SCHOOL OF DIGITAL MEDIA AND INFOCOMM TECHNOLOGY (DMIT)

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: 22nd August 2017

Prepared for: Ms Dora Chua

Class: DIT/FT/3A/42
DISM/FT/3A/01

Submitted by:

Student ID	Name
1444559	Gabriel Kok Heng Hong
1530333	Shawn Liow Peng
1551271	Bavani D/o Raman

Table of Contents

Section 1
Overview of project

A. Where we have uploaded our tutorial

https://github.com/Bavani94/Greenhouse-IoT-

B. Why have we chosen to upload to this site

GitHub is a development platform build for developers. From open source to business. User can host and review the codes, manage projects or even build software alongside millions of other developers. **GitHub** is a web-based Git repository and Internet hosting service.

C. What have we uploaded

All the codes, Step by step tutorial, Guide on how to setup Twilio, Firebase and AWS, CA2 folder which consists of the .fzz files

D. What is the application about?

The aim of this project was to design and build a greenhouse that helps to sustain plant life automatically without human intervention, by acting upon live sensor readings and be able to display the status of the system to the owner.

The project was split into two parts: programming that manages the sensors and transducers; and creating a web application to allow the user to interact with the greenhouse.

E. Summary of the steps that will be described

Provide a bullet list of the steps that will be covered in the other parts of this tutorial

	Section	Description
1)	Overview	Get an overview of what you will do for the lab
Sections 2 to 5 will guide you through the steps of the building of greenhourse		
2)	Hardware requirements	Provides overview of hardware required
3)	Connect The Components	Guide you on how to connect the T-Cobble to Hardware
4)	Softwares/External Libraries /Templates	Shows the Softwares/External Libraries /Templates that are being used
5)	Programming Codes	Shows the code

Section 2
Hardware Requirements

Hardware Background

Background

This section talks about which environmental factors will be monitored by the automated system and discusses the choices of sensor or actuatorthat were available to use.

Temperature
The temperature of the environment is extremely important for plants as it effects multiple growing factors. Extreme temperatures can negatively impact plant productivity so maintaining the temperature in a greenhouse is crucial. Each plant also has its own specific temperature range so being able to adjust the settings in the greenhouse is equally as important.

Humidity
Humidity is the measure of how much moisture is present in the air. When plants transpire, water vapour are released from the leaves, increasing the humidity.

A high humidity can be fatal to plants if it is not monitored, as a build-up of moisture on plants. It is important to make sure that air is circulating through the greenhouse to reduce the water vapour around the plants.

Light
Plants use the energy from light for photosynthesis and so without light, plants would not be able to grow. Different plant need different amounts of light but if a plant receives too little light then problems will start occurring, such as

- Leaves turn yellow
- Leaves are too small
- Leaves dry up

Soil Moisture
Water is another key element in photosynthesis. Plants absorb water through their roots where it then travels up the stem and into the leaves where photosynthesis occurs. It's important to monitor the moisture levels in the soil, making sure that there is enough water for the plant but also taking care not to over saturate the soil, as this could cause the roots to rot.

Water Pump
Remembering to water the plant can be hard when we are busy. So instead of being countless disappointing harvests, Water pump can be used to maintain the water levels in the soil.

Small Fan
It will help to move air around to simulate humidity control so that there will be air for the plant

Hardware Design

Design
The temperature, humidity, light and soil moisture levels as these are four of the most important environmental factors during plant growth. The design step for this was to think about what can be done to control these factors.

Dealing with Temperature
All plants have a temperature range that they grow best in, which is why we see different plants at different times of the year. Greenhouse should be able to maintain a specific temperature range for however long is necessary.

If the temperature exceeds the maximum allowed temperature, the greenhouse needs cooling down in some way. In industry this would usually be achieved by opening vents or turning on fans, letting air circulate through the greenhouse. When the temperature becomes too low, the greenhouse would need to be heated up and all vents and fans would be closed and powered down.

For the project, fans is being used to turned on when the temperature is too high. If the temperature becomes too low.

Dealing with Humidity
The most effective way to maintain a low humidity is to keep the greenhouse dry, which can be done by warming up plants and keeping air circulating through it. If the humidity is too low however, then moisture needs to be added to the air.

The fans mentioned in ‘Dealing with Temperature’ can be used to reduce the humidity if it becomes too high.

Dealing with Soil Moisture
Plants absorb water through their roots for use in photosynthesis. This, along with other factors, such as a high temperature, will dry out the soil. Maintaining the soil moisture content is therefore essential for plants to continue to grow healthily and can be achieved in different ways. One of which would be to sprinkle water on the soil after a certain period.

Hardware Checklist

Temperature and Humidity

The DHT11 is a basic, digital temperature and humidity sensor. It will be used to track the temperature and humidity of the immediate surroundings of the plant. It's simple to use but requires careful timing to capture data.

The sensor has 4 pins
- VCC (this is where you connect to power)
- DATA (this is an output value you read to determine)
- NC (stands for no connection)
- GND (connect to ground)

Light Resistant Diode (LDR)

It's a light sensitive resistors which change resistance based on how much light they are exposed. It helps to track the light levels in the enclosure. This is not particularly good for accurate measurements but it is enough to indicate when it becomes dark.

LED

The longer leg (anode) should be connected to the positive supply of the circuit (power). The shorter leg (cathode) should be connected to the negative side of the power supply (ground). It will only work if the power is supplied in the correct way. LED will light up the enclosure when light levels are Low.

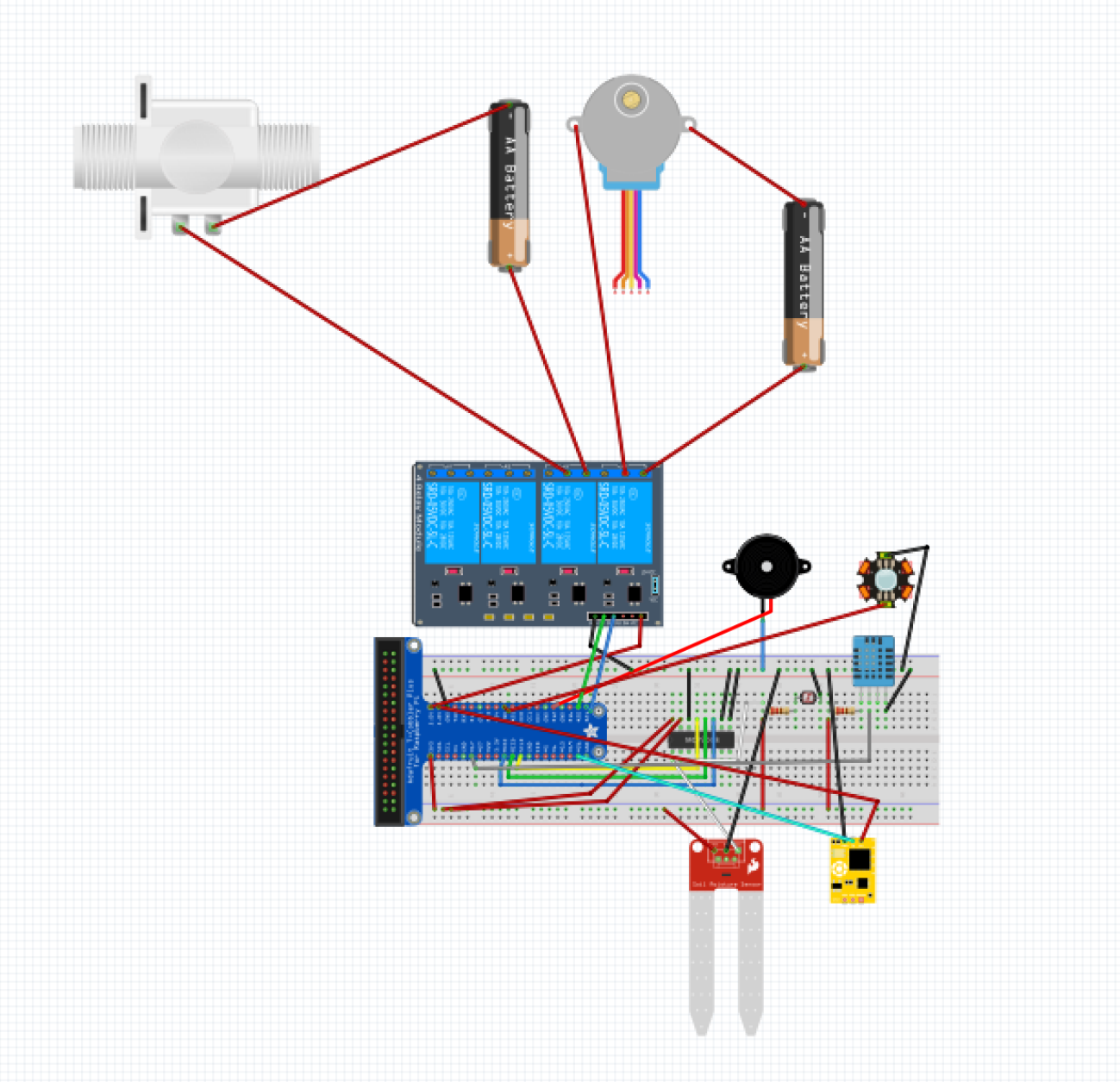
Soil Moisture Sensor

The Soil Moisture sensor can read the amount of moisture present in the soil surrounding it. It's a low tech sensor, but ideal for monitoring gardens. It is used for the soil moisture measurements as it will be inserted into the soil to track the soil moisture.

Small Generic FAN

The Fan voltage is bout 6 and it is to move air around to simulate humidity control so that the air can flow all the way through it. Whether the fan is used to take in or extract air simply depends on which direction it is facing, there is no difference in the way it is controlled.

Section 3
Connect your Components



Section 4
Softwares/External Libraries/Templates

Operating System:

Database:

- Firebase (https://firebase.google.com/)
- AWS (https://aws.amazon.com/)

SMS Service:

- Twilio (https://www.twilio.com/)

- To view step by step guide to create the Database and SMS Service go to another PDF file if not to the github link which is even in the first part .

External Libraries:

- Pyrebase (Firebase Library) (https://github.com/thisbejim/Pyrebase)
- PyCloudinary (Cloudinary library, where our images will be stored) (https://github.com/cloudinary/pycloudinary)
- gpiozero (https://gpiozero.readthedocs.io/en/stable/)

Templates:

-

Section 5
Programming Codes

A.Write code for dht22_test.py

- Create a python script dht22_test.py
- Sudo nano ~/CA2/ /dht22_test.py

```
import sys
import Adafruit_DHT
from time import sleep

pin = 26

while True:
 humidity, temperature = Adafruit_DHT.read_retry(11, pin)
 print('Temp: {:.1f} C'.format(temperature))
 print('Humidity: {:.1f}'.format(humidity))
 sleep(1)
```

B.Write code for fan_test.py

- Create a python script fan_test.py
- Sudo nano ~/CA2/ /fan_test.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# init list with pin numbers

pinList = [12]

# loop through pins and set mode and state to 'high'

for i in pinList:
    GPIO.setup(i, GPIO.OUT)
    GPIO.output(i, GPIO.HIGH)

# time to sleep between operations in the main loop

SleepTimeL = 2

# main loop

try:
  GPIO.output(12, GPIO.LOW)
  print "ONE"
  print ("Fan is on.")
  time.sleep(60);
  GPIO.cleanup()
  print "Good bye!"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()


# find more information on this script at
# http://youtu.be/WpM1aq4B8-A
```

C.Write code for full.py

- Create a python script full.py
  - Sudo nano ~/CA2/ /full.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# init list with pin numbers

pinList = [4, 17, 27, 22]

# loop through pins and set mode and state to 'high'

for i in pinList:
    GPIO.setup(i, GPIO.OUT)
    GPIO.output(i, GPIO.HIGH)

# time to sleep between operations in the main loop

SleepTimeL = 60
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
# main loop

try:
  GPIO.output(17, GPIO.LOW)
  GPIO.output(4, GPIO.LOW)
  print "Fan and Lights are ON"
  GPIO.output(20, GPIO.HIGH)
  print "Buzzer is ON"
  GPIO.output(21, GPIO.HIGH)
  print "Light is ON"
  time.sleep(SleepTimeL)
  GPIO.cleanup()
  print "Good bye!"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()


# find more information on this script at
# http://youtu.be/WpM1aq4B8-A
```

D.Write code for firebase_data.py

- Create a python script firebase_data.py
  - Sudo nano ~/CA2/ /firebase_data.py

```
import pyrebase
import time
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import json

# Configure Firebase
pyrebase_config = {
  "apiKey": " AIzaSyAlX8g6Ot35en0SAlY5LPP3DdMFD8ObG50",
  "authDomain": "iotca2-7591e.firebaseapp.com",
  "databaseURL": "https://iotca2-7591e.firebaseio.com",
  "storageBucket": "iotca2-7591e.appspot.com",
  "serviceAccount": "iotca2-7591e-firebase-adminsdk-e7sbb-19bff1dd8e.json"
}
```

```
pyrebaseDatabase = pyrebase.initialize_app(pyrebase_config)
db = pyrebaseDatabase.database()

# Configure MQTT
host = "a2g49f96gkk39l.iot.ap-southeast-1.amazonaws.com"
rootCApath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

def lightCallBack(client, userdata, message):
  print("Received new message: ")
  print(message.payload)
  light_data = json.loads(message.payload)
  print(light_data)
  print("Sending to Firebase...")
  result = db.child("light_readings").push(light_data)
  print("Done sending to Firebase.")
  print("From topic:")
  print(message.topic)
  print("----------------------")

def waterCallBack(client, userdata, message):
  print("Received new message: ")
  print(message.payload)
  water_data = json.loads(message.payload)
  print(water_data)
  print("Sending to Firebase...")
  result = db.child("water_level").push(water_data)
  print("Done sending to Firebase.")
  print("From topic:")
  print(message.topic)
  print("----------------------")

def tempHumCallBack(client, userdata, message):
  print("Received new message: ")
  print(message.payload)
  temp_hum_data = json.loads(message.payload)
  print(temp_hum_data)
  print("Sending to Firebase...")
  result = db.child("temperature_humidity_readings").push(temp_hum_data)
  print("Done sending to Firebase.")
  print("From topic:")
  print(message.topic)
  print("----------------------")

def soilCallBack(client, userdata, message):
  print("Received new message: ")
  print(message.payload)
  soil_data = json.loads(message.payload)
  print(soil_data)
  print("Sending to Firebase...")
  result = db.child("soil_moisture_readings").push(soil_data)
  print("Done sending to Firebase.")
  print("From topic:")
  print(message.topic)
  print("----------------------")

rpi = AWSIoTMQTTClient("Firebase")
rpi.configureEndpoint(host, 8883)
rpi.configureCredentials(rootCApath, privateKeyPath, certificatePath)
rpi.configureOfflinePublishQueueing(-1)
rpi.configureDrainingFrequency(2)
rpi.configureConnectDisconnectTimeout(10)
rpi.configureMQTTOperationTimeout(5)
rpi.connect()
rpi.subscribe("sensors/light", 1, lightCallBack)
rpi.subscribe("sensors/waterlevel", 1, waterCallBack)
rpi.subscribe("sensors/temphum", 1, tempHumCallBack)
rpi.subscribe("sensors/soil", 1, soilCallBack)

while True:
  print("Waiting for MQTT message.")
  time.sleep(5)
```

E. Write code for light.py

- Create a python script light.py
  - Sudo nano ~/CA2/ /light.py

```
import spidev
import time
from datetime import datetime
import os
import cloudinary
import cloudinary.uploader
import cloudinary.api
import pyrebase
import logging
import json
import RPi.GPIO as GPIO

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

# Configure Cloudinary
cloudinary.config(
  cloud_name = "iotca2",
  api_key = "732475871262856",
  api_secret = "XbjpVSAgUc_PypzoipnXkHyPmG0"
)

# Configure Firebase
pyrebase_config = {
  "apiKey": " AIzaSyAlX8g6Ot35en0SAlY5LPP3DdMFD8ObG50",
  "authDomain": "iotca2-7591e.firebaseapp.com",
  "databaseURL": "https://iotca2-7591e.firebaseio.com",
  "storageBucket": "iotca2-7591e.appspot.com",
  "serviceAccount": "iotca2-7591e-firebase-adminsdk-e7sbb-19bff1dd8e.json"
}

pyrebaseDatabase = pyrebase.initialize_app(pyrebase_config)
db = pyrebaseDatabase.database()

# Configure GPIO
relay_pin = 4

# Function to read SPI data from MCP3008 chip
def ReadChannel(channel):
   adc = spi.xfer2([1,(8+channel)<<4,0])
   data = ((adc[1]&3) << 8) + adc[2]
   return data

# Main loop - read raw data and display
while True:
   light_value = ReadChannel(1)
   print("Light Value: " + str(light_value))
   time.sleep(0.1)
```

F. Write code for light_automation.py

- Create a python script light_automation.py
  - Sudo nano ~/CA2/ /light_automation.py

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import spidev
import time
from datetime import datetime
import os
import cloudinary
import cloudinary.uploader
import cloudinary.api
```

```
import pyrebase
import logging
import json
import RPi.GPIO as GPIO

count = 0

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

# Configure Cloudinary
cloudinary.config(
  cloud_name = "iotca2",
  api_key = "732475871262856",
  api_secret = "XbjpVSAgUc_PypzoipnXkHyPmG0"
)

# Configure Firebase
pyrebase_config = {
  "apiKey": " AIzaSyAlX8g6Ot35en0SAlY5LPP3DdMFD8ObG50",
  "authDomain": "iotca2-7591e.firebaseapp.com",
  "databaseURL": "https://iotca2-7591e.firebaseio.com",
  "storageBucket": "iotca2-7591e.appspot.com",
  "serviceAccount": "iotca2-7591e-firebase-adminsdk-e7sbb-19bff1dd8e.json"
}

pyrebaseDatabase = pyrebase.initialize_app(pyrebase_config)
db = pyrebaseDatabase.database()
duration = db.child("led_status").child("duration").get().val()
threshold = db.child("led_status").child("threshold").get().val()

#Configure MQTT
host = "a2g49f96gkk39l.iot.ap-southeast-1.amazonaws.com"
rootCApath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

def customCallback(client, userdata, message):
  global duration
  global threshold
  global count
  count = 0
  print('count resetted to value : ' + str(count))
  print("Received new message: ")
  print(message.payload)
  complete = True
  #pump_config = json.loads(message.payload)
  #duration = pump_config['duration']
  #threshold = pump_config['threshold']
  print("From topic:")
  print(message.topic)
  print("----------------------")

rpi = AWSIoTMQTTClient("light")
rpi.configureEndpoint(host, 8883)
rpi.configureCredentials(rootCApath, privateKeyPath, certificatePath)
rpi.configureOfflinePublishQueueing(-1)
rpi.configureDrainingFrequency(2)
rpi.configureConnectDisconnectTimeout(10)
rpi.configureMQTTOperationTimeout(5)
rpi.connect()
rpi.subscribe("sensors/lightconfig", 1, customCallback)

# Configure GPIO
led_pin = 18

# Function to read SPI data from MCP3008 chip
def ReadChannel(channel):
   adc = spi.xfer2([1,(8+channel)<<4,0])
   data = ((adc[1]&3) << 8) + adc[2]
   return data

# Main loop - read raw data and display
while True:
```

```
  global count
  print("Subscribed duration: " + str(duration))
  GPIO.setmode(GPIO.BCM)
  GPIO.setup(led_pin, GPIO.OUT)
  GPIO.output(led_pin, GPIO.LOW)
  light_value = ReadChannel(1)
  print("Light Value: " + str(light_value))

  if (light_value > threshold):
    GPIO.output(led_pin, GPIO.HIGH)
    count += 1
    time.sleep(duration)
  else:
    count += 1
    GPIO.output(led_pin, GPIO.LOW)

  data = { 'timestamp': int(time.time()), 'light_value': int(light_value)}
  # result = db.child("light_readings").push(data)
  json_string = json.dumps(data)
  print("Publishing to AWS MQTT Broker.")
  rpi.publish("sensors/light", json_string, 1)
  print("Done publishing to AWS MQTT Broker.")
  time.sleep(1)
  if (count == 1):
    GPIO.cleanup()
```

G. Write code for publish_light_mqtt.py

- Create a python script publish_light_mqtt.py
  - Sudo nano ~/CA2/ publish_light_mqtt.py

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from time import sleep
import pyrebase
import spidev
import time
import json

spi= spidev.SpiDev()
spi.open(0,0)

# Configure Firebase
pyrebase_config = {
  "apiKey": " AIzaSyAlX8g6Ot35en0SAlY5LPP3DdMFD8ObG50",
  "authDomain": "iotca2-7591e.firebaseapp.com",
  "databaseURL": "https://iotca2-7591e.firebaseio.com",
  "storageBucket": "iotca2-7591e.appspot.com",
  "serviceAccount": "iotca2-7591e-firebase-adminsdk-e7sbb-19bff1dd8e.json"
}

pyrebaseDatabase = pyrebase.initialize_app(pyrebase_config)
db = pyrebaseDatabase.database()

# Function to read SPI data from MCP3008 chip
def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data

def customCallback(client, userdata, message):
        print("Received new message: ")
        print(message.payload)
        print("From topic:")
        print(message.topic)
        print("----------------------")

host = "a2g49f96gkk39l.iot.ap-southeast-1.amazonaws.com"
rootCApath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"
```

```
rpi = AWSIoTMQTTClient("basicPubSub")
rpi.configureEndpoint(host, 8883)
rpi.configureCredentials(rootCApath, privateKeyPath, certificatePath)
rpi.configureOfflinePublishQueueing(-1)
rpi.configureDrainingFrequency(2)
rpi.configureConnectDisconnectTimeout(10)
rpi.configureMQTTOperationTimeout(5)

rpi.connect()
rpi.subscribe("sensors/light", 1, customCallback)
sleep(2)

while True:
        light_value = ReadChannel(1)
        data = {'date': int(time.time()), 'light_value': str(light_value)}
        result = db.child("test_light_mqtt").push(data)
        json_string = json.dumps(data)
        rpi.publish("sensors/light", json_string, 1)
        sleep(2)
```

H. Write code for soil_moisture.py

- Create a python script soil_moisture.py
- Sudo nano ~/CA2/ / soil_moisture.py

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import spidev
import time
from datetime import datetime
import os
import cloudinary
import cloudinary.uploader
import cloudinary.api
import pyrebase
import logging
import json
import RPi.GPIO as GPIO

count = 0

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

# Configure Cloudinary
cloudinary.config(
  cloud_name = "iotca2",
  api_key = "732475871262856",
  api_secret = "XbjpVSAgUc_PypzoipnXkHyPmG0"
)

# Configure Firebase
pyrebase_config = {
  "apiKey": " AIzaSyAlX8g6Ot35en0SAlY5LPP3DdMFD8ObG50",
  "authDomain": "iotca2-7591e.firebaseapp.com",
  "databaseURL": "https://iotca2-7591e.firebaseio.com",
  "storageBucket": "iotca2-7591e.appspot.com",
  "serviceAccount": "iotca2-7591e-firebase-adminsdk-e7sbb-19bff1dd8e.json"
}

pyrebaseDatabase = pyrebase.initialize_app(pyrebase_config)
db = pyrebaseDatabase.database()
duration = db.child("pump_status").child("duration").get().val()
threshold = db.child("pump_status").child("threshold").get().val()

#Configure MQTT
host = "a2g49f96gkk39l.iot.ap-southeast-1.amazonaws.com"
rootCApath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"
```

```
def customCallback(client, userdata, message):
  global duration
  global threshold
  global count
  count = 0
  print("Received new message: ")
  print(message.payload)
  pump_config = json.loads(message.payload)
  # duration = pump_config['duration']
  threshold = int(pump_config['threshold'])
  print("New threshold: " + str(threshold))
  print("From topic:")
  print(message.topic)
  print("----------------------")

rpi = AWSIoTMQTTClient("soil")
rpi.configureEndpoint(host, 8883)
rpi.configureCredentials(rootCApath, privateKeyPath, certificatePath)
rpi.configureOfflinePublishQueueing(-1)
rpi.configureDrainingFrequency(2)
rpi.configureConnectDisconnectTimeout(10)
rpi.configureMQTTOperationTimeout(5)
rpi.connect()
rpi.subscribe("sensors/soilconfig", 1, customCallback)

# Configure GPIO
relay_pin = 20

# Function to read SPI data from MCP3008 chip
def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data

# Main loop - read raw data and display
while True:
  global count
  GPIO.setmode(GPIO.BCM)
  GPIO.setup(relay_pin, GPIO.OUT)
  GPIO.output(relay_pin, GPIO.HIGH)
  print("Relay Pin is now set to HIGH")
  moisture_value = ReadChannel(0)
  print("Moisture value taken.")
  moisture_percentage = 100.0000 - (moisture_value / 10.24)
  print("Dryness percentage calculated.")
  print ("Moisture Value: " + str(moisture_value) + " Moisture Percentage: " + str(round(moisture_percentage, 2)) + "%")
  print("Threshold: " + str(threshold))

  if (moisture_percentage < threshold):
     GPIO.output(relay_pin, GPIO.LOW)
     count += 1
     print("Relay is now turning on.")
     time.sleep(duration)
     print("Relay is now off.")
  else:
     GPIO.output(relay_pin, GPIO.HIGH)
     count += 1

  data = { 'timestamp': int(time.time()), 'moisture_value': str(moisture_value), 'moisture_percentage': str(round(moisture_percentage, 2))}
  # result = db.child("soil_moisture_readings").push(data)
  json_string = json.dumps(data)
  print("Publishing to AWS MQTT Broker...")
  rpi.publish("sensors/soil", json_string, 1)
  print("Done publishing to AWS MQTT Broker.")
  time.sleep(1)
  if (count == 1):
    GPIO.cleanup()
```

I. Write code for receive_commands.py

- Create a python script receive_commands.py
  - Sudo nano ~/CA2/ / receive_commands.py

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import RPi.GPIO as GPIO
import json
import cloudinary
import cloudinary.uploader
import cloudinary.api
import time
import pyrebase
import picamera

c = picamera.PiCamera()
bools = False

# Configure Firebase
pyrebase_config = {
  "apiKey": " AIzaSyAlX8g6Ot35en0SAlY5LPP3DdMFD8ObG50",
  "authDomain": "iotca2-7591e.firebaseapp.com",
  "databaseURL": "https://iotca2-7591e.firebaseio.com",
  "storageBucket": "iotca2-7591e.appspot.com",
  "serviceAccount": "iotca2-7591e-firebase-adminsdk-e7sbb-19bff1dd8e.json"
}

pyrebaseDatabase = pyrebase.initialize_app(pyrebase_config)
db = pyrebaseDatabase.database()

# Cloudinary Configuration
cloudinary.config(
  cloud_name = "overkill",
  api_key = "876111269332755",
  api_secret = "8AMs3sZqg5HgEJiRFr9HuEUwzoA"
)
camera = 888
lightValue = None
lightStatus = None
fanValue = None
fanStatus = None

pumpValue = None
pumpStatus = None

cameraValue = None
# Configure GPIO
led_pin = 18
water_pump_pin = 20
fan_pin = 21

GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)
GPIO.setup(water_pump_pin, GPIO.OUT)
GPIO.setup(fan_pin, GPIO.OUT)

# LED
GPIO.output(led_pin, GPIO.LOW)

# Water Pump
GPIO.output(water_pump_pin, GPIO.HIGH)

# Fan
GPIO.output(fan_pin, GPIO.LOW)

#Configure MQTT
host = "a2g49f96gkk39l.iot.ap-southeast-1.amazonaws.com"
rootCApath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

def lightControlCallback(client, userdata, message):
  global lightValue
```

```
  global lightStatus
  print("Received new message: ")
  print(message.payload)
  json_data = json.loads(message.payload)
  lightValue = int(json_data["value"])
  lightStatus = json_data["status"]
  print("From topic:")
  print(message.topic)
  print("----------------------")

def pumpControlCallback(client, userdata, message):
  global pumpValue
  global pumpStatus
  global pumpDuration
  print("Received new message: ")
  print(message.payload)
  json_data = json.loads(message.payload)
  pumpValue = int(json_data["value"])
  pumpStatus = json_data["status"]
  pumpDuration = json_data["duration"]
  print("From topic:")
  print(message.topic)
  print("----------------------")

def fanControlCallback(client, userdata, message):
  global fanValue
  global fanStatus
  print("Received new message: ")
  print(message.payload)
  json_data = json.loads(message.payload)
  fanValue = int(json_data["value"])
  fanStatus = json_data["status"]
  print("From topic:")
  print(message.topic)
  print("----------------------")

def cameraControlCallback(client, userdata, message):
  global cameraValue
  global bools
  print("Received new message: ")
  print(message.payload)
  json_data = json.loads(message.payload)
  cameraValue = int(json_data["value"])
  print("From topic:")
  print(message.topic)
  print("----------------------")
  bools = False


rpi = AWSIoTMQTTClient("control")
rpi.configureEndpoint(host, 8883)
rpi.configureCredentials(rootCApath, privateKeyPath, certificatePath)
rpi.configureOfflinePublishQueueing(-1)
rpi.configureDrainingFrequency(2)
rpi.configureConnectDisconnectTimeout(10)
rpi.configureMQTTOperationTimeout(5)
rpi.connect()
rpi.subscribe("sensors/commands/light", 1, lightControlCallback)
rpi.subscribe("sensors/commands/pump", 1, pumpControlCallback)
rpi.subscribe("sensors/commands/fan", 1, fanControlCallback)
rpi.subscribe("sensors/commands/camera", 1, cameraControlCallback)

while True:
  pumpDuration = db.child("pump_status").child("duration").get().val()
  print("Waiting for command.")
  # Logic for LED
  if lightValue == led_pin:
    if lightStatus == 'On':
      GPIO.output(led_pin, GPIO.HIGH)
      print("LED turned on.")
    else:
      GPIO.output(led_pin, GPIO.LOW)
      print("LED turned off.")
    # Logic for Water Pump
  if pumpValue == water_pump_pin:
```

```
    if pumpStatus == 'On':
      GPIO.output(water_pump_pin, GPIO.LOW)
      time.sleep(pumpDuration)
      print("Water pump turned on.")
    else:
      GPIO.output(water_pump_pin, GPIO.HIGH)
      print("Water pump turned off.")
  # Logic for Fan
  if fanValue == fan_pin:
    if fanStatus == 'On':
      GPIO.output(fan_pin, GPIO.HIGH)
      print("Fan turned on.")
    else:
      GPIO.output(fan_pin, GPIO.LOW)
      print("Fan turned off.")
    # Logic for Camera
  if cameraValue == camera and bools == False:
    global bools
    bools = True
    timestamp = int(time.time())
    filename = '/home/pi/photo_'+str(timestamp)+'.jpg'
    c.capture(filename)
    upload_response = cloudinary.uploader.upload_large(filename, resource_type = "auto")
    url = str(upload_response['secure_url'])
    data = { 'image_url' : url, 'date': timestamp }
    result = db.child("captured_images").push(data)
  time.sleep(5)
```

J. Write code for relay_test.py

- Create a python script relay_test.py
- Sudo nano ~/CA2/ /relay_test.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

# init list with pin numbers

pinList = [4, 17, 27, 22]

# loop through pins and set mode and state to 'high'

for i in pinList:
    GPIO.setup(i, GPIO.OUT)
    GPIO.output(i, GPIO.HIGH)

# time to sleep between operations in the main loop

SleepTimeL = 60

# main loop

try:
  GPIO.output(17, GPIO.LOW)
  GPIO.output(4, GPIO.LOW)
  print "Fan and Pump is ON"
  time.sleep(SleepTimeL);
  print "TWO"
  time.sleep(SleepTimeL);
  GPIO.output(27, GPIO.LOW)
  print "THREE"
  time.sleep(SleepTimeL);
  GPIO.output(22, GPIO.LOW)
  print "FOUR"
  time.sleep(SleepTimeL);
  GPIO.cleanup()
  print "Good bye!"
```

```
# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()


# find more information on this script at
# http://youtu.be/WpM1aq4B8-A
```

K. Write code for temperature_humidity.py

- Create a python script temperature_humidity.py
- Sudo nano ~/CA2/ / temperature_humidity.py

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import spidev
import time
from datetime import datetime
import os
import cloudinary
import cloudinary.uploader
import cloudinary.api
import Adafruit_DHT
import pyrebase
import logging
import json
import RPi.GPIO as GPIO

count = 0

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

# Configure Cloudinary
cloudinary.config(
  cloud_name = "iotca2",
  api_key = "732475871262856",
  api_secret = "XbjpVSAgUc_PypzoipnXkHyPmG0"
)

# Configure Firebase
pyrebase_config = {
  "apiKey": " AIzaSyAlX8g6Ot35en0SAlY5LPP3DdMFD8ObG50",
  "authDomain": "iotca2-7591e.firebaseapp.com",
  "databaseURL": "https://iotca2-7591e.firebaseio.com",
  "storageBucket": "iotca2-7591e.appspot.com",
  "serviceAccount": "iotca2-7591e-firebase-adminsdk-e7sbb-19bff1dd8e.json"
}

pyrebaseDatabase = pyrebase.initialize_app(pyrebase_config)
db = pyrebaseDatabase.database()
duration = db.child("fan_status").child("duration").get().val()
threshold = db.child("fan_status").child("threshold").get().val()

#Configure MQTT
host = "a2g49f96gkk39l.iot.ap-southeast-1.amazonaws.com"
rootCApath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

def customCallback(client, userdata, message):
  global duration
  global threshold
  global count
```

```
  count = 0
  print("Received new message: ")
  print(message.payload)
  #pump_config = json.loads(message.payload)
  #duration = pump_config['duration']
  #threshold = pump_config['threshold']
  print("From topic:")
  print(message.topic)
  print("----------------------")

rpi = AWSIoTMQTTClient("temphum")
rpi.configureEndpoint(host, 8883)
rpi.configureCredentials(rootCApath, privateKeyPath, certificatePath)
rpi.configureOfflinePublishQueueing(-1)
rpi.configureDrainingFrequency(2)
rpi.configureConnectDisconnectTimeout(10)
rpi.configureMQTTOperationTimeout(5)
rpi.connect()
rpi.subscribe("sensors/temphumconfig", 1, customCallback)

# Configure GPIO
relay_pin = 21
dht11_pin = 19

# Main loop - read raw data and display
while True:
  global count
  GPIO.setmode(GPIO.BCM)
  GPIO.setup(relay_pin, GPIO.OUT)
  GPIO.output(relay_pin, GPIO.HIGH)
  print("Relay Pin is now set to HIGH")
  humidity, temperature = Adafruit_DHT.read_retry(11, dht11_pin)
  print(humidity)
  print(temperature)
  print ("Temperature: " + str(temperature) + " Humidity: " + str(humidity))

  if (int(humidity) > threshold):
     GPIO.output(relay_pin, GPIO.LOW)
     count += 1
     print("Relay is now turning on.")
     time.sleep(duration)
     print("Relay is now off.")
  else:
     GPIO.output(relay_pin, GPIO.LOW)
     count += 1

  data = { 'timestamp': int(time.time()), 'temperature': int(temperature), 'humidity': int(humidity)}
  # result = db.child("temperature_humidity_readings").push(data)
  json_string = json.dumps(data)
  print("Publishing to AWS MQTT Broker...")
  rpi.publish("sensors/temphum", json_string, 1)
  print("Done publishing to AWS MQTT Broker.")
  time.sleep(1)
  if (count == 1):
    GPIO.cleanup()
```

L. Write code for water.py

- Create a python script water.py
  - Sudo nano ~/CA2/ /water.py

```
import spidev
import time
from datetime import datetime
import os
import cloudinary
import cloudinary.uploader
import cloudinary.api
import Adafruit_DHT
import pyrebase
import logging
import json
import RPi.GPIO as GPIO

# Configure Cloudinary
cloudinary.config(
  cloud_name = "iotca2",
  api_key = "732475871262856",
  api_secret = "XbjpVSAgUc_PypzoipnXkHyPmG0"
)

# Configure Firebase
pyrebase_config = {
  "apiKey": " AIzaSyAlX8g6Ot35en0SAlY5LPP3DdMFD8ObG50",
  "authDomain": "iotca2-7591e.firebaseapp.com",
  "databaseURL": "https://iotca2-7591e.firebaseio.com",
  "storageBucket": "iotca2-7591e.appspot.com",
  "serviceAccount": "iotca2-7591e-firebase-adminsdk-e7sbb-19bff1dd8e.json"
}

pyrebaseDatabase = pyrebase.initialize_app(pyrebase_config)
db = pyrebaseDatabase.database()

# Configure GPIO
water_level_pin = 24

# Main loop - read raw data and display
while True:
   GPIO.setmode(GPIO.BCM)
   GPIO.setup(water_level_pin, GPIO.IN)
   water_level = GPIO.input(water_level_pin)
   print (water_level)
   time.sleep(0.1)
   GPIO.cleanup()
```

M. Write code for water_level_detection.py

- Create a python script water_level_detection.py
  - Sudo nano ~/CA2/ / water_level_detection.py

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import spidev
import time
from datetime import datetime
import os
import cloudinary
import cloudinary.uploader
import cloudinary.api
import Adafruit_DHT
import pyrebase
import logging
import json
import RPi.GPIO as GPIO
```

```
count = 0

# Configure Cloudinary
cloudinary.config(
  cloud_name = "iotca2",
  api_key = "732475871262856",
  api_secret = "XbjpVSAgUc_PypzoipnXkHyPmG0"
)

# Configure Firebase
pyrebase_config = {
  "apiKey": " AIzaSyAlX8g6Ot35en0SAlY5LPP3DdMFD8ObG50",
  "authDomain": "iotca2-7591e.firebaseapp.com",
  "databaseURL": "https://iotca2-7591e.firebaseio.com",
  "storageBucket": "iotca2-7591e.appspot.com",
  "serviceAccount": "iotca2-7591e-firebase-adminsdk-e7sbb-19bff1dd8e.json"
}

pyrebaseDatabase = pyrebase.initialize_app(pyrebase_config)
db = pyrebaseDatabase.database()

#Configure MQTT
host = "a2g49f96gkk39l.iot.ap-southeast-1.amazonaws.com"
rootCApath = "rootca.pem"
certificatePath = "certificate.pem.crt"
privateKeyPath = "private.pem.key"

rpi = AWSIoTMQTTClient("basicPubSub")
rpi.configureEndpoint(host, 8883)
rpi.configureCredentials(rootCApath, privateKeyPath, certificatePath)
rpi.configureOfflinePublishQueueing(-1)
rpi.configureDrainingFrequency(2)
rpi.configureConnectDisconnectTimeout(10)
rpi.configureMQTTOperationTimeout(5)
rpi.connect()

# Configure GPIO
water_level_pin = 25
buzzer_pin = 17

# Main loop - read raw data and display
while True:
   global count
   GPIO.setmode(GPIO.BCM)
   GPIO.setup(water_level_pin, GPIO.IN)
   GPIO.setup(buzzer_pin, GPIO.OUT)
   GPIO.output(buzzer_pin, GPIO.LOW)
   water_level = GPIO.input(water_level_pin)
   print (water_level)

   if (water_level == 0):
      GPIO.output(buzzer_pin, GPIO.HIGH)
      count += 1
      time.sleep(5)
   else:
      GPIO.output(buzzer_pin, GPIO.LOW)
      count += 1

   data = { 'timestamp': int(time.time()), 'water_level': water_level}
   # result = db.child("water_level").push(data)
   json_string = json.dumps(data)
   print("Publishing to AWS MQTT Broker...")
   rpi.publish("sensors/waterlevel", json_string, 1)
   print("Done publishing to AWS MQTT Broker.")
   time.sleep(5)
   if (count == 1):
     GPIO.cleanup()
```

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