SRI BALAJI CHOCKALINGAM ENGINEERING COLLEGE – ARNI

DEPARTMENT OF ELECTRONICS AND COMMUNICARTION ENGINEERING

PROJECT REPORT

Smart Farmer – IOT Enabled Smart Farming Application

TEAM ID – PNT2022TMID40222

SUBMMITED BY

SI NO	STUDENT NAME	REGISTER NUMBER
1	M. SRIDHAR	512719106701
2	E. TAMIL SELVAN	512719106020
3	A. KIRUBHANITHI	512719106301
4	J. DEENATHAYAL	512719106004
5	SIVAKUMAR	512719106303

UNDER THE GUIDANCE OF FACULTY MENTOR

THIRU. Dr. K. VIJAYAKANTHAN

UNDER THE GUIDANCE OF INDUSTRIYAL MENTOR

THIRU. BHARATHWAJ

TABLE OF CONTENT

SI NO	CONTENT	PAGE NO
1	SYNOPSIS	3
2	INTRODUCTION	4
3	REQUIREMENT GATHERING	6
4	ARCHITECTURE DESIGN	7
5	MIT APP INVENTOR	8
6	APP INTERFACE	9
7	NODE RED	12
8	DESIGNING IN NODE RED FLOW EDITOR	13
9	DESIGNING OF WEB APPLICATION	14
10	DH11 TEMPERATURE AND HUMIDITY SENSOR	16
11	SOIL MOISTURE SENSOR	17
12	ELECTRIC GATE VALVE	18
13	DEVELOPING PYTHON SCRIPT	19
14	CONCLUTION	23
15	DEMO VIDEO LINK	24

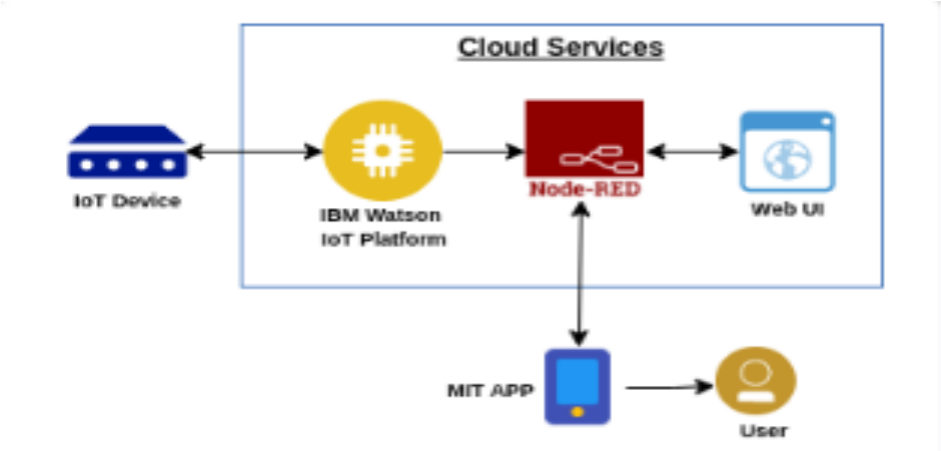
SYNOPSIS

Since a few years, smart devices have become an integral part of our daily lives. As a result, on smart devices, offering facilities and security is becoming more important. The goal of this Project is to create a **Smart Farming system** that works with Android mobile devices. IOT allows the mobile device and the system to communicate with one another through **Node MCU, IBM Cloud, Node RED application and Raspberry pi**. Current studies discusses the IoT-Based Smart Farming System. Any suitable device may load the mobile application and interact with the system. Commands to turn on/off electrical equipment such as Motor, Light, and Motorized ball valve, as well as setting timers, can be delivered simply and fast from mobile devices through a simple and pleasant GUI application that is straightforward to use for any average user. The system then acts and responds to these orders by performing the tasks specified in the commands and informing the user of the outcome. At any place in the world, the user may also view the result on an Android mobile application. As a result, developing a Smart Farming system to reduce workload for farmers that strives to create an advanced Smart Farmer system utilizing Wi-Fi technology, Cloud and Node RED as a solid option.

APP LOGO

INTRODUCTION:

Technological improvements and innovations are commonplace in today's society, and people's living standards are rising as well. People's lives have been inspired by mobile phones in recent years. In recent years, the mobile phone has emerged as the most important aspect of people's lives. Humans can conduct a range of jobs, both on and off the internet, with the help of these smart gadgets, like as make our homes and companies smarter and more elegant. We demonstrated a method for **connecting and controlling electric devices like Motor, Light, and Motorized ball valve using an Android mobile app and a Wi-Fi module with help of Node MCU, IBM Cloud, Node RED and Raspberry pi**.



To communicate the data given by the apps, the Wi-Fi transmitter employs radio waves. Wi-Fi data is converted into an electronic signals, which is then sent via the antenna; Wi-Fi is based on radios signals technology. This signal is sent to the Controller. The data is then manipulated and operations are performed by the Raspberry pi. This controllers may be connected to the Relaying terminals of different switches to transfer the current after producing the magnetic field. The system's reliability may be ensured by adding new appliances at any time. Smart Farming is particularly significant in today's world because of its ability to be used in a variety of locations with great accuracy, saving money and time by reducing human effort.

The primary goal of this technology is to automate the control of domestic appliances such as Motor, Light, and Motorized ball valve. This literature survey goes into great depth on Smartand security systems that use Raspberry pi and GSM, as well as how we can manage Farming applications using an Android application. When a person enters the home, the number of people entering the house is increased, and in Home Automation mode, appliances are switched on, but in Security Mode, the security light is tuned on along with the alert. On the LCD panel, the number of people who have entered the residence is also shown.

When the room becomes empty, i.e., the number of people in the room drops to zero, the appliances are shut off, making the system more energy efficient. Furthermore, a person may operate his home appliances using an android application on his phone, reducing the amount of manual labor required. Simultaneously, if someone enters when security mode is activated, an SMS would be sent to the home owner's phone, indicating the presence inside the house. The alarm may be turned off by sending a text message or using an Android app.

REQUIREMENT GATHERING

USER REQUIREMENTS:

- Mobile application to control:
    - Motor Pump.
    - Lights.
    - Gate Valves.
    - Light Traps.
- Mobile application to monitor:
    - Temperature.
    - Humidity.
    - Moisture.
- Web application to control and monitor as same as in mobile application.

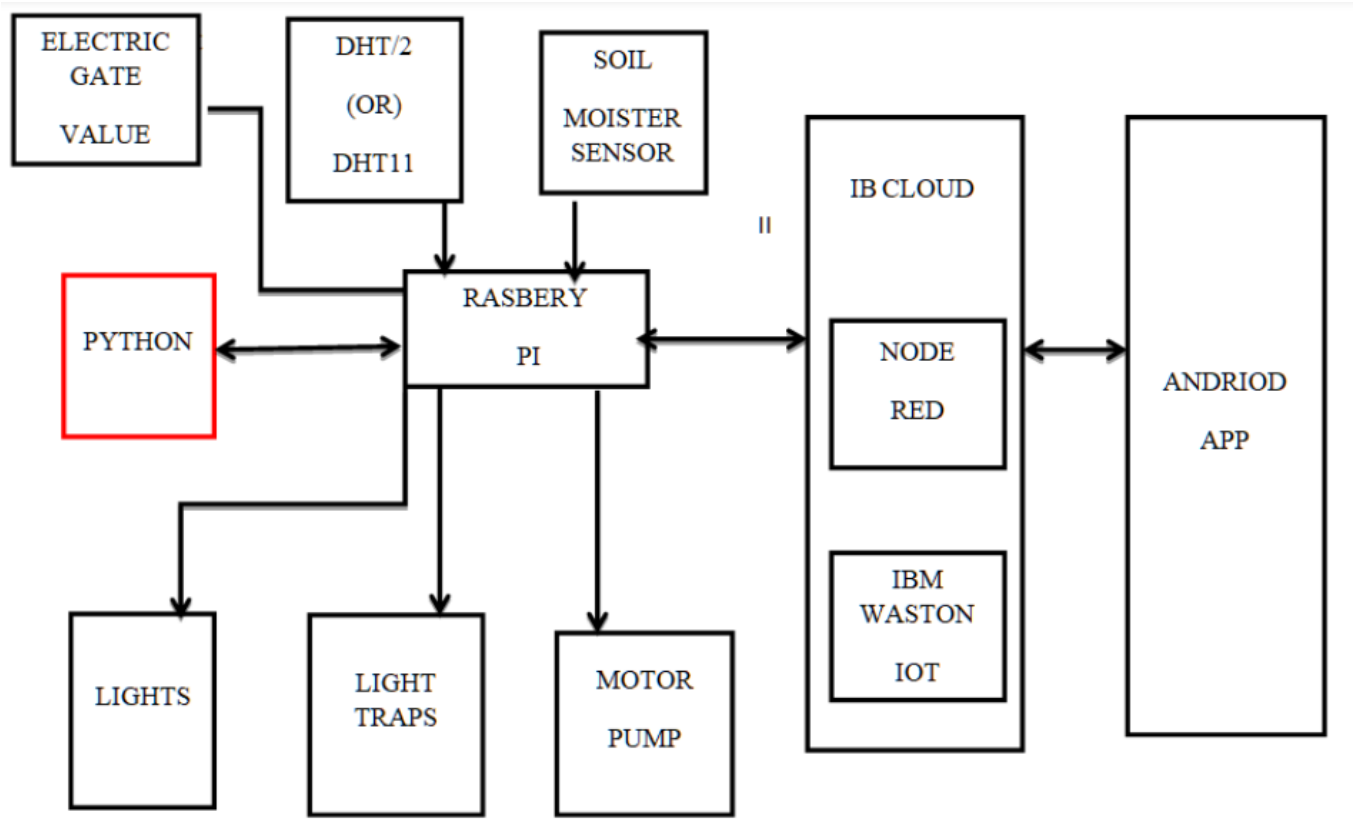
SYSTEM REQUIREMENT:

- MIT app inventor.
- Cloud Storage.
- Node RED.
- Watson IOT device.
- Node MCU.
- Sensors.
- Python Script.

SYSTEM AND SOFTWARE DESIGN

ARCHITECTURE DESIGN OF THE SYSTEM:

The architecture is a conceptual model that describes the structure and behavior of multiple components and subsystems like multiple software applications, network devices, hardware, and even other machinery of a system.



MIT APP INVENTOR

MIT App Inventor is an online platform designed to teach computational thinking concepts through development of mobile applications. Students create applications by dragging and dropping components into a design view and using a visual blocks language to program application behavior. App Inventor is a free, cloud-based service that allows you to make your own mobile apps using a **blocks-based programming language**.

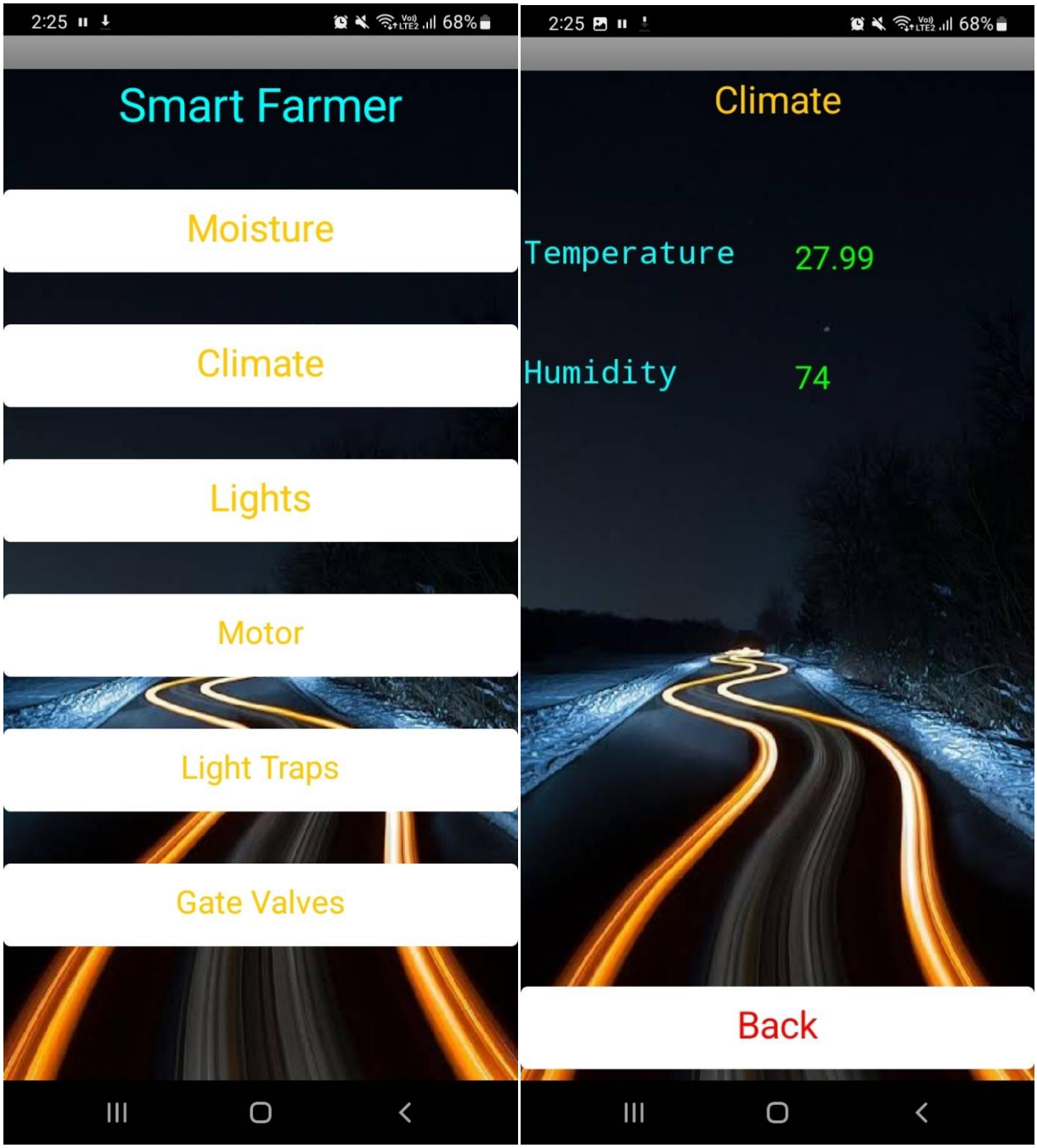
Benefits of MIT app inventor:

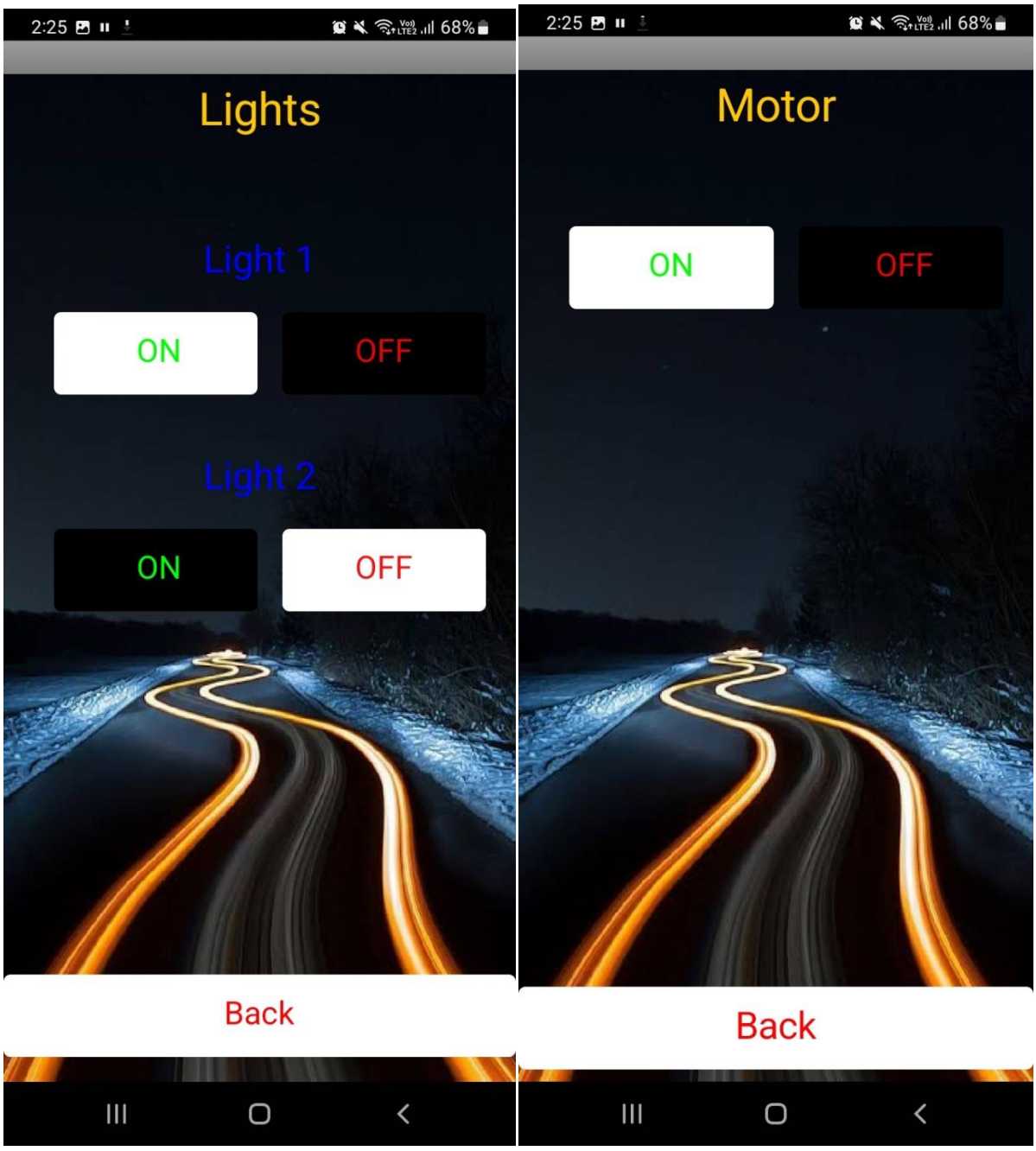
- Everything is done through a select and drop manner.
- Easy to test your app.
- MIT provides the user with some basic lessons which help in building that apps and that helps in a proper understanding of how the MIT app inventor platform works for the user.
- Useful for novices.

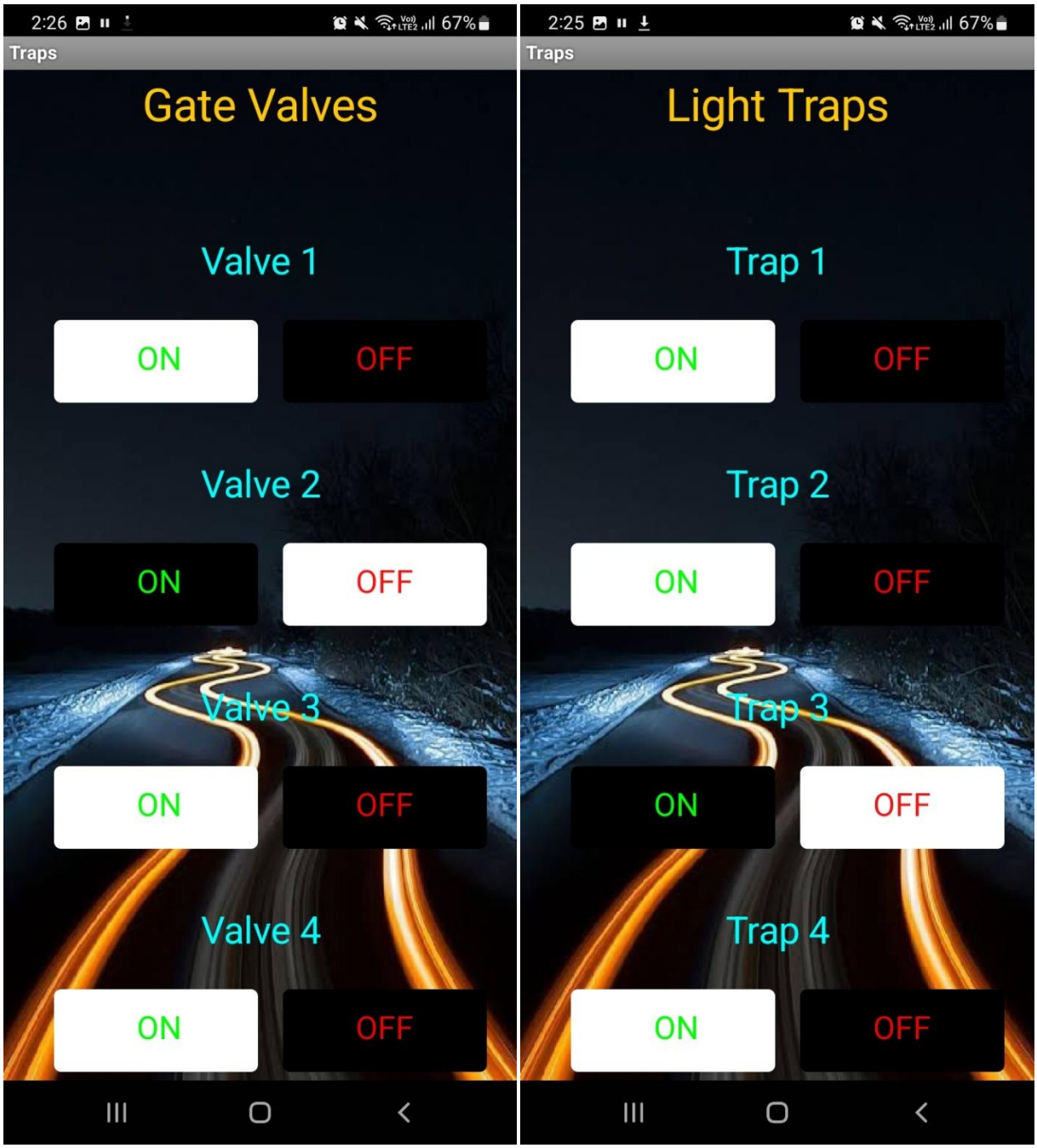
App Development in MIT app inventor:

APP LOGO

APP INTERFACE




2:25
68%
Lights
Light 1
ON
OFF
Light 2
ON
OFF
Back
2:25
68%
Motor
ON
OFF
Back


2:26
67%
Traps
Gate Valves
Valve 1
ON
OFF
Valve 2
ON
OFF
Valve 3
ON
OFF
Valve 4
ON
OFF
2:25
67%
Traps
Light Traps
Trap 1
ON
OFF
Trap 2
ON
OFF
Trap 3
ON
OFF
Trap 4
ON
OFF

NODE RED

Node-RED is **a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways**. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

Node-RED **allows you to create functionality by wiring together flows of data between nodes using a browser**. And it has gained tremendous popularity in the IoT space, by modeling bits of application functionality between IoT devices like sensors, cameras, and wireless routers.

The system stores the variables **in a JSON file in a folder called context under the . node-red folder**. Even though you are storing data in the file system it is still possible to loose data as the data is only flushed to the file system every 30 seconds.

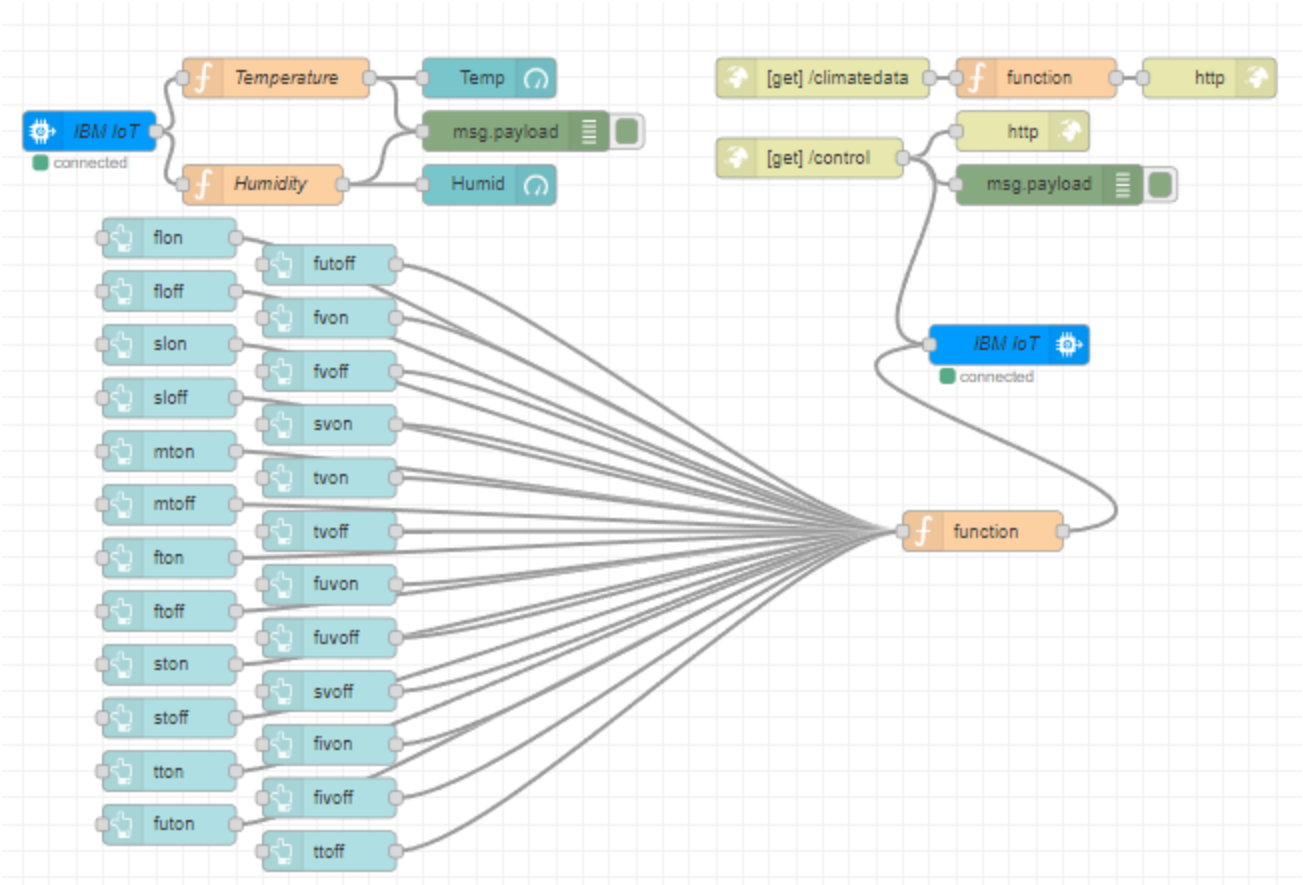
Node-RED includes built-in support for multiple communication protocols, e.g. **MQTT, HTTP, Raw TCP/IP**, and so on.

The function-npm node behaves like a normal function node, with the exception of allowing the use of npm modules within the script inside the block. It scans the script for any require statements and downloads the modules specified in the require statements.

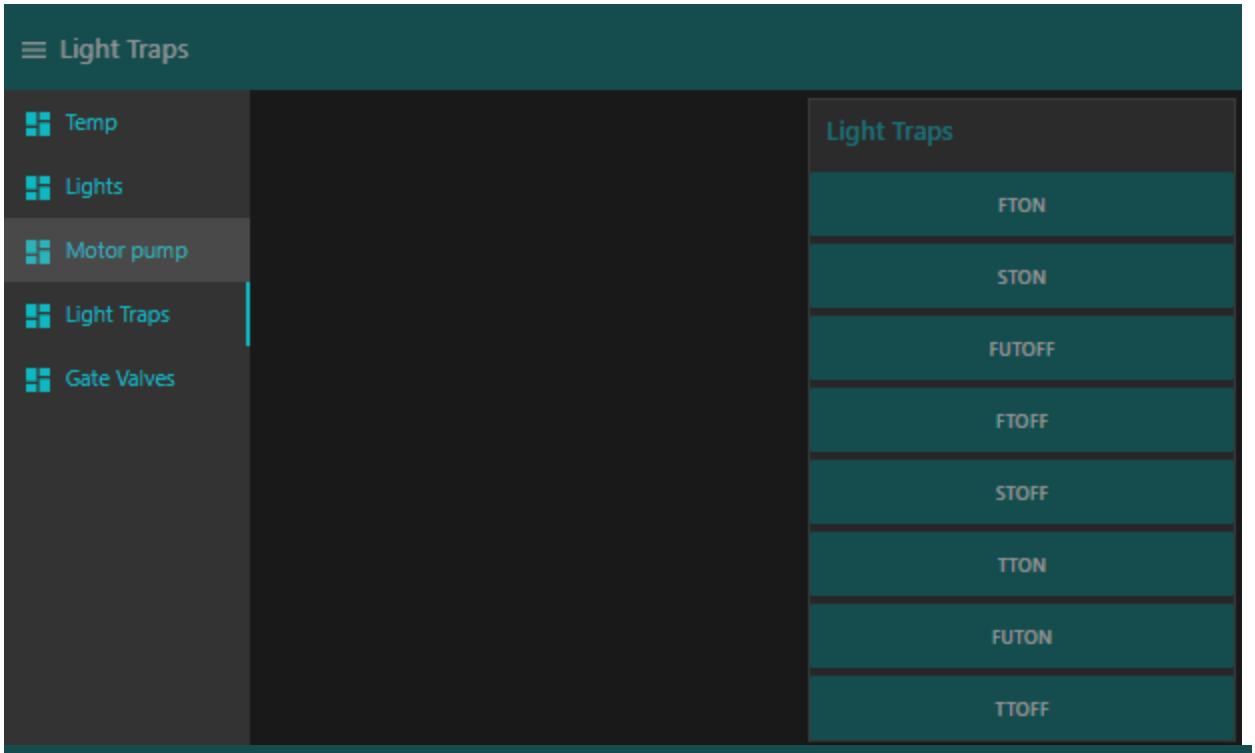
Key features of Node-RED:

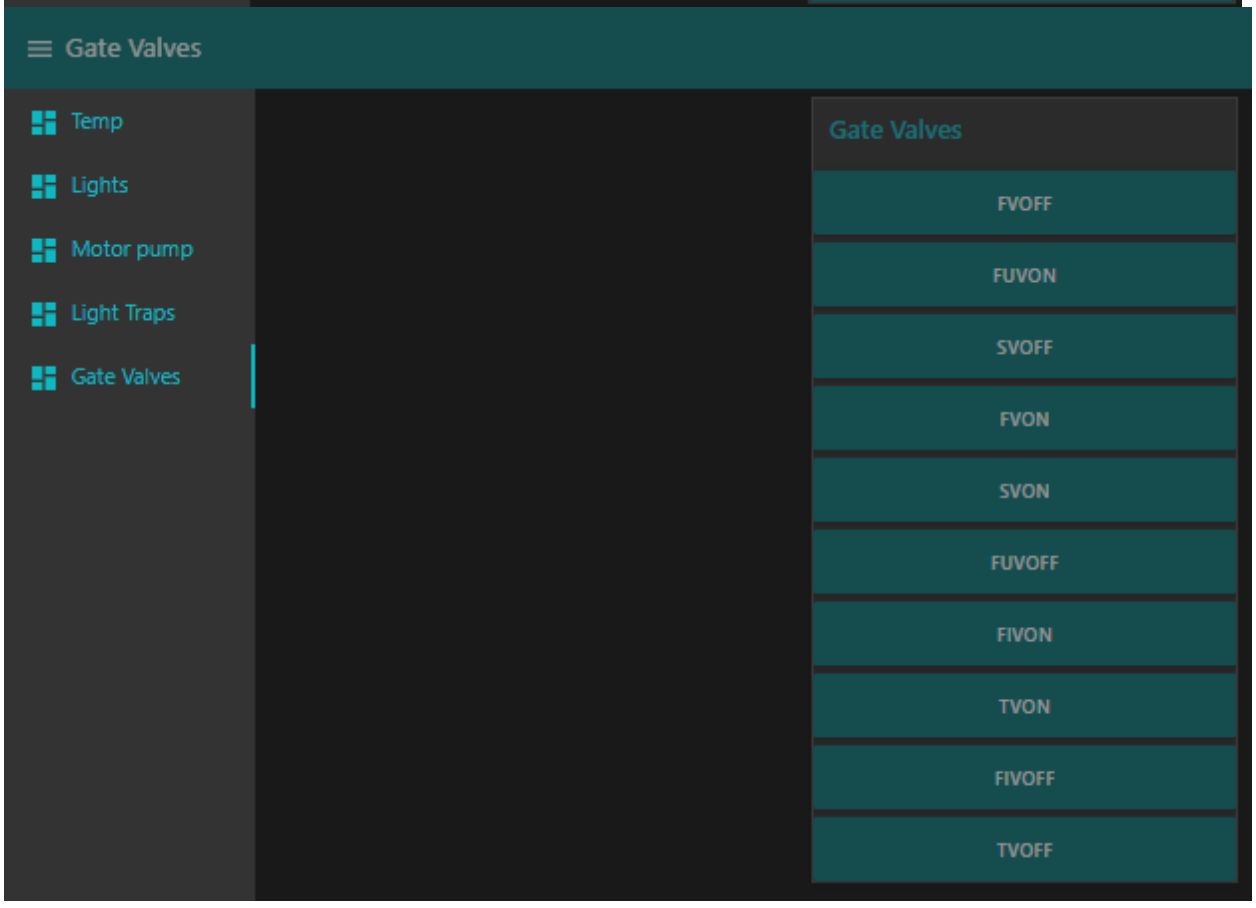
Supports browser-based flow editing making it user friendly, accessible and visual. It is built on Node. js, which is a none-blocking, lightweight I/O model, making it lightweight and efficient. Flows created in Node-RED are stored using JSON, and can imported and exported and shared with ease.

DESIGNING IN NODE RED FLOW EDITOR



DESIGNING THE WEB APPLICATION


Light Traps
Temp
Lights
Motor pump
Light Traps
Gate Valves
Light Traps
FTON
STON
FUTOFF
FTOFF
STOFF
TTON
FUTON
TTOFF

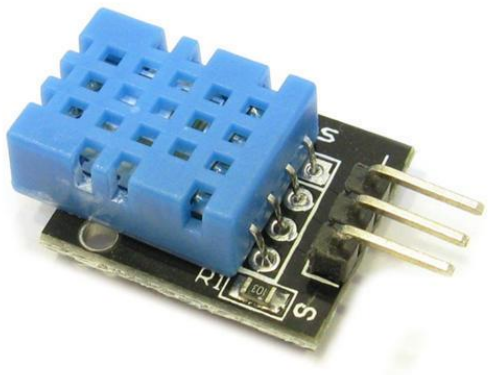

Gate Valves
Temp
Lights
Motor pump
Light Traps
Gate Valves
Gate Valves
FVOFF
FUVON
SVOFF
FVON
SVON
FUVOFF
FIVON
TVON
FIVOFF
TVOFF

SENSORS

DH11 TEMPERATURE AND HUMIDITY SENSOR:

The DHT11 is **a basic, ultra low-cost digital temperature and humidity sensor**. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). Its fairly simple to use, but requires careful timing to grab data.

DHT11 Temperature & Humidity Sensor features **a temperature & humidity sensor complex with a calibrated digital signal output**. By using the exclusive digital-signal-acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability.



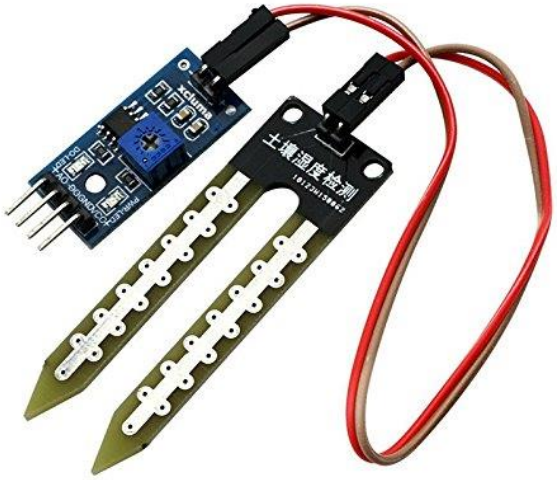
This sensor is used in various applications such as **measuring humidity and temperature values in heating, ventilation and air conditioning systems**. Weather stations also use these sensors to predict weather conditions. The humidity sensor is used as a preventive measure in homes where people are affected by humidity.

Its temperature measuring range is from -40 to +125 degrees Celsius with +-0.5 degrees accuracy, while the DHT11 temperature range is from **0 to 50 degrees Celsius with +-2 degrees accuracy**.

SOIL MOISTURE SENSOR:

Tensiometers are soil moisture sensors that measure this tension between soil particles and water molecules. In order for plants to access this water they must overcome the tension to draw water molecules away from the soil particles and into their roots.

Two sensors per site should be placed at about 1/4 and 3/4, or 1/3 and 2/3, of the root zone. With three sensors, **the shallowest sensor should be placed about 4 – 6 inches deep, then the next one at 1/2 – 2/3 of the root zone depth, and the last one towards the bottom of the root zone**.



Soil moisture monitoring **allows monitoring of what is happening in the soil root zone with regard to water infiltration during and after irrigations, and to water uptake by plants between irrigations**, thus enabling informed irrigation decisions on when to irrigate and how much water to apply to avoid crop water stress.

Place the sensor **near a tree's dripline on its Southwest side** (if planted in the Northern Hemisphere). Install the sensor at the tree canopy's dripline and 12 to 18 inches from an emitter, 24 to 36 inches from a micro-sprinkler, or in the wetted area of a conventional sprinkler.

ELECTRIC GATE VALVE:

Gate valves **incorporate a sliding gate to block fluid flow** (slide gate valves). The design of the valve operating and sealing systems typically requires that they should be operated either fully open or fully closed. They open by lifting a round or rectangular gate/wedge out of the path of the fluid.

Historically speaking, gate valve operations have been automated with multi-turn electric actuators. However, the preferred choice may be linear actuators with applications now requiring higher thrust, more precise positioning, faster stroke speed, or a specific failure safety position of the valve.

Every single electric gate motor comes with a manual release mechanism of some sort. Regardless of whether it is an underground, gate arm or sliding gate motor, they all come with a key to disengage the gates from the motor.

Automatic gates commonly **use electric motors or hydraulics to operate them**. Swing gates rotate around hinges on gate posts at the sides of driveway entrances. Sliding gates are moved by a static motor. They typically have backup batteries, remote controls, other settings and can be solar powered too.

DEVELOPING PYTHON SCRIPT

This python script is developed to publish data in IBM cloud and also receive data from the cloud.

CODE:

```
import wiotp.sdk.device

import time

import random

import requests

import json

myConfig = {

    "identity": {

        "orgId": "--------",

        "typeId": "---------",

        "deviceId":"--------"

    },

    "auth": {

        "token": "----------"

    }

}

def myCommandCallback(cmd):

  if(cmd.data['cm']=="flon"):
```

```
  print("-----Light 1 is ON-----")
if(cmd.data['cm']=="floff"):
  print("-----Light 1 is OFF-----")
if(cmd.data['cm']=="mton"):
  print("-----Motor is ON-----")
if(cmd.data['cm']=="mtoff"):
  print("-----Motor is OFF-----")
if(cmd.data['cm']=="slon"):
  print("-----Light 2 is ON-----")
if(cmd.data['cm']=="sloff"):
  print("-----Light 2 is OFF-----")
if(cmd.data['cm']=="fton"):
  print("-----Trap 1 is ON-----")
if(cmd.data['cm']=="ftoff"):
  print("-----Trap 1 is OFF-----")
if(cmd.data['cm']=="ston"):
  print("-----Trap 2 is ON-----")
if(cmd.data['cm']=="stoff"):
  print("-----Trap 2 is OFF-----")
if(cmd.data['cm']=="tton"):
  print("-----Trap 3 is ON-----")
```

```
if(cmd.data['cm']=="ttoff"):
  print("-----Trap 3 is OFF-----")
if(cmd.data['cm']=="futon"):
  print("-----Light 4 is ON-----")
if(cmd.data['cm']=="futoff"):
  print("-----Trap 4 is OFF-----")
if(cmd.data['cm']=="fvon"):
  print("-----Gate Valve 1 is ON-----")
if(cmd.data['cm']=="fvoff"):
  print("-----Gate Valve 1 is OFF-----")
if(cmd.data['cm']=="svon"):
  print("-----Gate Valve 2 is ON-----")
if(cmd.data['cm']=="svoff"):
  print("-----Gate Valve 2 is OFF-----")
if(cmd.data['cm']=="tvon"):
  print("-----Gate Valve 3 is ON-----")
if(cmd.data['cm']=="tvoff"):
  print("-----Gate Valve 3 is OFF-----")
if(cmd.data['cm']=="fuvon"):
  print("-----Gate Valve 4 is ON-----")
if(cmd.data['cm']=="fuvoff"):
```

```
    print("-----Gate Valve 4 is OFF-----")
  if(cmd.data['cm']=="fivon"):
    print("-----Gate Valve 5 is ON-----")
  if(cmd.data['cm']=="fivoff"):
    print("-----Gate Valve 5 is OFF-----")
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()
while(True):
weatherdata=requests.get('https://api.openweathermap.org/data/2.5/weather?q=Chennai&appid=3942dd61b26410206c466575e5addfbf')
  a=weatherdata.text
  b=json.loads(a)
  c=(b["main"]["temp"]-273.15)
  d=(b["main"]["humidity"])
  myData={'temp':c, 'humid':d}
  client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
  print("Published data Successfully: %s", myData)
  client.commandCallback = myCommandCallback
  time.sleep(2)
  print()
client.disconnect()
```

CONCLUTION

Through the experience gained by this project, we can now say that a lot of work has been donein this field, and still there is a lot more to be done. For various causes well understood like environmental concerns, deteriorating water reservoirs, and the need to conserve water for the future those areas must be explored and developed. Steps should be taken to achieve mentioned targets and analyze the present situation.

Here we have shown a small step to do the same and mentioned the futuristic improvements possible. This is a demonstration of what the pace of time requires and if we as students can do abit along that way then a lot more can be expected from the industries and thinkers. With more contribution and emphasis on research and development by the industry, this seems to be a possible endeavor in near future.

DEMO VIDEO LINK

VIDEO LINK - https://youtu.be/-63cYjrPIyo