScript-->
<script src="static/vendor/jquery/jquery.min.js"></script>
<script src="static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->
<script src="static/vendor/jquery-easing/jquery.easing.min.js"></script>

<!-- Custom scripts for all pages-->
<script src="static/js/sb-admin-2.min.js"></script>

<!-- Page level plugins -->
<script src="static/vendor/chart.js/Chart.min.js"></script>

<!-- Page level custom scripts -->
<script src="static/js/demo/chart-area-demo.js"></script>
<script src="static/js/demo/chart-pie-demo.js"></script>
<script type="text/javascript">



      function getCurrentData(){
          jQuery.ajax({
              url: '/api/getData',
              type: 'POST',
              success: function(ndata){
                  console.log(ndata);
        current = ndata[0];
        $("#currentTemp").text(current.temperature);
        $("#currentHum").text(current.humidity);
        $("#currentSoil").text(current.soil);
        $("#currentLight").text(current.light);
              }//end success
          });//end ajax
        } //end getNewData


      $(document).ready(function(){
  getStatus();
  getCurrentData();
          setInterval(function () {
              getCurrentData()
          }, 5000);
      });
```

```
      function getStatus(){
           jQuery.ajax({
               url: '/api/status',
               type: 'POST',
               success: function(ndata){
                   console.log(ndata);
          status = ndata[0].status;
          if (status=="A"){
          autoSwitch.checked = true;
             console.log("auto");
             $("#manual").hide();
          }
          else{
          autoSwitch.checked = false;
          $("#manual").show();
          if(status=="O"){
          manualSwitch.checked = true;
          }
          else{
          manualSwitch.checked = false;
          }
          }
                }//end success
            });//end ajax
          } //end getNewData




const autoSwitch = document.getElementById("autoSwitch");
const manualSwitch = document.getElementById("manualSwitch");


function auto() {
  let autoStatus;
  if (autoSwitch.checked) {
    autoStatus = "A";
  } else {
    autoStatus = "M";
  }
  console.log(autoStatus);

  $.ajax({
```

```
    url: '/api/changeStatus/' + autoStatus,
  type: 'POST',
  success: function(ndata){
  console.log(ndata);
    if (autoSwitch.checked) {
    $("#manual").hide();
    manualSwitch.disabled = true;
    manualSwitch.checked = false;
  } else {
  $("#manual").show();
    manualSwitch.disabled = false;
  }
  console.log("auto-success");
                }//end success
            });//end ajax
}



function manual() {
  let manualStatus;
  if (manualSwitch.checked) {
    manualStatus = "O";
  } else {
    manualStatus = "F";
  }
  console.log(manualStatus);
$.ajax({
    url: '/api/changeStatus/' + manualStatus,
  type: 'POST',
  success: function(ndata){
  console.log(ndata);
}//end success
            });//end ajax
}



$('#logout').on('click',function () {
   window.location.replace("logout");

});
```

```
</script>
</body>

</html>
```

charts.html

```
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>Smart Garden</title>

  <!-- Custom fonts for this template-->
  <link href="static/vendor/fontawesome-
free/css/all.min.css" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,4
00i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">

  <!-- Custom styles for this template-->
  <link href="static/css/sb-admin-2.min.css" rel="stylesheet">
 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></scr
ipt>



</head>

<body id="page-top">

  <!-- Page Wrapper -->
  <div id="wrapper">

    <!-- Sidebar -->
    <ul class="navbar-nav bg-gradient-primary sidebar sidebar-
dark accordion" id="accordionSidebar">
```

```
      <!-- Sidebar - Brand -->
      <a class="sidebar-brand d-flex align-items-center justify-content-
center" href="index.html">
        <div class="sidebar-brand-icon rotate-n-5 pl-3">
    <i class="fas fa-seedling"></i>

        </div>
        <div class="sidebar-brand-text mx-3">Smart Garden<sup>1</sup></div>
      </a>

      <!-- Divider -->
      <hr class="sidebar-divider my-0">

      <!-- Nav Item - Dashboard -->
      <li class="nav-item">
        <a class="nav-link" href="index.html">
          <i class="fas fa-fw fa-tachometer-alt"></i>
          <span>Current Data</span></a>
      </li>


      <!-- Divider -->
      <hr class="sidebar-divider">

      <!-- Heading -->
      <div class="sidebar-heading">
        Addons
      </div>


      <!-- Nav Item - Charts -->
      <li class="nav-item active">
        <a class="nav-link" href="charts.html">
          <i class="fas fa-fw fa-chart-area"></i>
          <span>Charts</span></a>
      </li>

      <!-- Nav Item - Tables -->
      <li class="nav-item">
        <a class="nav-link" href="tables.html">
          <i class="fas fa-fw fa-table"></i>
          <span>Tables</span></a>
      </li>

      <!-- Nav Item - Gallery -->
```

```
      <li class="nav-item">
        <a class="nav-link" href="gallery.html">
      <i class="far fa-image"></i>
          <span>Gallery</span></a>
      </li>

      <!-- Divider -->
      <hr class="sidebar-divider d-none d-md-block">

      <!-- Sidebar Toggler (Sidebar) -->
      <div class="text-center d-none d-md-inline">
        <button class="rounded-circle border-0" id="sidebarToggle"></button>
      </div>

    </ul>
    <!-- End of Sidebar -->

    <!-- Content Wrapper -->
    <div id="content-wrapper" class="d-flex flex-column">

      <!-- Main Content -->
      <div id="content">

        <!-- Topbar -->
        <nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-
top shadow">

          <!-- Sidebar Toggle (Topbar) -->
          <button id="sidebarToggleTop" class="btn btn-link d-md-none rounded-
circle mr-3">
            <i class="fa fa-bars"></i>
          </button>

                    <!-- Topbar Navbar -->
          <ul class="navbar-nav ml-auto">
            <!-- Nav Item - User Information -->
            <li class="nav-item dropdown no-arrow">
              <a class="nav-link dropdown-toggle" id="logout" role="button" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false"><i class="fas fa-sign-
out-alt fa-sm fa-fw mr-2 text-gray-400"></i>
                  Logout
              </a>
            </li>

          </ul>
```

```
        </nav>
        <!-- End of Topbar -->


        <!-- Begin Page Content -->
        <div class="container-fluid">

          <!-- Page Heading -->
          <h1 class="h3 mb-2 text-gray-800">Charts</h1>
          <p class="mb-4"></p>

          <!-- Content Row -->
          <div class="row">
            <div class="col-xl-10 col-lg-10">
              <!-- Area Chart -->
              <div class="card shadow mb-4">
                <div class="card-header py-3">
                  <h6 class="m-0 font-weight-bold text-
primary">Temperature Chart</h6>
                </div>
                <div class="card-body">
                  <div class="chart-area" id="temChart">

                  </div>
                  <hr>
          <p>It monitors the temperature condition of the plant..</p>
                </div>
              </div>
            </div>

      <div class="col-xl-10 col-lg-10">
              <!-- Area Chart -->
              <div class="card shadow mb-4">
                <div class="card-header py-3">
                  <h6 class="m-0 font-weight-bold text-success">Humidity Chart</h6>
                </div>
                <div class="card-body">
                  <div class="chart-area" id="humChart">

                  </div>
                  <hr>
          <p>It monitors the humidity of the environment where the plant is located.
</p>
                </div>
```

```
            </div>
          </div>

      <div class="col-xl-10 col-lg-10">
            <!-- Area Chart -->
            <div class="card shadow mb-4">
              <div class="card-header py-3">
                <h6 class="m-0 font-weight-bold text-
secondary">Soil Moisture Chart</h6>
              </div>
              <div class="card-body">
                <div class="chart-area" id="soilChart">

                </div>
                <hr>
          <p>It monitors the soil moisture of the plant. When the soil moisture leve
l goes above 500, the user will receive an alert message.</p>
              </div>
            </div>
          </div>

      <div class="col-xl-10 col-lg-10">
            <!-- Area Chart -->
            <div class="card shadow mb-4">
              <div class="card-header py-3">
                <h6 class="m-0 font-weight-bold text-
warning">Light Level Chart</h6>
              </div>
              <div class="card-body">
                <div class="chart-area" id="lightChart">

                </div>
                <hr>
          <p>It monitors the light level of the environment where the plant is locat
ed.</p>
              </div>
            </div>
          </div>
        </div>
        <!-- /.container-fluid -->
      </div>
      <!-- End of Main Content -->

      <!-- Footer -->
      <footer class="sticky-footer bg-white">
```

```
        <div class="container my-auto">
          <div class="copyright text-center my-auto">
            <span>Copyright &copy; Your Website 2019</span>
          </div>
        </div>
      </footer>
      <!-- End of Footer -->

    </div>
    <!-- End of Content Wrapper -->

  </div>
  <!-- End of Page Wrapper -->

  <!-- Scroll to Top Button-->
  <a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
  </a>

  <!-- Logout Modal-->
  <div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
          <button class="close" type="button" data-dismiss="modal" aria-
label="Close">
            <span aria-hidden="true">×</span>
          </button>
        </div>
        <div class="modal-
body">Select "Logout" below if you are ready to end your current session.</div>
        <div class="modal-footer">
          <button class="btn btn-secondary" type="button" data-
dismiss="modal">Cancel</button>
          <a class="btn btn-primary" href="login.html">Logout</a>
        </div>
      </div>
    </div>
  </div>


  <!-- Bootstrap core JavaScript-->
  <script src="static/vendor/jquery/jquery.min.js"></script>
```

```
<script src="static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->
<script src="static/vendor/jquery-easing/jquery.easing.min.js"></script>

<!-- Custom scripts for all pages-->
<script src="static/js/sb-admin-2.min.js"></script>

<!-- Page level plugins -->
<script src="static/vendor/chart.js/Chart.min.js"></script>

<!-- Page level custom scripts -->
<script src="static/js/demo/chart-area-demo.js"></script>
<script src="static/js/demo/chart-pie-demo.js"></script>
<script src="https://www.gstatic.com/charts/loader.js"></script>
  <script type="text/javascript">

      $(document).ready(function(){
  tempChart();
  humChart();
  soilChart();
  lightChart();
          setInterval(function () {
                  tempChart();
  humChart();
  soilChart();
  lightChart();
          }, 5000);

      });

 google.charts.load("current", {
      packages : [ "corechart","table" ]
  });




 function tempChart(){
          jQuery.ajax({
              url: '/api/getChart',
              method: 'POST',
              success: function(ndata){
            console.log(ndata);
```

```
                chartdata = ndata;
                let tempGraphdata = createTempDataTable(chartdata);
                drawTempLineChart(tempGraphdata);
            }//end success
        });//end ajax
      } //end getNewData



    function createTempDataTable(newdata){
        let tempGraphdata = new google.visualization.DataTable();
        tempGraphdata.addColumn('string', 'Time');
        tempGraphdata.addColumn('number', 'Temperature');
        for (i in newdata) {
            datetime = newdata[i].datetimeid;
            jsdatetime = new Date(Date.parse(datetime));
            jstime = jsdatetime.toLocaleTimeString();
            temp = newdata[i].temperature;
            tempGraphdata.addRows([[jstime,temp]]);
        }//end for
        return tempGraphdata
    }


    function drawTempLineChart(graphdata) {
        chart = new google.visualization.LineChart(
        document.getElementById('temChart'));
        chart.draw(graphdata, {
      title: 'Historical Temperature Record',
titleTextStyle :{
      fontSize : 18,
color: '#428bca'
    },
hAxis: {
    title: 'Time',
textStyle :{
      fontSize : 10
    }},
    vAxis: {
baseline: 0,
title: 'Temperature (°C)',
textStyle :{
      fontSize : 10},
scaleType: 'log',
    },
```

```
        pointSize: 8,
    legend: 'none',
        colors: ['#428bca']});
          return
      } //end drawChart




function humChart(){
        jQuery.ajax({
            url: '/api/getChart',
            method: 'POST',
            success: function(ndata){
                chartdata = ndata;
                let humGraphdata = createhumDataTable(chartdata);
                drawHumLineChart(humGraphdata);
            }//end success
        });//end ajax
      } //end getNewData



    function createhumDataTable(newdata){
        let tempGraphdata = new google.visualization.DataTable();
        tempGraphdata.addColumn('string', 'Time');
        tempGraphdata.addColumn('number', 'Humidity');
        for (i in newdata) {
            datetime = newdata[i].datetimeid;
            jsdatetime = new Date(Date.parse(datetime));
            jstime = jsdatetime.toLocaleTimeString();
            hum = newdata[i].humidity;
            tempGraphdata.addRows([[jstime,hum]]);
        }//end for
        return tempGraphdata
    }


    function drawHumLineChart(graphdata) {
        chart = new google.visualization.LineChart(
        document.getElementById('humChart'));
        chart.draw(graphdata, {
      title: 'Historical Humidity Record',
  titleTextStyle :{
```

```
            fontSize : 18,
        color: '#5cb85c'
            },
        hAxis: {
            title: 'Time',
        textStyle :{
            fontSize : 10
            }},
            vAxis: {
        baseline: 0,
        title: 'Humidity',
        textStyle :{
            fontSize : 10},
        scaleType: 'log',
            },
            pointSize: 8,
        legend: 'none',
            colors: ['#5cb85c']});
            return
          } //end drawChart




function soilChart(){
            jQuery.ajax({
                url: '/api/getChart',
                method: 'POST',
                success: function(ndata){
                    chartdata = ndata;
                    let soilGraphdata = createSoilDataTable(chartdata);
                    drawSoilLineChart(soilGraphdata);
                }//end success
            });//end ajax
          } //end getNewData



        function createSoilDataTable(newdata){
            let tempGraphdata = new google.visualization.DataTable();
            tempGraphdata.addColumn('string', 'Time');
            tempGraphdata.addColumn('number', 'Soil');
            for (i in newdata) {
                datetime = newdata[i].datetimeid;
```

```
            jsdatetime = new Date(Date.parse(datetime));
            jstime = jsdatetime.toLocaleTimeString();
            soil = newdata[i].soil;
            tempGraphdata.addRows([[jstime,soil]]);
        }//end for
        return tempGraphdata
    }


    function drawSoilLineChart(graphdata) {
        chart = new google.visualization.LineChart(
        document.getElementById('soilChart'));
        chart.draw(graphdata, {
      title: 'Historical Soil Moisture Record',
  titleTextStyle :{
        fontSize : 18,
  color: '#292b2c'
      },
  hAxis: {
      title: 'Time',
  textStyle :{
        fontSize : 10
      }},
      vAxis: {
  baseline: 0,
  title: 'Soil Moisture',
  textStyle :{
        fontSize : 10},
  scaleType: 'log',
      },
      pointSize: 8,
  legend: 'none',
      colors: ['#292b2c']});
        return
    } //end drawChart




function lightChart(){
        jQuery.ajax({
            url: '/api/getChart',
            method: 'POST',
```

```
            success: function(ndata){
                chartdata = ndata;
                let lightGraphdata = createLightDataTable(chartdata);
                drawLightLineChart(lightGraphdata);
            }//end success
        });//end ajax
      } //end getNewData



    function createLightDataTable(newdata){
        let tempGraphdata = new google.visualization.DataTable();
        tempGraphdata.addColumn('string', 'Time');
        tempGraphdata.addColumn('number', 'Light');
        for (i in newdata) {
            datetime = newdata[i].datetimeid;
            jsdatetime = new Date(Date.parse(datetime));
            jstime = jsdatetime.toLocaleTimeString();
            light = newdata[i].light;
            tempGraphdata.addRows([[jstime,light]]);
        }//end for
        return tempGraphdata
    }


    function drawLightLineChart(graphdata) {
        chart = new google.visualization.LineChart(
        document.getElementById('lightChart'));
        chart.draw(graphdata, {
      title: 'Historical Light Level Record',
titleTextStyle :{
        fontSize : 18,
color: '#f0ad4e'
    },
hAxis: {
    title: 'Time',
textStyle :{
        fontSize : 10
    }},
    vAxis: {
baseline: 0,
title: 'Light Level',
textStyle :{
        fontSize : 10},
scaleType: 'log',
```

```
          },
          pointSize: 8,
      legend: 'none',
          colors: ['#f0ad4e']});
            return
        } //end drawChart




    $('#logout').on('click',function () {
   window.location.replace("logout");

});

  </script>
</body>

</html>
```

tables.html

```
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>Smart Garden</title>

  <!-- Custom fonts for this template-->
  <link href="static/vendor/fontawesome-
free/css/all.min.css" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,4
00i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">

  <!-- Custom styles for this template-->
  <link href="static/css/sb-admin-2.min.css" rel="stylesheet">
```

```
 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></scr
ipt>



</head>

<body id="page-top">

  <!-- Page Wrapper -->
  <div id="wrapper">

    <!-- Sidebar -->
    <ul class="navbar-nav bg-gradient-primary sidebar sidebar-
dark accordion" id="accordionSidebar">

      <!-- Sidebar - Brand -->
      <a class="sidebar-brand d-flex align-items-center justify-content-
center" href="index.html">
        <div class="sidebar-brand-icon rotate-n-5 pl-3">
    <i class="fas fa-seedling"></i>

        </div>
        <div class="sidebar-brand-text mx-3">Smart Garden<sup>1</sup></div>
      </a>

      <!-- Divider -->
      <hr class="sidebar-divider my-0">

      <!-- Nav Item - Dashboard -->
      <li class="nav-item">
        <a class="nav-link" href="index.html">
          <i class="fas fa-fw fa-tachometer-alt"></i>
          <span>Current Data</span></a>
      </li>


      <!-- Divider -->
      <hr class="sidebar-divider">

      <!-- Heading -->
      <div class="sidebar-heading">
        Addons
      </div>
```

```
      <!-- Nav Item - Charts -->
      <li class="nav-item">
        <a class="nav-link" href="charts.html">
          <i class="fas fa-fw fa-chart-area"></i>
          <span>Charts</span></a>
      </li>

      <!-- Nav Item - Tables -->
      <li class="nav-item active">
        <a class="nav-link" href="tables.html">
          <i class="fas fa-fw fa-table"></i>
          <span>Tables</span></a>
      </li>

    <!-- Nav Item - Gallery -->
      <li class="nav-item">
        <a class="nav-link" href="gallery.html">
      <i class="far fa-image"></i>
          <span>Gallery</span></a>
      </li>

      <!-- Divider -->
      <hr class="sidebar-divider d-none d-md-block">

      <!-- Sidebar Toggler (Sidebar) -->
      <div class="text-center d-none d-md-inline">
        <button class="rounded-circle border-0" id="sidebarToggle"></button>
      </div>

    </ul>
    <!-- End of Sidebar -->

    <!-- Content Wrapper -->
    <div id="content-wrapper" class="d-flex flex-column">

      <!-- Main Content -->
      <div id="content">

        <!-- Topbar -->
        <nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-
top shadow">

          <!-- Sidebar Toggle (Topbar) -->
          <button id="sidebarToggleTop" class="btn btn-link d-md-none rounded-
circle mr-3">
```

```
            <i class="fa fa-bars"></i>
          </button>

                    <!-- Topbar Navbar -->
          <ul class="navbar-nav ml-auto">
            <!-- Nav Item - User Information -->
            <li class="nav-item dropdown no-arrow">
              <a class="nav-link dropdown-toggle" id="logout" role="button" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false"><i class="fas fa-sign-
out-alt fa-sm fa-fw mr-2 text-gray-400"></i>
                  Logout
              </a>
            </li>

          </ul>

        </nav>
        <!-- End of Topbar -->


        <!-- Begin Page Content -->
        <div class="container-fluid">

          <!-- Page Heading -->
          <h1 class="h3 mb-2 text-gray-800">Tables</h1>


          <!-- smartGarden DataTables  -->
          <div class="card shadow mb-4">
            <div class="card-header py-3">
              <h6 class="m-0 font-weight-bold text-
primary">Smart Garden Data Monitoring</h6>
            </div>
            <div class="card-body">
              <div class="table-responsive" id="smartGardenTable">

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


       <!-- statusTable DataTables -->
          <div class="card shadow mb-4">
            <div class="card-header py-3">
              <h6 class="m-0 font-weight-bold text-warning">Status Control</h6>
            </div>
```

```
            <div class="card-body">
              <div class="table-responsive" id="controlTable">

              </div>
            </div>
          </div>
        </div>
        <!-- /.container-fluid -->



      </div>
      <!-- End of Main Content -->

      <!-- Footer -->
      <footer class="sticky-footer bg-white">
        <div class="container my-auto">
          <div class="copyright text-center my-auto">
            <span>Copyright &copy; Your Website 2019</span>
          </div>
        </div>
      </footer>
      <!-- End of Footer -->

    </div>
    <!-- End of Content Wrapper -->

  </div>
  <!-- End of Page Wrapper -->

  <!-- Scroll to Top Button-->
  <a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
  </a>

  <!-- Logout Modal-->
  <div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
          <button class="close" type="button" data-dismiss="modal" aria-
label="Close">
            <span aria-hidden="true">×</span>
```

```
          </button>
        </div>
        <div class="modal-
body">Select "Logout" below if you are ready to end your current session.</div>
        <div class="modal-footer">
          <button class="btn btn-secondary" type="button" data-
dismiss="modal">Cancel</button>
          <a class="btn btn-primary" href="login.html">Logout</a>
        </div>
      </div>
    </div>
  </div>

  <!-- Bootstrap core JavaScript-->
  <script src="static/vendor/jquery/jquery.min.js"></script>
  <script src="static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

  <!-- Core plugin JavaScript-->
  <script src="static/vendor/jquery-easing/jquery.easing.min.js"></script>

  <!-- Custom scripts for all pages-->
  <script src="static/js/sb-admin-2.min.js"></script>

  <!-- Page level plugins -->
  <script src="static/vendor/chart.js/Chart.min.js"></script>

  <!-- Page level custom scripts -->
  <script src="static/js/demo/chart-area-demo.js"></script>
  <script src="static/js/demo/chart-pie-demo.js"></script>
   <script src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">

        $(document).ready(function(){
    loadTable();
    loadStatusTable();
            setInterval(function () {
                    loadTable();
            loadStatusTable();
            }, 5000);

        });

   google.charts.load("current", {
        packages : [ "corechart","table" ]
    });
```

```
   function loadTable(){
            jQuery.ajax({
                url: '/api/getChart',
                method: 'POST',
                success: function(ndata){
                    chartdata = ndata;
                    let graphdata = createDataTable(chartdata);
                    drawDataTable(graphdata);
                }//end success
            });//end ajax
           } //end getNewData



        function createDataTable(newdata){
            let graphdata = new google.visualization.DataTable();
            graphdata.addColumn('string', 'Time');
            graphdata.addColumn('number', 'Temperature');
       graphdata.addColumn('number', 'Humidity');
       graphdata.addColumn('number', 'Soil Moisture');
       graphdata.addColumn('number', 'Light Level');
            for (i in newdata) {
            console.log(newdata[i]);
                datetime = newdata[i].datetimeid;
                jsdatetime = new Date(Date.parse(datetime));
                jstime = jsdatetime.toLocaleTimeString();

                temp = newdata[i].temperature;
          hum = newdata[i].humidity;
          soil = newdata[i].soil;
          light = newdata[i].light;
                graphdata.addRows([[jstime,temp,hum,soil,light]]);
            }//end for
            return graphdata
        }


      function drawDataTable(graphdata){
            var table = new google.visualization.Table(document.getElementById('smar
tGardenTable'));
```

```
            table.draw(graphdata, {showRowNumber: true, width: '100%', height: '100%
'});

        }//end drawTable


       function loadStatusTable(){
            jQuery.ajax({
                url: '/api/getAllStatus',
                method: 'POST',
                success: function(ndata){
                    chartdata = ndata;
          console.log(chartdata);
                    let statusData = createStatusDataTable(chartdata);
                    drawStatusDataTable(statusData);
                }//end success
            });//end ajax
          } //end getNewData



        function createStatusDataTable(newdata){
            let statusGraphdata = new google.visualization.DataTable();
            statusGraphdata.addColumn('string', 'Time');
            statusGraphdata.addColumn('string', 'Status');
            for (i in newdata) {
                datetime = newdata[i].datetimeid;
                jsdatetime = new Date(Date.parse(datetime));
                jstime = jsdatetime.toLocaleTimeString();
                status = newdata[i].status;
        if(status=="O"){
           status="Manual-On"
        }
        else if(status=="F" || status=="M"){
           status="Manual-Off"
        }
        else if(status=="A"){
           status="Auto-Control"
        }
        console.log(status);
                statusGraphdata.addRows([[jstime,status]]);
            }//end for
            return statusGraphdata
        }
```

```
    function drawStatusDataTable(graphdata){
            var table = new google.visualization.Table(document.getElementById('cont
rolTable'));
            table.draw(graphdata, {showRowNumber: true, width: '70%', height: '100%'
});

        }//end drawTable



    $('#logout').on('click',function () {
   window.location.replace("logout");

});

  </script>




</body>

</html>
```

gallery.html

```
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>Smart Garden</title>

  <!-- Custom fonts for this template-->
  <link href="static/vendor/fontawesome-
free/css/all.min.css" rel="stylesheet" type="text/css">
```

```
  <link href="https://fonts.googleapis.com/css?family=Nunito:200,200i,300,300i,400,4
00i,600,600i,700,700i,800,800i,900,900i" rel="stylesheet">

  <!-- Custom styles for this template-->
  <link href="static/css/sb-admin-2.min.css" rel="stylesheet">
 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></scr
ipt>



</head>

<body id="page-top">

  <!-- Page Wrapper -->
  <div id="wrapper">

    <!-- Sidebar -->
    <ul class="navbar-nav bg-gradient-primary sidebar sidebar-
dark accordion" id="accordionSidebar">

      <!-- Sidebar - Brand -->
      <a class="sidebar-brand d-flex align-items-center justify-content-
center" href="index.html">
        <div class="sidebar-brand-icon rotate-n-5 pl-3">
    <i class="fas fa-seedling"></i>

        </div>
        <div class="sidebar-brand-text mx-3">Smart Garden<sup>1</sup></div>
      </a>

      <!-- Divider -->
      <hr class="sidebar-divider my-0">

      <!-- Nav Item - Dashboard -->
      <li class="nav-item">
        <a class="nav-link" href="index.html">
          <i class="fas fa-fw fa-tachometer-alt"></i>
          <span>Current Data</span></a>
      </li>


      <!-- Divider -->
      <hr class="sidebar-divider">
```

```
      <!-- Heading -->
      <div class="sidebar-heading">
        Addons
      </div>


      <!-- Nav Item - Charts -->
      <li class="nav-item">
        <a class="nav-link" href="charts.html">
          <i class="fas fa-fw fa-chart-area"></i>
          <span>Charts</span></a>
      </li>

    <!-- Nav Item - Tables -->
      <li class="nav-item">
        <a class="nav-link" href="tables.html">
          <i class="fas fa-fw fa-table"></i>
          <span>Tables</span></a>
      </li>

      <!-- Nav Item - Gallery -->
      <li class="nav-item active">
        <a class="nav-link" href="gallery.html">
      <i class="far fa-image"></i>
          <span>Gallery</span></a>
      </li>

      <!-- Divider -->
      <hr class="sidebar-divider d-none d-md-block">

      <!-- Sidebar Toggler (Sidebar) -->
      <div class="text-center d-none d-md-inline">
        <button class="rounded-circle border-0" id="sidebarToggle"></button>
      </div>

    </ul>
    <!-- End of Sidebar -->


    <!-- Content Wrapper -->
    <div id="content-wrapper" class="d-flex flex-column">

      <!-- Main Content -->
      <div id="content">

        <!-- Topbar -->
```

```
        <nav class="navbar navbar-expand navbar-light bg-white topbar mb-4 static-
top shadow">


                    <!-- Topbar Navbar -->
          <ul class="navbar-nav ml-auto">
            <!-- Nav Item - User Information -->
            <li class="nav-item dropdown no-arrow">
              <a class="nav-link dropdown-toggle" id="logout" role="button" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false"><i class="fas fa-sign-
out-alt fa-sm fa-fw mr-2 text-gray-400"></i>
                  Logout
              </a>
            </li>

          </ul>

        </nav>
        <!-- End of Topbar -->


  <div class="container">

    <div class="col-12">
  <h3 id="pagetitle" class="center"></h3>
  <button type="button" id="plantRad" class="btn btn-
primary disabled">Bugs on Plant</button>
    <button type="button" id="flowerRad" class="btn btn-
primary active">Bugs on Flower</button>

    <table class="table table-image table-responsive mt-3">
      <thead>
        <tr>
          <th scope="col">Index</th>
          <th scope="col">Image</th>
          <th scope="col">Detected By</th>
        <th scope="col">Detected At (Date)</th>
          <th scope="col">Detected At (Time)</th>
        </tr>
      </thead>
      <tbody id="tablebody">
      </tbody>
    </table>
    </div>
```

```
</div>



      </div>
      <!-- End of Main Content -->

      <!-- Footer -->
      <footer class="sticky-footer bg-white">
        <div class="container my-auto">
          <div class="copyright text-center my-auto">
            <span>Copyright &copy; Your Website 2019</span>
          </div>
        </div>
      </footer>
      <!-- End of Footer -->

    </div>
    <!-- End of Content Wrapper -->

  </div>
  <!-- End of Page Wrapper -->

  <!-- Scroll to Top Button-->
  <a class="scroll-to-top rounded" href="#page-top">
    <i class="fas fa-angle-up"></i>
  </a>

  <!-- Logout Modal-->
  <div class="modal fade" id="logoutModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>
          <button class="close" type="button" data-dismiss="modal" aria-
label="Close">
            <span aria-hidden="true">×</span>
          </button>
        </div>
        <div class="modal-
body">Select "Logout" below if you are ready to end your current session.</div>
        <div class="modal-footer">
          <button class="btn btn-secondary" type="button" data-
dismiss="modal">Cancel</button>
```

```
        <a class="btn btn-primary" href="login.html">Logout</a>
      </div>
    </div>
  </div>
</div>

<!-- Bootstrap core JavaScript-->
<script src="static/vendor/jquery/jquery.min.js"></script>
<script src="static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->
<script src="static/vendor/jquery-easing/jquery.easing.min.js"></script>

<!-- Custom scripts for all pages-->
<script src="static/js/sb-admin-2.min.js"></script>

<!-- Page level plugins -->
<script src="static/vendor/chart.js/Chart.min.js"></script>

<!-- Page level custom scripts -->
<script src="static/js/demo/chart-area-demo.js"></script>
<script src="static/js/demo/chart-pie-demo.js"></script>
 <script src="https://www.gstatic.com/charts/loader.js"></script>
  <script type="text/javascript">

      $(document).ready(function(){
  loadPlantPic();
  $('#plantRad').click(function() {
      loadPlantPic();
  });
$('#flowerRad').click(function() {
      loadFlowerPic();
  });

          setInterval(function () {
          }, 5000);

      });

  function loadPlantPic(){

  $('#tablebody').html('');
  $('#pagetitle').text('Bugs on Plant');
  $('#plantRad').prop('disabled', true);
    $("#flowerRad").removeAttr('disabled');
```

```
  $("#plantRad").removeClass("active");
      $("#flowerRad").removeClass("disabled");
  $("#plantRad").addClass("disabled");
      $("#flowerRad").addClass("active");

     $.ajax({
  url: '/api/getImg/plant',
type: 'POST',
success: function(ndata){
console.log(ndata);
  for (i in ndata) {
    console.log(ndata[i]);
  datetime = ndata[i].datetimeid;
      jsdatetime = new Date(Date.parse(datetime));
      jstime = jsdatetime.toLocaleTimeString();
  uploaddate = jsdatetime.toDateString()
     console.log("pic" + i + ndata[i].filename);

     let row = $('<tr></tr>');

     let indexCol = $('<td scope="row"></td>');
     indexCol.text("#"+(i*1+1));

     let imgCol = $('<td class="w-25"></td>');
     let imgEle = $('<img class="img-fluid img-thumbnail" alt="imgage">');
     imgEle.attr('id', 'img_src'+i);
     imgCol.append(imgEle);

     let detectByCol = $('<td></td>');
     detectByCol.attr('id', 'detectby'+i);

     let detectedDateCol = $('<td></td>');
     detectedDateCol.attr('id', 'date'+i);

         let detectedAtCol = $('<td></td>');
     detectedAtCol.attr('id', 'time'+i);

     row.append(indexCol);
     row.append(imgCol);
     row.append(detectByCol);
     row.append(detectedDateCol);
     row.append(detectedAtCol);

     $('#tablebody').append(row);
```

```
      $("#img_src"+i).attr("src","http://s3.amazonaws.com/smartgarden-
wonderwoman/plant/"+ndata[i].filename);
      $("#time"+i).text(jstime);
      $("#date"+i).text(uploaddate);
      $("#detectby"+i).text(ndata[i].updatedBy);
     }
}//end success

       });
   }




       function loadFlowerPic(){

             $('#tablebody').html('');
   $('#flowerRad').prop('disabled', true);
     $("#plantRad").removeAttr('disabled');
   $('#pagetitle').text('Bugs on Flower');

   $("#flowerRad").removeClass("active");
       $("#plantRad").removeClass("disabled");
   $("#flowerRad").addClass("disabled");
       $("#plantRad").addClass("active");
      $.ajax({
   url: '/api/getImg/flower',
  type: 'POST',
  success: function(ndata){
  console.log(ndata);
    for (i in ndata) {
      console.log(ndata[i]);
    datetime = ndata[i].datetimeid;
       jsdatetime = new Date(Date.parse(datetime));
       jstime = jsdatetime.toLocaleTimeString();
      console.log("pic" + i + ndata[i].filename);

      let row = $('<tr></tr>');

      let indexCol = $('<td scope="row"></td>');
      indexCol.text("#"+(i*1+1));

      let imgCol = $('<td class="w-25"></td>');
      let imgEle = $('<img class="img-fluid img-thumbnail" alt="imgage">');
      imgEle.attr('id', 'img_src'+i);
```

```
        imgCol.append(imgEle);

        let detectByCol = $('<td></td>');
        detectByCol.attr('id', 'detectby'+i);

        let detectedDateCol = $('<td></td>');
        detectedDateCol.attr('id', 'date'+i);

            let detectedAtCol = $('<td></td>');
        detectedAtCol.attr('id', 'time'+i);

        row.append(indexCol);
        row.append(imgCol);
        row.append(detectByCol);
        row.append(detectedDateCol);
        row.append(detectedAtCol);

        $('#tablebody').append(row);

        $('#tablebody').append(row);

        $("#img_src"+i).attr("src","http://s3.amazonaws.com/smartgarden-
wonderwoman/flower/"+ndata[i].filename);
        $("#time"+i).text(jstime);
        $("#date"+i).text(uploaddate);
        $("#detectby"+i).text(ndata[i].updatedBy);
       }
}//end success

         });
     }


    $('#logout').on('click',function () {
   window.location.replace("logout");

});

  </script>

</body>

</html>
```

Server files

server.py

```
from flask import Flask, render_template, jsonify,  url_for, redirect, request,Respo
nse, session

import mysql.connector
import sys

import json
import numpy
import datetime
import decimal

import dynamodb as awsdb
import gevent
import gevent.monkey
from gevent.pywsgi import WSGIServer
import os


app = Flask(__name__)
gevent.monkey.patch_all()
app.secret_key = os.urandom(12)

print('The server is running!')

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

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

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


# pages
@app.route("/")
@app.route("/index.html")
def dashboard():
  if not session.get('logged_in'):
    return redirect(url_for('login'))
  else:
    return render_template('index.html')



@app.route("/charts.html")
def graph():
  if not session.get('logged_in'):
    return redirect(url_for('login'))
  else:
    return render_template('/charts.html')


@app.route("/tables.html")
def table():
  if not session.get('logged_in'):
    return redirect(url_for('login'))
  else:
    return render_template('/tables.html')


@app.route("/gallery.html")
def gallery():
  if not session.get('logged_in'):
    return redirect(url_for('login'))
  else:
    return render_template('/gallery.html')



@app.route("/api/verify/<name>/<password>", methods=['POST', 'GET'])
def verify(name,password):
  if request.method == 'POST':
    try:
```

```
      data = awsdb.login()
      loggedin = False
      for user in data:
        print(user['username'])
        print(user['password'])
        if user['username'] == name and user['password'] == password:
          loggedin = True
          session['logged_in'] = True
          return ("success")
      if loggedin == False:
        return ("fail")
    except:
      print(sys.exc_info()[0])
      print(sys.exc_info()[1])
      return None



@app.route("/logout")
def logout():
  session.pop('logged_in', None)
  return redirect(url_for('login'))

# api routes
@app.route("/api/getData", methods=['POST', 'GET'])
def api_getData():
  if request.method == 'POST':
    try:
      data = data_to_json(awsdb.get_data())
      loaded_data = json.loads(data)
      print(loaded_data)
      return jsonify(loaded_data)
    except:
      print(sys.exc_info()[0])
      print(sys.exc_info()[1])
      return None


@app.route("/api/status", methods=['GET', 'POST'])
def status():
  if request.method == 'POST':
    try:
      data = data_to_json(awsdb.get_status())
      loaded_data = json.loads(data)
      print(loaded_data)
      return jsonify(loaded_data)
```

```
    except:
      print(sys.exc_info()[0])
      print(sys.exc_info()[1])
      return None



@app.route("/api/getChart", methods=['POST', 'GET'])
def api_getChartData():
  if request.method == 'POST':
    try:
      data = data_to_json(awsdb.get_chart_data())
      loaded_data = json.loads(data)
      print(loaded_data)
      return jsonify(loaded_data)
    except:
      print(sys.exc_info()[0])
      print(sys.exc_info()[1])
      return None



@app.route("/api/changeStatus/<status>", methods=['GET', 'POST'])
def changeStatus(status):
  if request.method == 'POST':
    try:
      awsdb.send_status(status)
      print(status)
      return status
    except:
      print(sys.exc_info()[0])
      print(sys.exc_info()[1])
      return None



@app.route("/api/getAllStatus", methods=['POST', 'GET'])
def getAllStatus():
  if request.method == 'POST':
    try:
      data = data_to_json(awsdb.get_all_status())
      loaded_data = json.loads(data)
      print(loaded_data)
      return jsonify(loaded_data)
```

```
    except:
      print(sys.exc_info()[0])
      print(sys.exc_info()[1])
      return None



@app.route("/api/getImg/<type>", methods=['POST', 'GET'])
def getImg(type):
  if request.method == 'POST':
    try:
      data = data_to_json(awsdb.get_Img_url(type))
      loaded_data = json.loads(data)
      print(loaded_data)
      return jsonify(loaded_data)
    except:
      print(sys.exc_info()[0])
      print(sys.exc_info()[1])
      return None




if __name__ == '__main__':
  try:
    http_server = WSGIServer(('0.0.0.0', 8009), app)
    app.debug = True
    http_server.serve_forever()
    print('Server waiting for requests')
  except:
    print("Exception")
    print(sys.exc_info()[0])
    print(sys.exc_info()[1])
```

Section 6
Task List

Contribution by Jiang Lei

What was done by in CA1

- Create Arduino codes for water pump, Soil Moisture Sensor, LDR, DHT11 in Arduino
- 2 systems of water pump
- Used LCD to display real-time values
- Login feature
- 4 Charts and 2 Tables of historical data
- Twilio SMS Alerting message

What was done by in CA2

- Connect Hardware to Arduino except the MFRC522 Card reader
- Get data from sensors and write data to water pump
- Store and read data from DynamoDB
- Set up AWS S3 bucket and store images
- Set the water pump to work in 2 systems
- Activate 2 cameras and take pictures based on user command from telegram
- Perform image detection by using Rekognition
- Display charts, tables and gallery of images from S3 bucket in web server
- Consolidate the team coding

Contribution by Zi Yun

What was done by in CA1

- Portable LED light
- Used Soil Moisture Sensor to detect soil value
- Used LDR to detect light level
- Used LCD to display light values
- Twilio SMS Alerting message

What was done by in CA2

- Fritzing Diagram
- Setting up AWS services (Roles, Rules, Policy)
- Setting up DynamoDB
- Creating Twitter Developer account
- Setting up nfc codes and hardware with Arduino
- Setting up Aws_pubsub scripts (aws_pubsub_data.py, aws_pubsub_status.py, scripts.py)
- Step by step tutorial

Contribution by Jorin Seah

What was done by in CA1

- LDR sensor to detect light level
- LED light bulb auto on if light level is low
- Remote control to on/off lightbulb using telegram
- Twilio SMS real-time notification alert

What was done by in CA2

- Setting up AWS services (Roles, Rules, Policy)
- Setting up DynamoDB
- Creating Lambda function for Telegram
- Creating Twilio for Whatsapp
- Creating SNS services (Email, Telegram, lambda)
- Setting up telegram bots
- Step by step tutorial
- Editing videos

Section 7
References

References to online materials used

Inspiration of Smart Garden:
https://www.hackster.io/bobbyleonard84/python-micropython-iot-framework-example-auto- irrigation-6286ae
https://www.pubnub.com/blog/smart-automated-iot-plant-irrigation-system-raspberry-pi-pubnub/ https://www.hackster.io/ben-eagan/raspberry-pi-automated-plant-watering-with-website-8af2dc https://www.instructables.com/id/Raspberry-Pi-Powered-IOT-Garden/ https://www.hackster.io/mokxf16/smart-garden-raspberry-pi-arduino-65c7b7
https://www.hackster.io/pohwl/ecoplant-plant-monitoring-5e6767

Arduino codes study:
https://www.hackster.io/electronicprojects/smart-irrigation-system-using-arduino-uno-afcb31

Serial Communication study:
https://www.ladyada.net/learn/arduino/lesson4.html

Fetchall() study, for mysql select syntax:
https://pynative.com/python-cursor-fetchall-fetchmany-fetchone-to-read-rows-from-table/

Secret Key for flask:
https://www.pythonanywhere.com/forums/topic/12800/
https://stackoverflow.com/questions/34902378/where-do-i-get-a-secret-key-for-flask

Soil Moisture Circuit research:
http://www.circuitstoday.com/arduino-soil-moisture-sensor

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