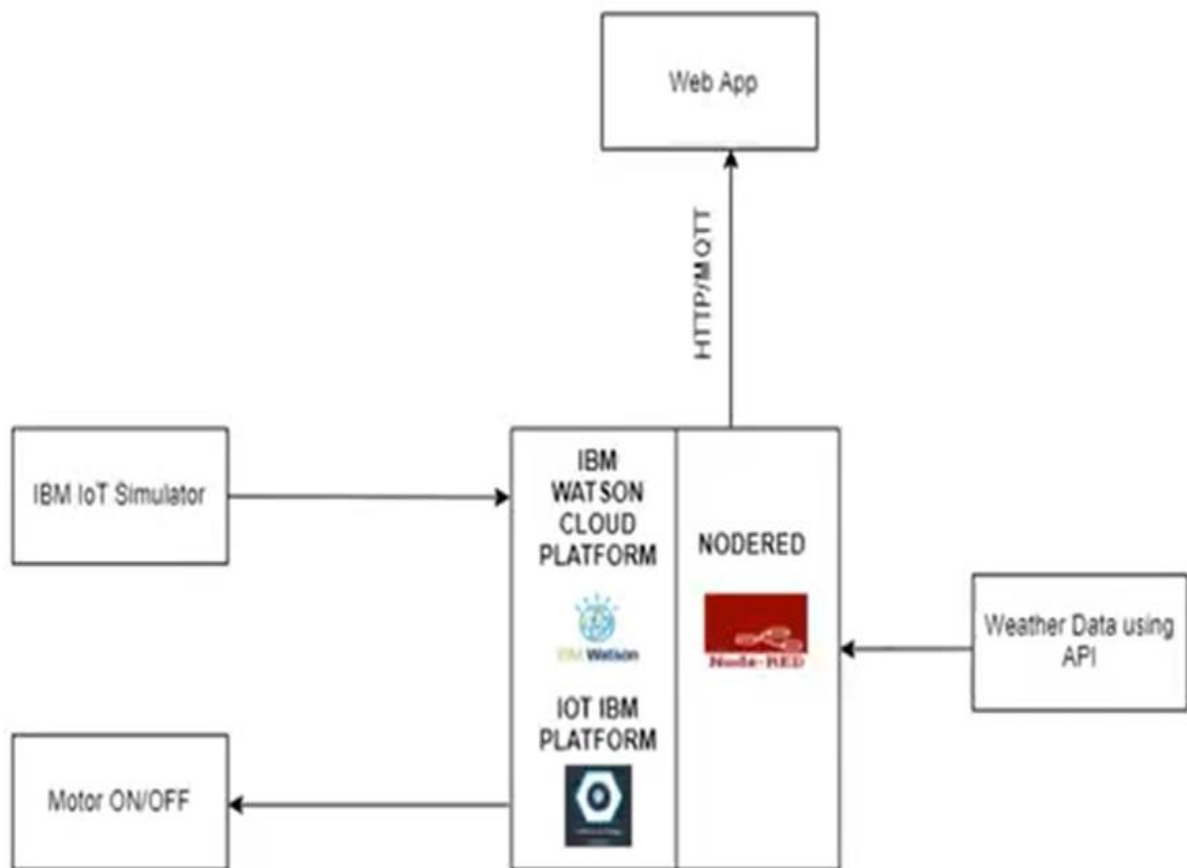


SMART AGRICULTURE SYSTEM BASED ON IOT

Aim and Scope:

Smart Agriculture System based on Iot can monitor soil moisture and climatic conditions to grow and yield a good crop. The farmer can also get the real-time weather forecasting data by using external platforms like Open Weather API. Farmer will be provided a mobile app using which he can monitor the temperature, humidity and soil moisture parameters along with weather forecasting details. Based on all the parameter, farmer can water his crop by controlling the motor using the mobile application. Thus even if the farmer is not present near his crop he can water his crop by controlling the motors using the application from anywhere.

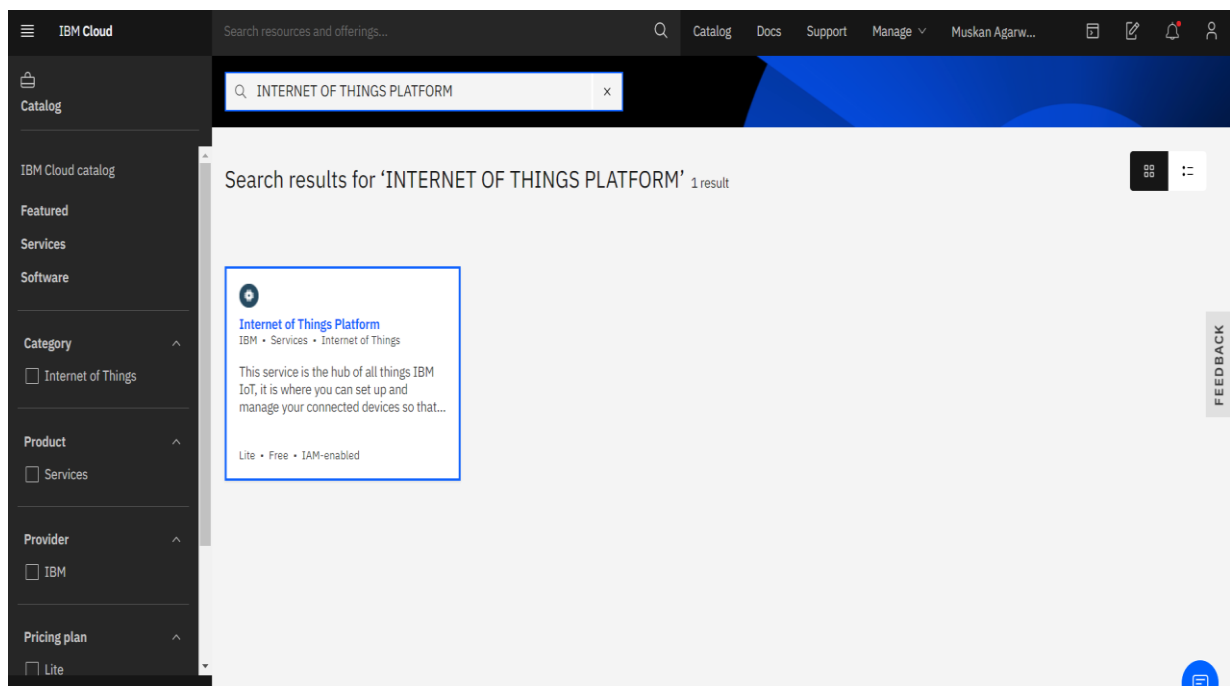
The Project Flow can be seen from the following diagram:



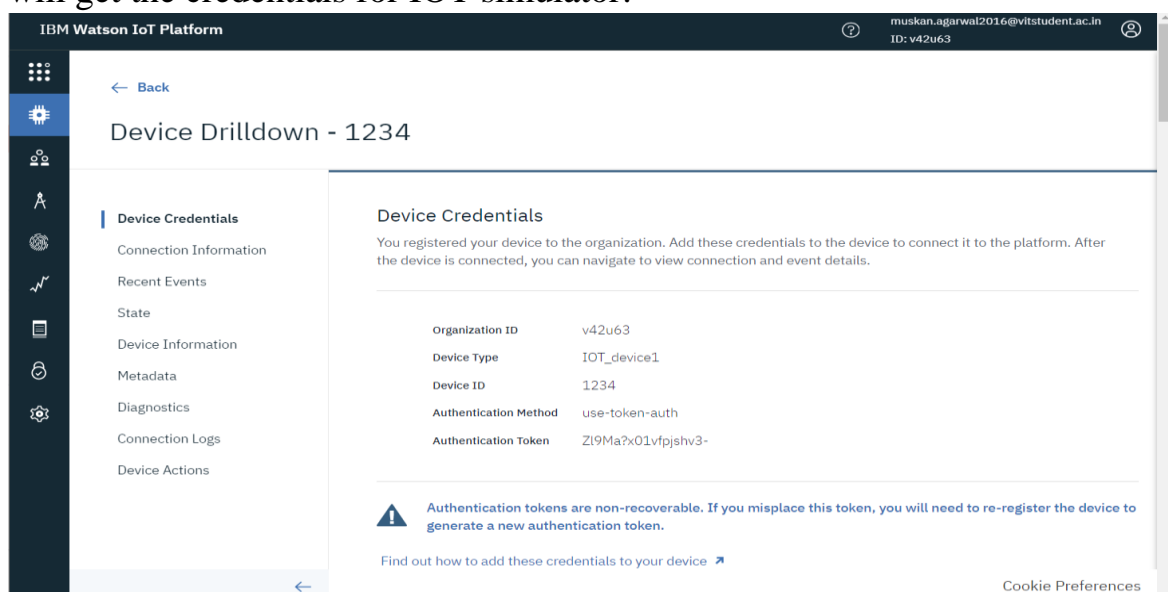
Getting Sensor Inputs into IBM Cloud:

First task to build the web app is to get the sensor data in the cloud. I am using IBM cloud for this. We need an IBM account for the same. Steps to take sensor data in the cloud:

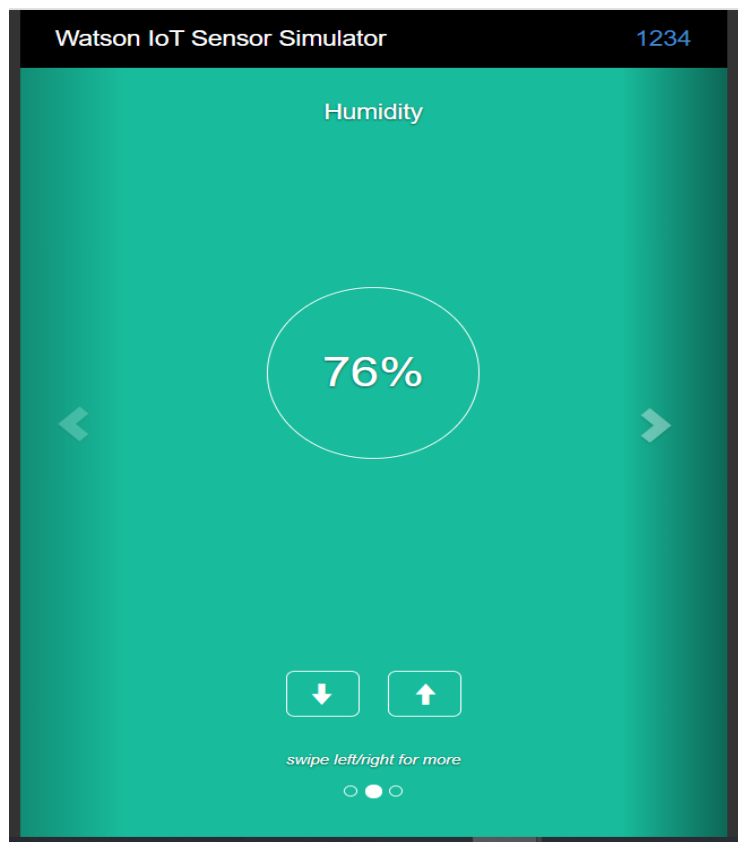
1. Sign up for IBM Academic Initiative Account using this [link](#).
2. After this Sign up for IBM Cloud using [link](#).
3. Go to IBM Watson IOT Platform by searching IOT platform in the catalogue in IBM Cloud



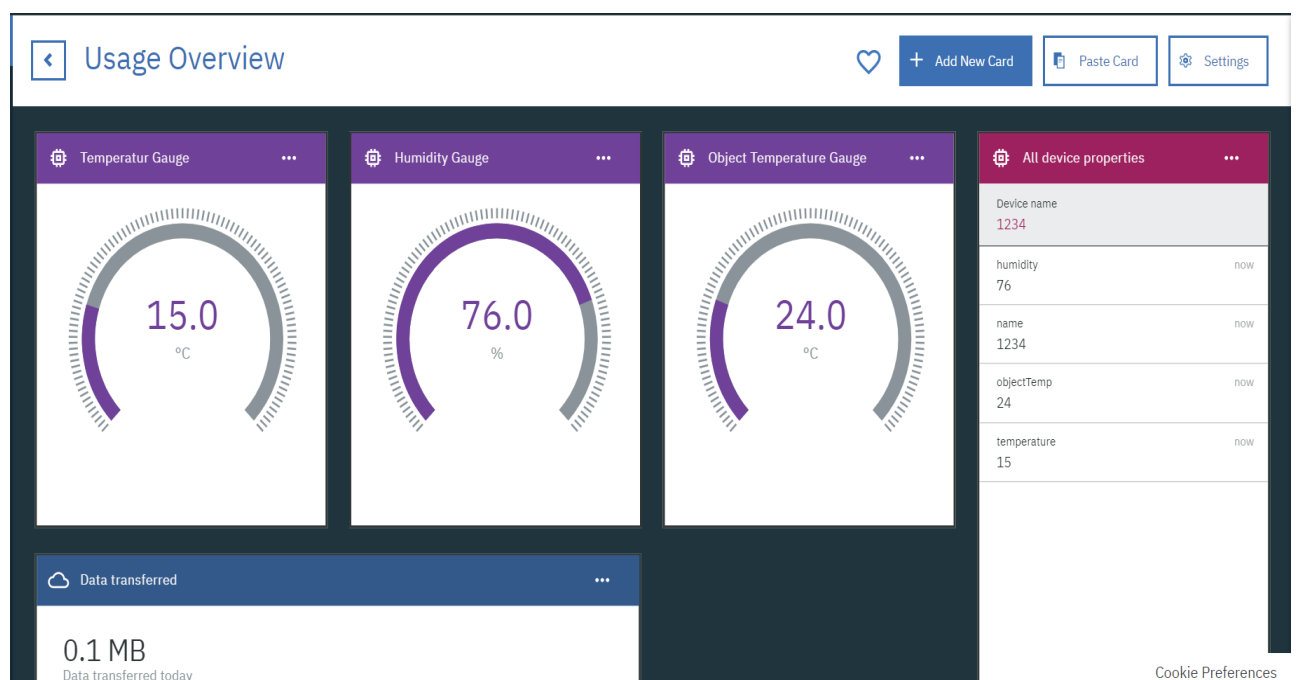
4. Go to the IOT Platform and now we will create a device here after which we will get the credentials for IOT simulator.



5. You will get Device credentials save them in a notepad so that we connect to IOT simulator. Go to [link](#) for IOT simulator. The following screen appears once your simulator get's connected.



6. Now in the cloud we can create cards to view the simulator data.



Node-Red

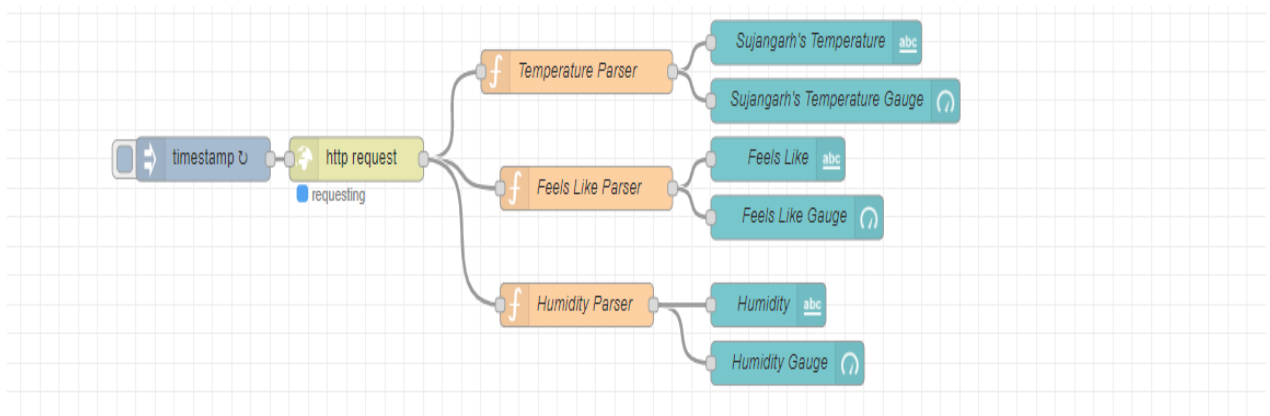
Now once we got the data in the cloud, we will use node red to get the data in a web app. To install node-red in windows follow this [link](#). After this we would need to externally install IBM iot node in node red using the below code.

Node-red-contrib-scx-ibmiotapp

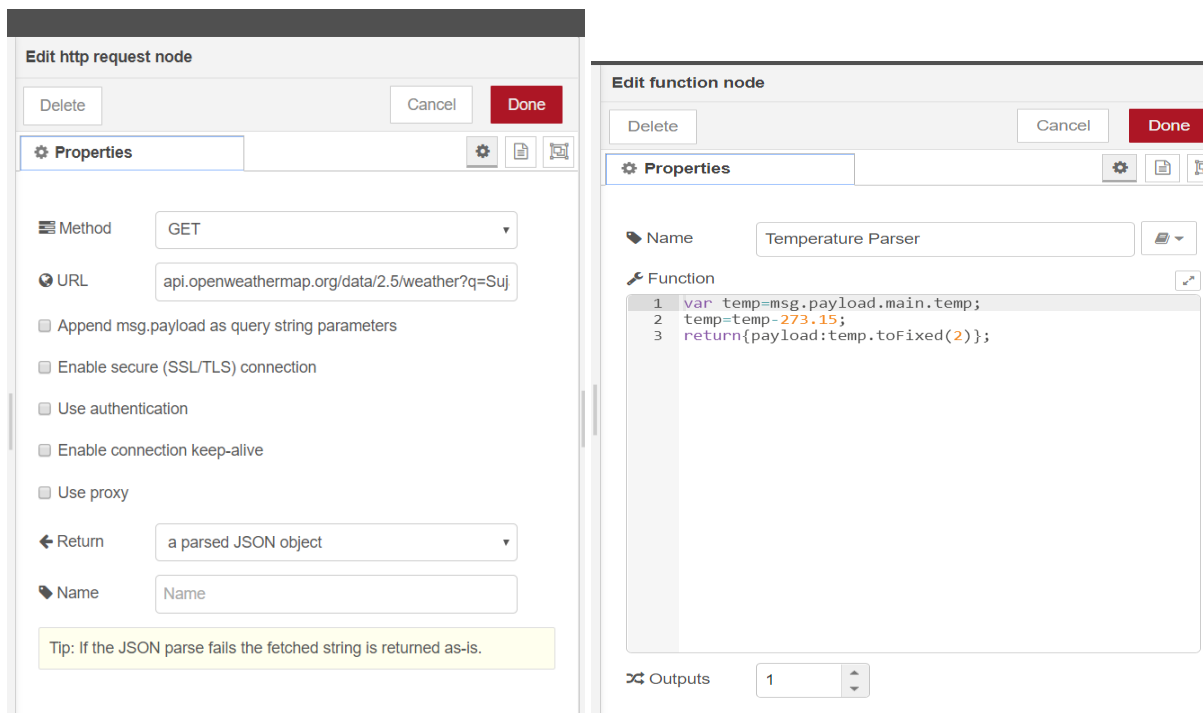
We would need 3 flows:

1. To take the weather data from OpenWeather API.
2. To take sensor data from the IBM cloud.
3. Finally, to transfer the motor control data to the cloud.

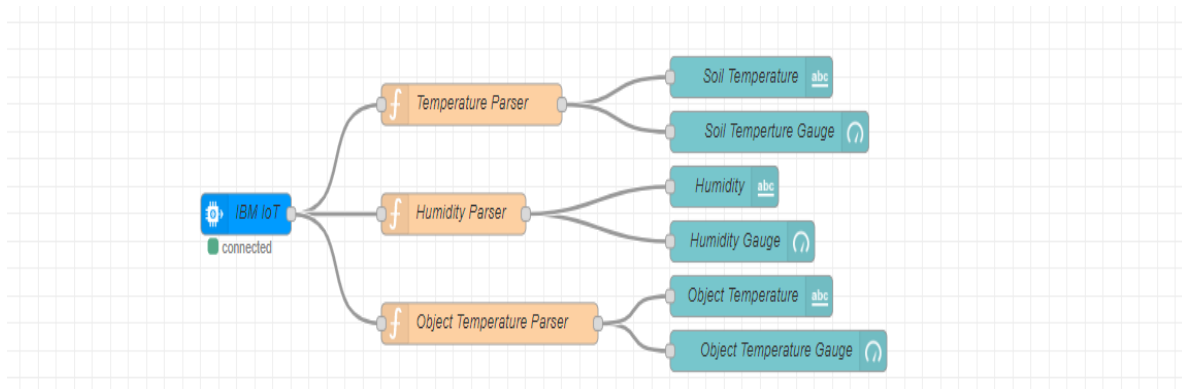
Flow1



In this the settings for the function nodes and http nodes are as follows:



Flow-2



Edit ibmiot in node

Delete

Cancel

Done

Properties

Authentication

API Key

API Key

f1733a44.3950f8

Input Type

Device Event

Device Type

All or Agriculture

Device Id

All or 123456789

Event

All or +

Format

All or json

QoS

0

Name

IBM IoT

Service

registered

Edit function node

Delete

Cancel

Done

Properties

Name

Temperature Parser

Function

1

msg.payload=msg.payload.d.temperature;

2

return msg;

Outputs

1

Now we need to put API in IBM IOT node which we can get form IBM cloud from Apps tab.

[Browse](#) IBM Cloud Apps

The API key has been added.

Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the API key to generate a new authentication token.

Generated Details

API Key a-v42u63-6bimomwbwa

Authentication Token QEikSOZ4IT#I2@L9

API Key Information

Description Smart Agriculture system

Role Standard Application

Expires Never



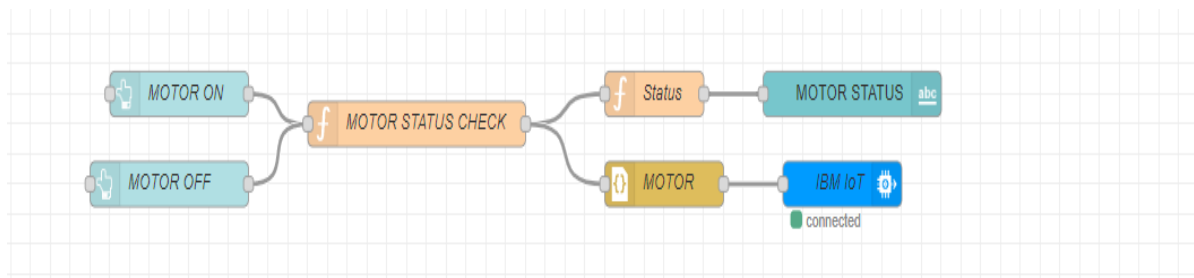
Make a note of the generated authentication token. Lost authentication tokens cannot be recovered. If you lose the token, you must reregister the API to generate a new token.

[View API Key](#)

[Add Another](#)

[Close](#)

Flow 3



Code for “Motor Status Check”

```
if(msg.payload === true)
    msg.payload = '{"command":"ON"}'
else
    msg.payload = '{"command":"OFF"}'
return msg;
```

Code for “status”

```
var data=JSON.parse(msg.payload);
msg.payload=data.command;
return msg;
```

Settings for “MOTOR” and Buttons

Edit json node

Delete Cancel Done

Properties

Action

Convert between JSON String & Object

Property

msg.payload

Name

MOTOR

Object to JSON options

☐ Format JSON string

Edit button node

Delete Cancel Done

Properties

Group

[Controls] Motor Control

Size

auto

Icon

optional icon

Label

MOTOR ON

Tooltip

optional tooltip

Colour

optional text/icon color

Background

optional background color

When clicked, send:

Payload

true

Topic

→ If msg arrives on input, emulate a button click:

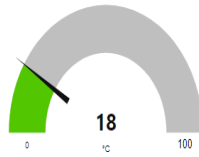
☐

Web APP

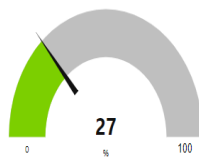
≡ Home

IBM Watson

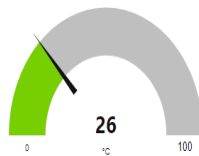
Soil Temperature



Humidity

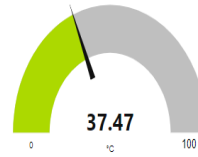


Object Temperature

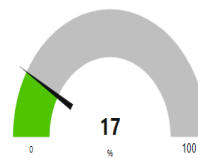


Open Weather API

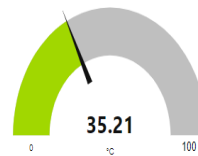
Sujangarh's Temperature



Humidity



Feels Like Gauge



≡ Controls

Motor Control

MOTOR ON

MOTOR OFF

MOTOR STATUS

OFF

Python program to receive commands from Watson IOT platform

```
import time
import sys
import ibmiotf.application # to install pip install ibmiotf
import ibmiotf.device

#Provide your IBM Watson Device Credentials
organization = "v42u63" #replace the ORG ID
deviceType = "Agriculture"#replace the Device type wi
deviceId = "123456789"#replace Device ID
authMethod = "token"
authToken = "123456789" #Replace the authtoken

def myCommandCallback(cmd): # function for Callback
    print("Command received: %s" % cmd.data)
    if cmd.data['command']=='ON':
        print("Motor ON IS RECEIVED")

    elif cmd.data['command']=='OFF':
        print("Motor OFF IS RECEIVED")

    if cmd.command == "setInterval":

        if 'interval' not in cmd.data:
            print("Error - command is missing required information: 'interval'")
        else:
            interval = cmd.data['interval']
    elif cmd.command == "print":
        if 'message' not in cmd.data:
            print("Error - command is missing required information: 'message'")
```



```

        else:

            output=cmd.data['message']

            print(output)

try:

    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-
method": authMethod, "auth-token": authToken}

    deviceCli = ibmiotf.device.Client(deviceOptions)

    #.....

except Exception as e:

    print("Caught exception connecting device: %s" % str(e))

    sys.exit()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type
"greeting" 10 times
deviceCli.connect()

while True:

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()

```

Final Output

```
17 def myCommandReceived(cmd): # function for callback
```

```
20     print("Command received: %s" % cmd.data)
21     if cmd.data['command']=='ON':
22         print("Motor ON IS RECEIVED")
23
24     elif cmd.data['command']=='OFF':
25         print("Motor OFF IS RECEIVED")
26
27     if cmd.command == "setInterval":
28
29         if 'interval' not in cmd.data:
30             print("Error - command is missing required information: 'interval'")
31         else:
32             interval = cmd.data['interval']
33     elif cmd.command == "print":
34         if 'message' not in cmd.data:
35             print("Error - command is missing required information: 'message'")
36     else:
37         output=cmd.data['message']
38         print(output)
39
40 .....
```

```
d:\v42u63\Agriculture\123456789
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
```