

PROJECT REPORT

IOT BASED SMART CROP PROTECTION SYSTEM

FOR AGRICULTURE

TEAM ID: PNT2022TMID21131

ROLL NUMBER	NAME
714019205002	ABIRAMI PAVISYA S
714019205021	JEEVIKAA SHRI G N
714019205030	NARMATA CR
714019205042	SNEHA M

CONTENT

1. Introduction
2. Problem Statement
3. Proposed Solution
4. Theoretical Analysis
 - 4.1 System Architecture
 - 4.2 Software Requirements
 - 4.21 Node-Red Services
 - 4.22 IBM Watson IOT Platform
 - 4.23 Python Ide
 - 4.24 IBM Cloud ant
 - 4.3 MIT-App Simulator
 - 4.4 Open Weather API and Wowki platform
5. Implementation Of System
 - 5.1 IBM Watson IOT Device Deployment
 - 5.2 Deployment Of Node-Red Services
 - 5.3 Node-Red Connectivity to Python
 - 5.4 Deployment Of MIT-App Simulator
 - 5.5 Deployment Of Wowki platform
6. Observations And Results
7. Advantages And Disadvantages
8. Conclusion
9. Reference
10. Appendix

ABSTRACT

IOT Based Smart Crop-Protection for Agriculture monitoring is a system describes how to monitor crop field. It is developed by using sensors and according to the decision from a server based on sensed data, the irrigation and monitoring system is enhanced. Through wireless transmission the sensed data is forwarded to web server database. If the irrigation is automated, then the moisture and temperature fields are decreased below the potential range. The user can monitor and control the system remotely with the help of application which provides a web interface to user. By smart Agriculture monitoring system and one of the oldest ways in agriculture is the manual method of checking the parameters. In this method farmers by themselves verify all the parameter and calculate the reading. It aims at making agriculture smart using automation and IoT. The cloud computing devices are used at the end of the system that can create a whole computingsystem from sensors to tools that observe data from agriculture field. It proposes a novel methodology for smart farming by including a smart sensing system and smart irrigator system through wireless communication technology. This system is cheap at cost for installation. Here one can access and control the agriculture system in laptop, cell phone or a computer.

Chapter – 1

INTRODUCTION

A system using sensors that monitor different conditions of environment like humidity, temperature etc., the processor and GUI module is used. The field condition is sent to the farmer via mobile text messages. With this system Soil moisture, humidity and energy efficiency are managed. A system is proposed for intelligent agriculture monitoring system based on IOT technology. The main aim of this project is to help farmers to automate their farms by providing them with a Web App through which they can monitor the parameters of the field like Temperature, humidity etc. and control the equipment like water motor and other devices remotely via internet without their actual presence in the field.

Chapter-2

PROBLEM STATEMENT

- Farmers are to be present at farm for maintenance irrespective of the weather conditions.
- They must ensure that the crops are well irrigated, and the routine activities of the field must be monitored by them physically.
- To get good, cultivate or yield farmers need to stay in the field for longer time for good yield
- Demand and supply are more if the field is to be monitored monotonously if it covers vast area
- Anytime crops may prone to various calamities which leads to poor yield of the crops

And these are some cons that is been faced by the farmers and which leads them to cultivate crops for less yield despite heavy work

Chapter-3

PROPOSED SOLUTION

To reduce the cost-effective work, expenses and to work in an effective way, we introduce IoT services in which we use cloud services and internet to enable farmer to continue their work remotely via through internet. And they can also be able to monitor the field parameters and control the devices in farm to harvest higher yield with less consumption of manpower. The aim of the project is to protect fields to take account temperature and humidity of the fields to reduce the farming expenses and to save energy. Overall, the design realizes remote intelligent monitoring and control of fields and replaces the traditional wired technology to wireless, also reduces manpower cost.

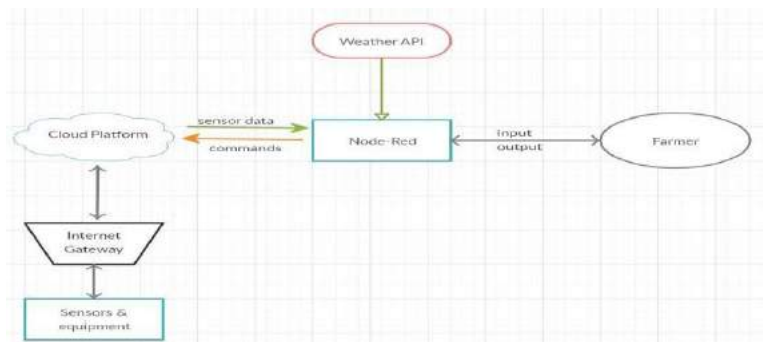
Chapter-4

THEORETICAL ANALYSIS

In this theoretical analysis we can be able to get detailed overview about the project and its implementation, deployment of the software used in each phase of the project

4.1 SYSTEM ARCHITECTURE

It depicts the flow of the project from initiation to the final phase of the project.



4.2 SOFTWARE REQUIREMENTS

The web-based application is deployed by consolidating the essential requirements to build IOT web-based platform. The following credential are needed to deploy the web application using NODE-RED which are stated as below

4.21 NODE-RED SERVICES

Node-Red is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs, and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions.

Steps involved in configuring the Node-Red Services

1. Create an IBM cloud account
2. Select catalog → Node-Red Application
3. Give the required credentials to access the Node-Red like cloud foundry, Resource list and so on
4. Once you have finished the process just click on the URL to deploy the Node-Red dashboard

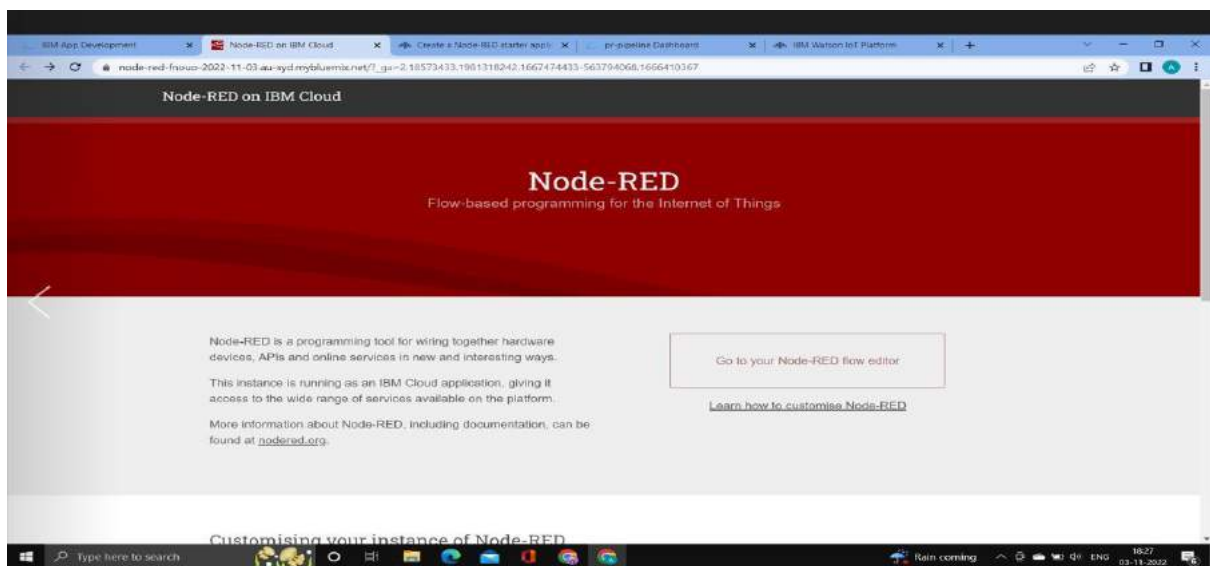


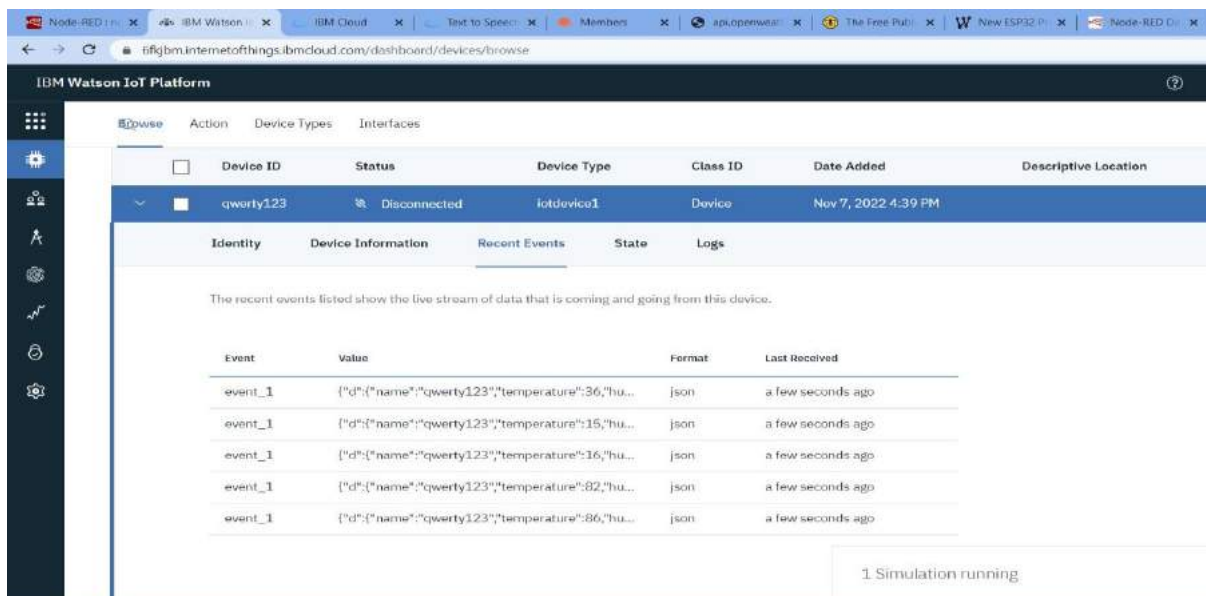
FIG-1 NODE-RED DASHBOARD

4.22 IBM Watson IOT Platform

A fully managed, cloud-hosted service with capabilities for device registration, connectivity, control, rapid visualization, and data storage. IBM Watson IoT Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT devices.

Steps involved in configuring the IBM-Watson

1. Create an IBM cloud account
2. Select catalog → IBM-Watson IOT
3. Give the required credentials to access the IBM-Watson like cloud foundry, Resource list and so on
4. Once you have finished the process just click on the LAUNCH to deploy the IBM Watson and Sign into it
5. Under the IBM Watson click on “Bluemix free” and you will direct to the IBM Watson-IOT where you can deploy your device



The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. The main content area shows a table of devices. The first device listed is 'qwerty123', which is 'Disconnected' and has a 'Device Type' of 'iotdevice1'. Below the table, there is a section for 'Recent Events' with a sub-header 'Identity'. The events table shows five entries, each with an 'Event' name, a 'Value' (JSON string), a 'Format' of 'json', and a 'Last Received' time of 'a few seconds ago'. The values represent temperature data for the device 'qwerty123'. At the bottom right, a status box indicates '1 Simulation running'.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
qwerty123	Disconnected	iotdevice1	Device	Nov 7, 2022 4:39 PM	

Event	Value	Format	Last Received
event_1	{"d":{"name":"qwerty123","temperature":36,"hu..."	json	a few seconds ago
event_1	{"d":{"name":"qwerty123","temperature":15,"hu..."	json	a few seconds ago
event_1	{"d":{"name":"qwerty123","temperature":16,"hu..."	json	a few seconds ago
event_1	{"d":{"name":"qwerty123","temperature":82,"hu..."	json	a few seconds ago
event_1	{"d":{"name":"qwerty123","temperature":86,"hu..."	json	a few seconds ago

1 Simulation running

FIG-2 DEVICE SIMULATION

4.23 Python Ide

Deployment of IOT platform is also initiated by the python interpreter that enables IOT platform to connect with devices. Few packages need to be installed to work in python interpreter so that it can be able to traverse between simulator and Node-Red many other services

```

C:\Users\HP>pip install ibm-cos-sdk
Collecting ibm-cos-sdk
  Using cached ibm-cos-sdk-2.12.0.tar.gz (55 kB)
  Preparing metadata (setup.py) ... done
Collecting ibm-cos-sdk-core==2.12.0
  Downloading ibm-cos-sdk-core-2.12.0.tar.gz (956 kB)
    ----- 956.0/956.0 kB 44.0 kB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting ibm-cos-sdk-s3transfer==2.12.0
  Downloading ibm-cos-sdk-s3transfer-2.12.0.tar.gz (135 kB)
    ----- 135.9/135.7 kB 47.0 kB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting jmespath<1.0.0, >=0.10.0
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting python-dateutil<1.0.0, >=2.8.2
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    ----- 247.7/247.7 kB 65.0 kB/s eta 0:00:00
Collecting requests<3.0, >=2.27.1
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
    ----- 62.8/62.8 kB 35.1 kB/s eta 0:00:00
Collecting urllib3<1.27, >=1.26.9
  Downloading urllib3-1.26.12-py2.py3-none-any.whl (140 kB)
    ----- 140.6/140.4 kB 54.1 kB/s eta 0:00:00
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting charset-normalizer<3, >=2
  Downloading charset-normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting certifi<2022.9.24, >=2017.4.17
  Downloading certifi-2022.9.24-py3-none-any.whl (161 kB)
    ----- 161.1/161.1 kB 94.7 kB/s eta 0:00:00
Collecting idna<4, >=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
    ----- 61.5/61.5 kB 84.2 kB/s eta 0:00:00
Building wheels for collected packages: ibm-cos-sdk, ibm-cos-sdk-core, ibm-cos-sdk-s3transfer
  Building wheel for ibm-cos-sdk (setup.py) ... done
  Created wheel for ibm-cos-sdk: filename=ibm_cos_sdk-2.12.0-py3-none-any.whl size=73910 sha256=9c7e17265f1b
  Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\53\89\fb\elf16b3874af57e2233ee18a438040b68
  Building wheel for ibm-cos-sdk-core (setup.py) ... done
  Created wheel for ibm-cos-sdk-core: filename=ibm_cos_sdk_core-2.12.0-py3-none-any.whl size=562947 sha256=
  Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\72\24\c4\b930b91571fe120c206726f285a26c93
  Building wheel for ibm-cos-sdk-s3transfer (setup.py) ... done
  Created wheel for ibm-cos-sdk-s3transfer: filename=ibm_cos_sdk_s3transfer-2.12.0-py3-none-any.whl size=897
  Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\3f\b7\b6\4c18868ed276686342eab42be15f0c0ac
Successfully built ibm-cos-sdk ibm-cos-sdk-core ibm-cos-sdk-s3transfer
Installing collected packages: urllib3, six, jmespath, idna, charset-normalizer, certifi, requests, python-
Successfully installed certifi-2022.9.24 charset-normalizer-2.1.1 ibm-cos-sdk-2.12.0 ibm-cos-sdk-core-2.12.0
urllib3-1.26.12
C:\Users\HP>

```

FIG-3 PYTHON COMMAND PROMPT

4.24 IBM Cloud ant

To store the NODE-RED value or any service provider we need to have a Database here the cloud ant database is created to store the values of the sensor reading of the devices

Name	Size	# of Docs	Partitioned	Actions
noderodino2021103	36.7 KB	4	No	[Icons for database actions]
sample	0 bytes	0	Yes	[Icons for database actions]
sample1	0.9 MB	4076	No	[Icons for database actions]

FIG-4 CLOUDANT DB

4.3 MIT-APP SIMULATOR

Simulation is carried on by using MIT-App inventor for Node-Red web application we have created a simulator with three modules they are:

- HOME SCREEN
- CONTROLLER SCREEN
- DATA SCREEN

There will be a designer and block panel separately and features and styles are being done in designer panel where we use to drag and drop desired icons.

Similarly block panel here we give functionality to the features which are done by drag and drop method

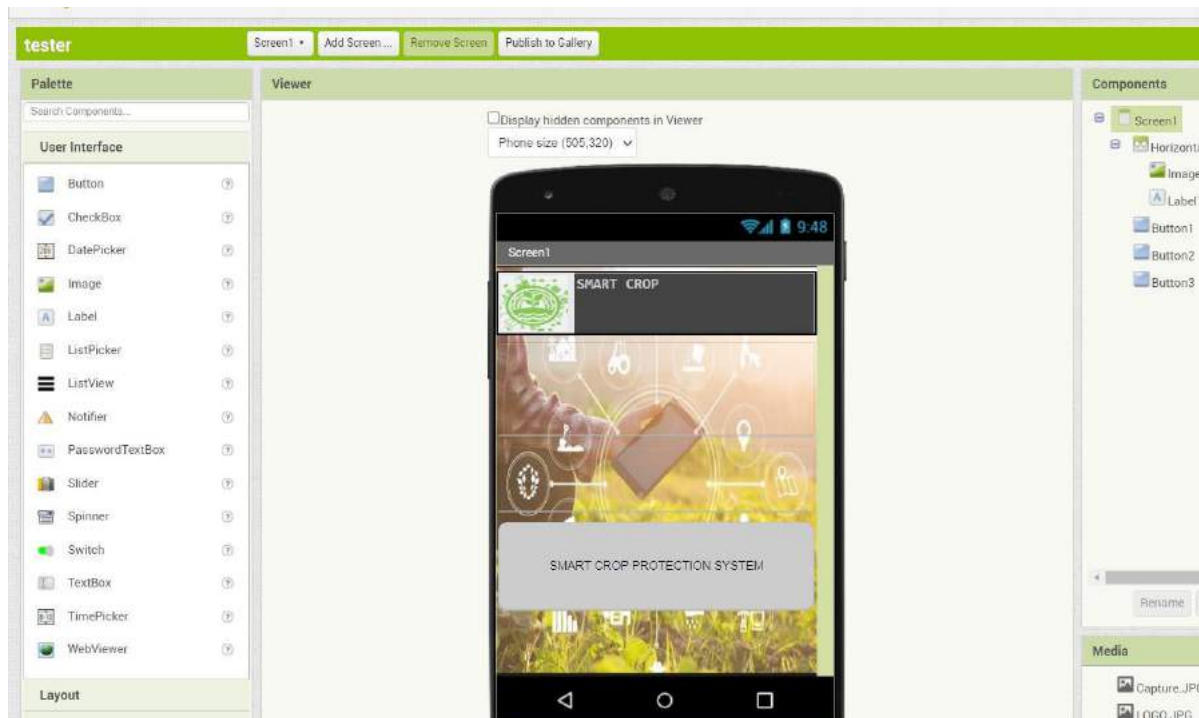


FIG-5 MIT-APP SIMULATOR

4.4 Open Weather API And Wowki platform

It is an online service that provides weather data. And to configure we need to create account in Open Weather and find the name of our city by searching then create API key to our account after that replace “city name” and “API key” with your city and API key. And for external hardware visualizer we deploy the Wowki platform where ESP22, DHT22 is connected to Node-red via Watson IOT platform so we can get reading both weather API and hardware appliance

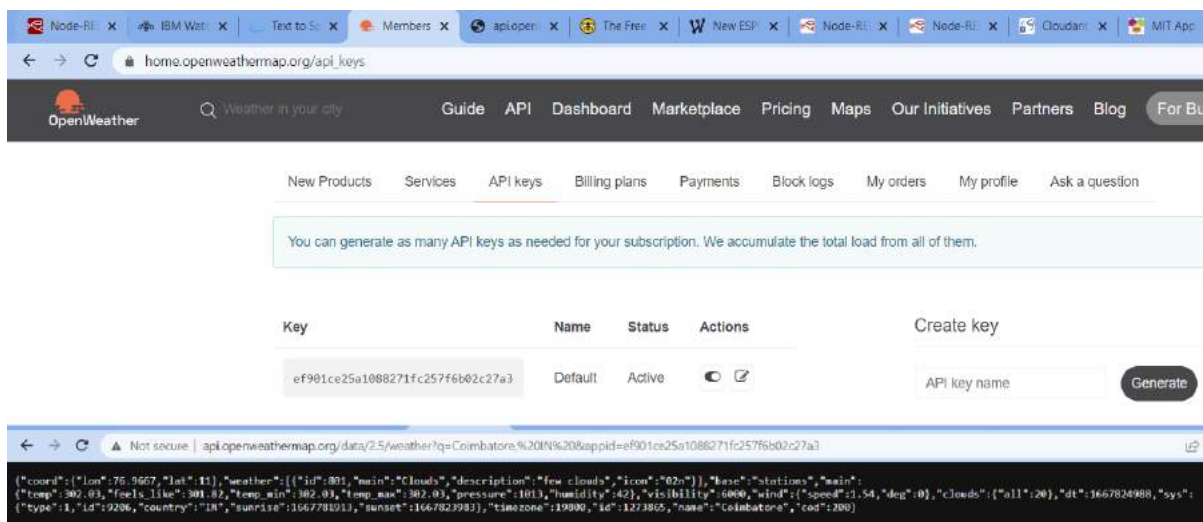


FIG – 6 WEATHER API

Chapter-5

IMPLEMENTATION OF THE SYSTEM

5.1 IBM Watson IOT Device Deployment

To deploy IBM-Watson the following credentials are important so that we can be able to connect our device to desired platform

1. OrgID
2. DeviceID
3. Device Type
4. Token
5. API

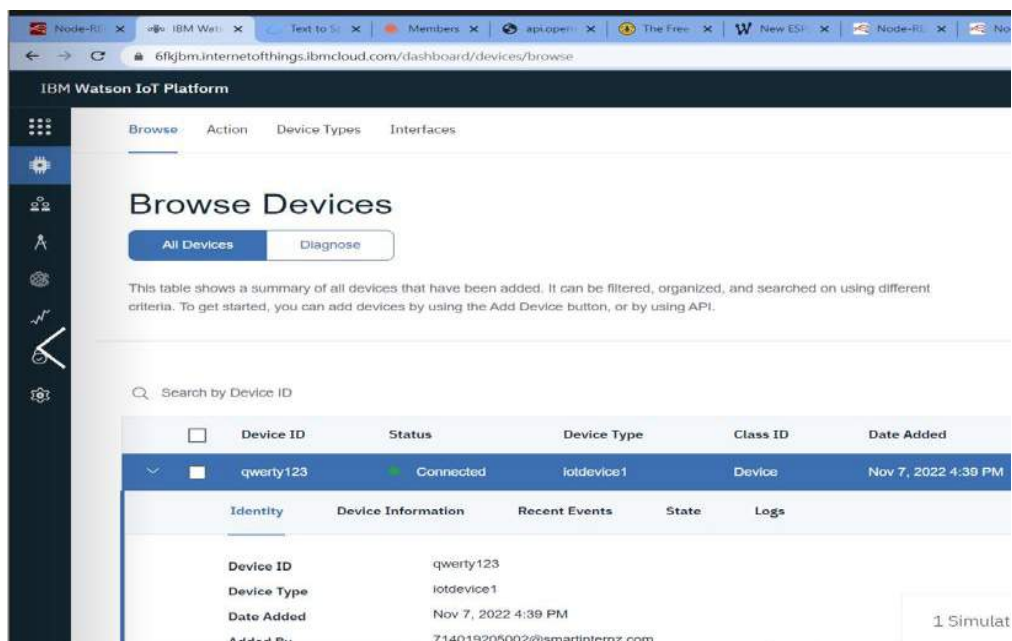


FIG-7 IBM WATSON IOT CREDENTIALS

5.2 Deployment Of Node-Red Services

The node IBM IoT App In is added to Node-Red workflow. Then the appropriate device credentials obtained earlier are entered into the node to connect and fetch device telemetry to Node-Red

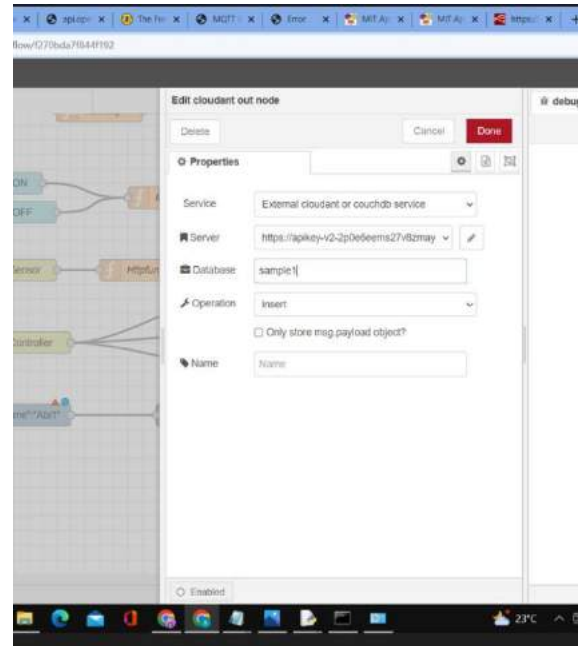
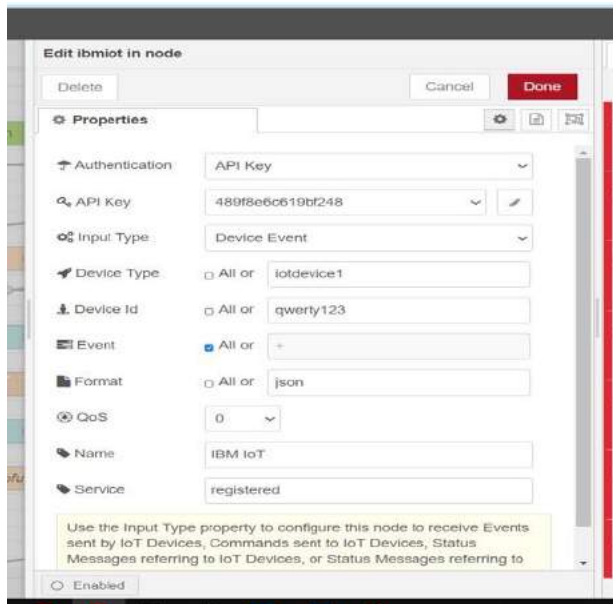


FIG-8 NODE-RED TO IOT WATSON FIG-9 NODE-RED TO CLOUDANT CONNECTIVITY

Once it is connected Node-Red receives data from the deviceDisplay the data using debug node for verification Then connect function node and write the Java script code to get each reading separately. The Java script code for the function node is:

```
msg.payload = msg.payload.d.temperature;
return msg;
msg.payload = msg.payload.d.humidity;
return msg;
```

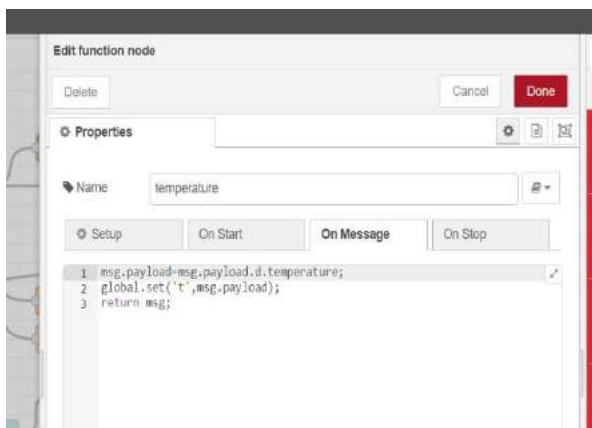


FIG-10 TEMPERATURE READING

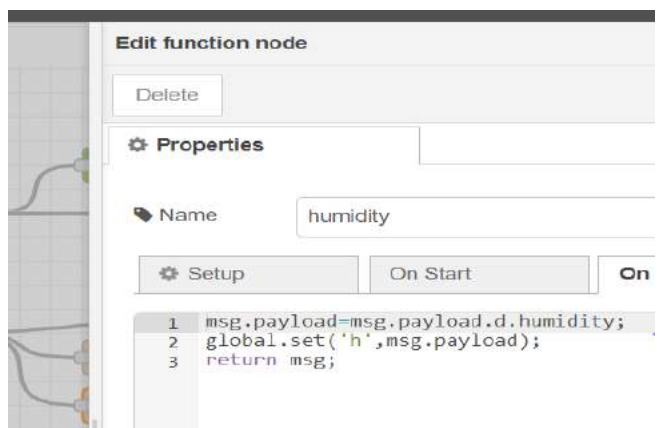


FIG -11 HUMIDITY READING

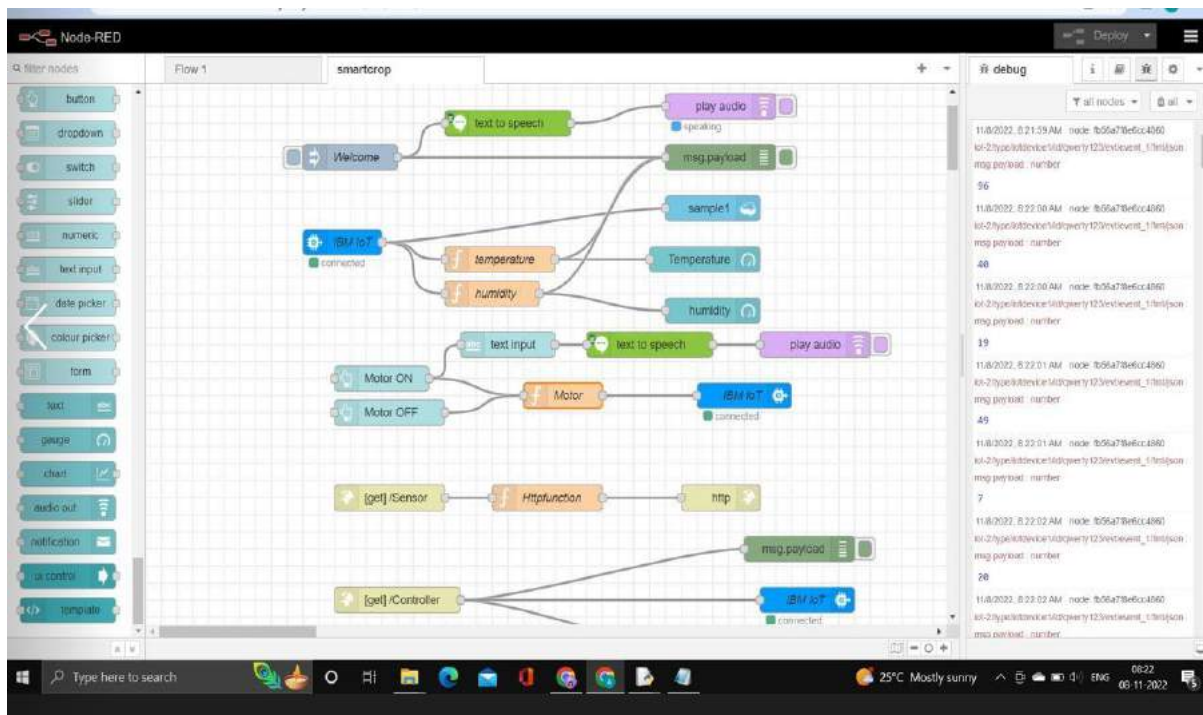


FIG- 12 NODE-RED FLOW EDITOR WITH TEXT TO SPEECH

5.3 Deployment Of MIT-App Simulator

MIT-APP Inventor has been deployed with the node red to render sensor data from the NODE-Red o the MIT-APP simulator

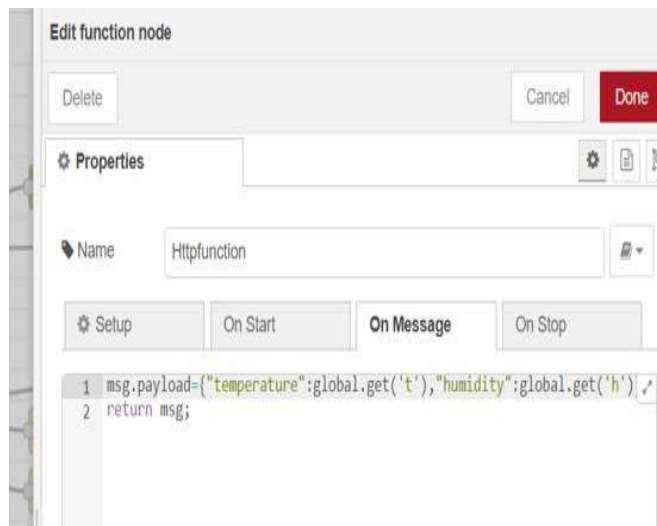
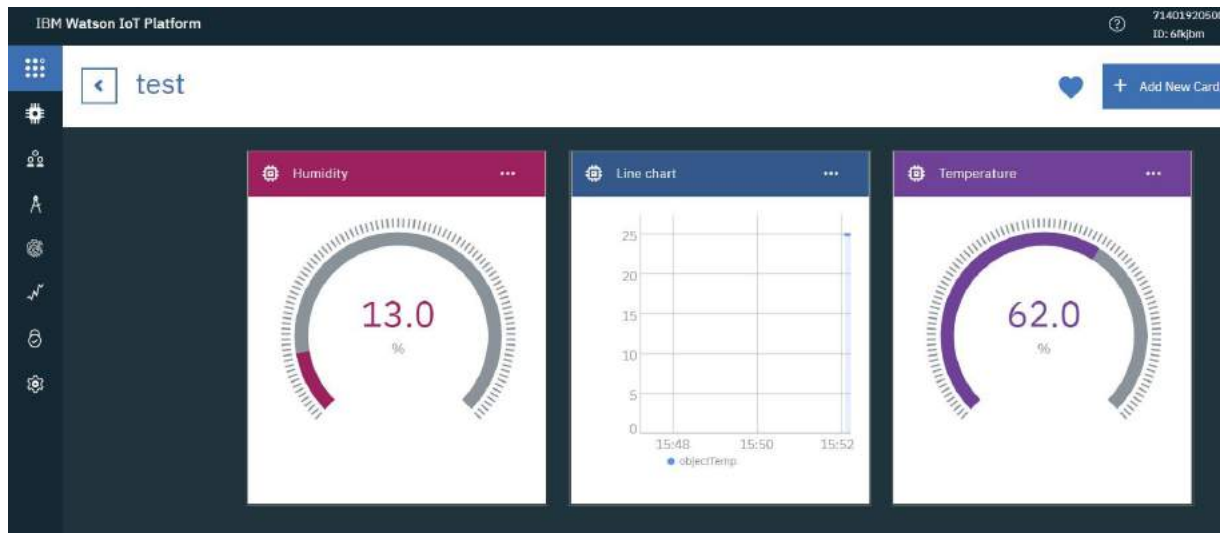


FIG – 13 HTTP CONNECTION TO MIT-APP

Chapter – 6

OBSERVATIONS AND RESULTS

Output Of Web-Based Application Using Node-Red are depicted as follows

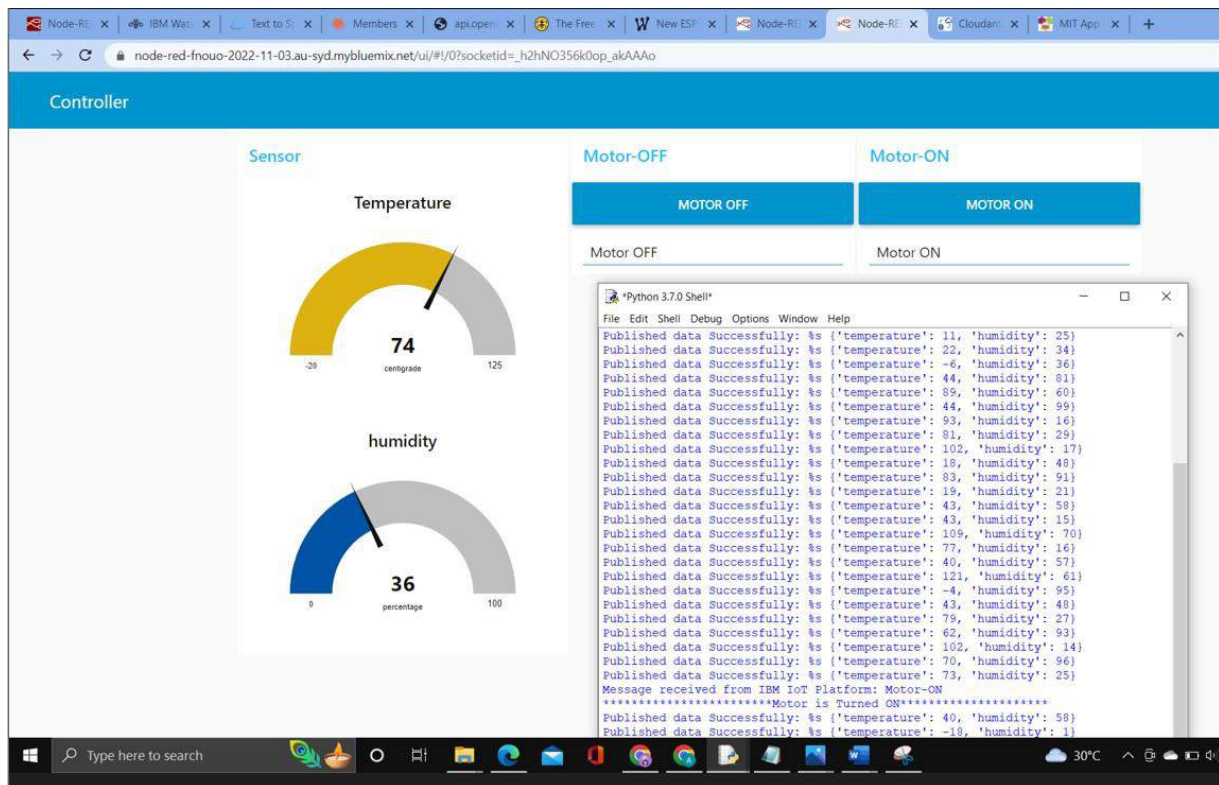


The screenshot shows the 'Browse Devices' page in the IBM Watson IoT Platform. It includes a search bar, a table of device information, and a '1 Simulat' button.

Device ID	Status	Device Type	Class ID	Date Added
qwerty123	Connected	iotdevice1	Device	Nov 7, 2022 4:39 PM

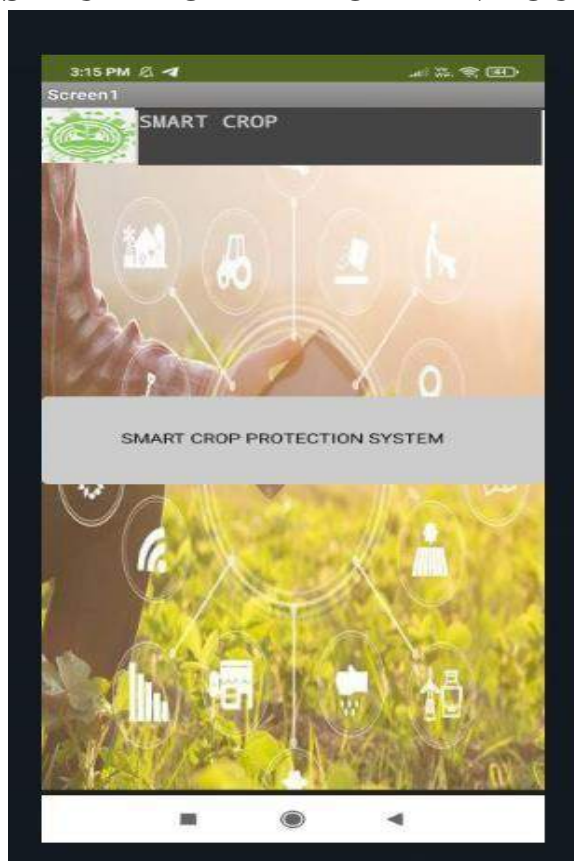
Identity	Device Information	Recent Events	State	Logs
Device ID	qwerty123			
Device Type	iotdevice1			
Date Added	Nov 7, 2022 4:39 PM			
Added By	714019205002@smartinternz.com			

Sensor Data Displayed in Device Boards As Separate Cards



NODE-RED WEB GUI

SIMULATOR DEPLOYMENT OUTPUT



HOME SCREEN



CONTROLLER SCREEN



DATA SCREEN

Chapter- 7

ADVANTAGES AND DISADVANTAGES

Advantages:

- ☐ Farms can be monitored and controlled remotely.
- ☐ Increase in convenience to farmers.
- ☐ Less Manpower.
- ☐ Better standards of living.

Disadvantages:

- ☐ Lack of internet/connectivity issues.
- ☐ Added cost of internet and internet gateway infrastructure.
- ☐ Farmers wanted to adapt the use of WebApp.

Chapter- 8

CONCLUSION

This system focuses on developing devices and tool to manage, display and alert the users using the advantages of a wireless sensor network system. It aims at making agriculture smart using automation and IoT. The cloud computing devices are used at the end of the system that can create a whole computing system from sensors to tools that observe data from agriculture field. It proposes a novel methodology for smart farming by including a smart sensing system and smart irrigator system through wireless communication technology. Thus, the objective of the project to implement an IoT system in order to help farmers to control and monitor their farms has been implemented successfully.

Chapter- 9

REFERENCE

1. Balaji Bhanu, Raghava Rao, J.V.N. Ramesh and Mohammed Ali hussain, “Agriculture Field Monitoring and Analysis using Wireless Sensor Networks for improving Crop Production”, Eleventh International Conference on Wireless and Optical Communications Networks (WOCN).2014.
2. Harshal Meharkure, Pargyline, Sheetal Israni, “Application of IOT Based System for Advance Agriculture in India”, International Journal of Innovative Research in Computer and Communication Engineering(IJIRCCE) Vol.
3. Issue 11, pp. 10831-10837, 2015. 3. LIU Dan, Cao Xin, Huang Chongwei, JI Liang Liang, “Intelligent agent greenhouse environment monitoring system based on IOT technology”, International Conference on Intelligent Transportation, Big Data & Smart City, 2015.
4. Mehdi Roopei, Paul Rad, Kim-Kwang Raymond Choo, “Cloud of Things in smart agriculture: Intelligent irrigation monitoring by Thermal Imaging” IEEE Cloud Computing, 2017.
5. IBM cloud reference: <https://cloud.ibm.com/>
6. Python code reference: <https://github.com/rachuriharish23/ibmsubscribe>
7. IoT simulator : <https://watson-iot-sensor-simulator.mybluemix.net/>

Chapter- 10

APPENDIX

CODE FOR PYTHON EDITOR

```
#IBM Watson IOT Platform
import wiotp.sdk.device
import time
import random
myConfig = {
    "identity": {
        "orgId": "6fkjbm",
        "typeId": "iotdevice1",
        "deviceId": "qwerty123"
    },
    "auth": {
        "token": "johnyjohnyyespapa"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']
    if(m=="Motor-ON"):
        print("*****Motor is Turned ON*****")
    else:
        print("*****Motor is Turned OFF*****")
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    temp=random.randint(-20,125)
    hum=random.randint(0,100)
    myData={'temperature':temp, 'humidity':hum}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0,
onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)
client.disconnect()
```

IBM TEXT TO SPEECH

```
from ibm_watson import TextToSpeechV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
authenticator =
IAMAuthenticator('M_u6yEvEGJylj_ysbL_pG0ZOKuRCQW1LgXUtv_IcBPCR')
text_to_speech = TextToSpeechV1(
    authenticator=authenticator
```

```

)
text_to_speech.set_service_url('https://api.au-syd.text-to-
speech.watson.cloud.ibm.com/instances/23724eb6-a096-4a3a-b914-da0e442c1c5f')
with open('hello_world.wav', 'wb') as audio_file:
    audio_file.write(
        text_to_speech.synthesize(
            'Alert',
            voice='en-US_AllisonV3Voice',
            accept='audio/wav'
        ).get_result().content)

```

WOWKI PLATFORM – TEMPERATURE SENSOR

SAVE
SHARE
sketch.ino

sketch.ino

diagram.json

libraries.txt

Library Manager

```

1  #include <WiFi.h>//library for wifi
2  #include <PubSubClient.h>//library for M
3  #include "DHT.h"// Library for dht11
4  #define DHTPIN 15    // what pin we're c
5  #define DHTTYPE DHT22    // define type o
6  #define LED 2
7  DHT dht (DHTPIN, DHTTYPE);// creating th
8
9  void callback(char* subscribetopic, byte
10
11  //-----credentials of IBM Accounts-----
12
13  #define ORG "6fkjbm"//IBM ORGANITION ID
14  #define DEVICE_TYPE "iotdevice1"//Device
15  #define DEVICE_ID "qwerty123"//Device ID
16  #define TOKEN "johnyjohnyespapa"    //
17  String data3;
18  float h, t;
19
20
21  //----- Customise the above values --
22  char server[] = ORG ".messaging.internet
23  char publishTopic[] = "iot-2/evt/Data/fm
24  char subscribetopic[] = "iot-2/cmd/cmd/f
25  char subscribetopic1[] = "iot-2/cmd/comm
26  char authMethod[] = "use-token-auth";//
27  char token[] = TOKEN;
28  char clientId[] = "d:" ORG ":" DEVICE_TY
29
30
31  //-----
32  WiFiClient wifiClient; // creating the i
33  PubSubClient client(server, 1883, callba
34  void setup()// configureing the ESP32

```

Simulation

00:17.461

95%

```

{"temperature":24.00,"humidity":40.00}
Publish ok
temperature:24.00
humidity:40.00
Sending payload:
{"temperature":24.00,"humidity":40.00}
Publish ok

```

CODE FOR MIT-APP INVENTOR (SCREEN 3)

