

## **SMARTFARMER - IoT ENABLED SMART FARMING APPLICATION**

**TEAM ID: PNT2022TMID18361**

**TEAM LEAD: DARANYA K**

**TEAM MEMBER 1: KIRUBA N**

**TEAM MEMBER 2: BHAVASRI V**

**TEAM MEMBER 3: ARCHANA S**

# **CONTENTS**

1. INTRODUCTION
2. PROBLEM STATEMENT
3. PROPOSED SOLUTION
4. THEORETICAL ANALYSIS
  - 4.1. BLOCK DIAGRAM
  - 4.2. REQUIRED SOFTWARE INSTALLATION
    - 4.1.A. PYTHON IDE
    - 4.1.B. IBM WATSON IoT PLATFORM
    - 4.1.C. NODE-RED
    - 4.1.D. MIT APP INVENTOR
5. BUILDING PROJECT
  - 5.1. BUILDING THE PYTHON CODE
  - 5.2. CONNECTING IBM IoT WATSON AND THE PYTHON  
CODE
  - 5.3. CONFIGURING DEVICE IN THE NODE-RED
  - 5.4. DEVELOPING APP IN MIT APP INVENTOR
6. FLOW DIAGRAM
7. OBSERVATIONS AND RESULTS
8. ADVANTAGES AND DISADVANTAGES
9. CONCLUSION
10. BIBLIOGRAPHY

## 1. INTRODUCTION

The main aim of this project is to help farmers automatically watch their farms by providing them with a Web App through which they can monitor the parameters of the field like Temperature, soil moisture, humidity using the sensors that are implanted on the fields and control the equipment like water motor remotely via internet without their actual presence in the field.

## 2. PROBLEM STATEMENT

Farmers are in need of present at farm for its maintenance irrespective of the weather conditions and any emergency works. They have to ensure that the crops are well watered which have to performed physically. Farmer have to stay most of the time infield in order to get a good yield. In the presence of pandemic also they have to work hard in their fields risking their lives to provide food for the country.

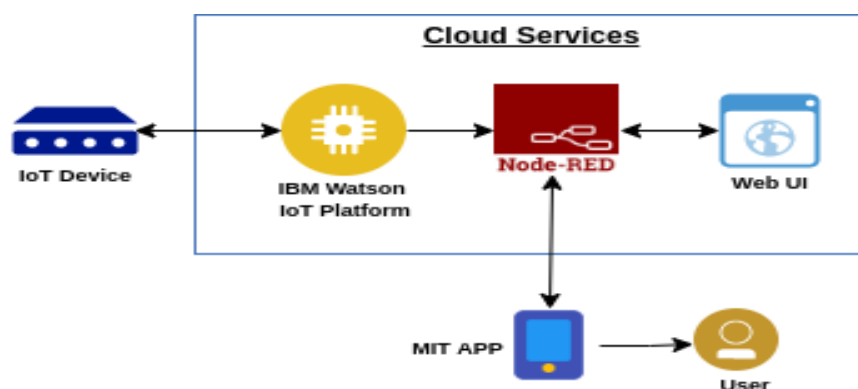
## 3. PROPOSED SOLUTION

In order to get rid of the conditions, the iot devices such as sensors are introduced in the fields. A web application is also developed which can be linked with the sensors in the field and this application can be viewed by the farmers on their smartphones. From this we can ensure that the fields can be monitored through the internet and they can also control the motor via internet according to the soil moisture sensor levels.

## 4. THEORETICAL ANALYSIS

For implementing the above proposed solution, we are considering the below block diagram.

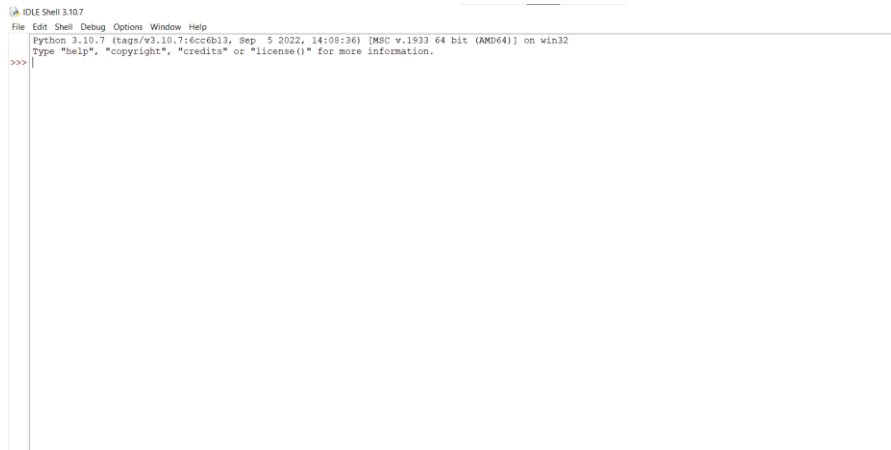
### 4.1. BLOCK DIAGRAM



## 4.2. REQUIRED SOFTWARE INSTALLATION

### 4.2.A. PYTHON IDE

- Install a python3 compiler.
- Install any python IDE to establish the code.



### 4.2.B. IBM WATSON IoT PLATFORM

IBM Watson IoT Platform is a fully managed, cloud-hosted service with capabilities for device registration, connectivity, control, rapid visualization and data storage. In Watson IoT platform, we can configure the devices such as sensors.

### 4.2.C. NODE-RED

IBM Node-red is a flow-based programming tool for wiring together hardware devices, APIs, and online services. Create event-based apps with simple flow-based programming. Get started with Node-RED.

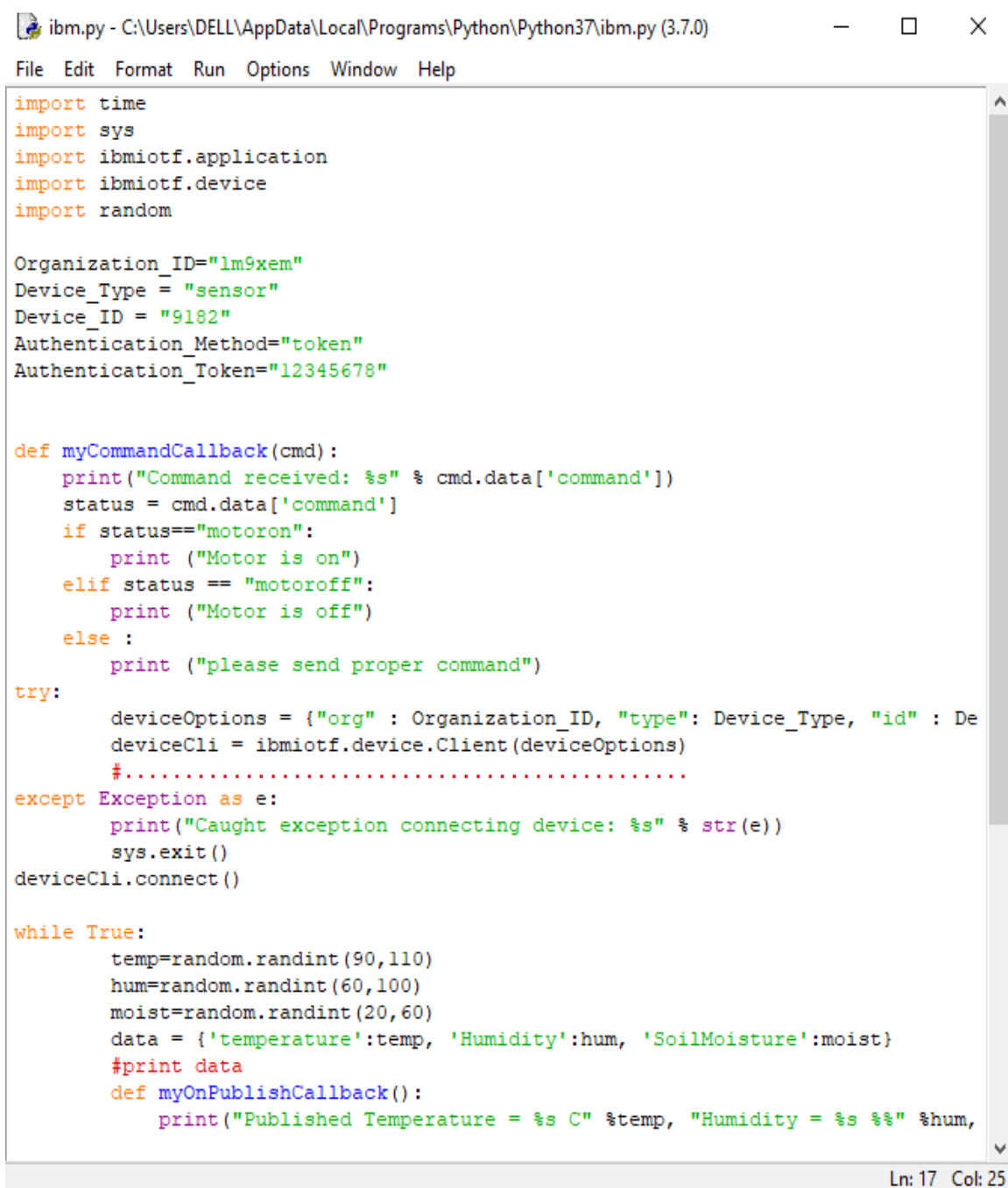
### 4.2.D. MIT APP INVENTOR

MIT App Inventor is an intuitive, visual programming environment that allows everyone even children to build fully functional apps for smartphones and tablets.

## 5. BUILDING PROJECT

### 5.1.BUILDING THE PYTHON CODE

- Implement the python code in order to turn on and off the motor via the mobile application.
- For implementing this, a device is configured on the IBM Iot Watson platform. This device is made up of sensors such as Temperature sensors, Humidity sensors and Soil moisture Sensors.
- The device is linked with the python code using the Device ID.



```
ibm.py - C:\Users\DELL\AppData\Local\Programs\Python\Python37\ibm.py (3.7.0)
File Edit Format Run Options Window Help

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

Organization_ID="lm9xem"
Device_Type = "sensor"
Device_ID = "9182"
Authentication_Method="token"
Authentication-Token="12345678"

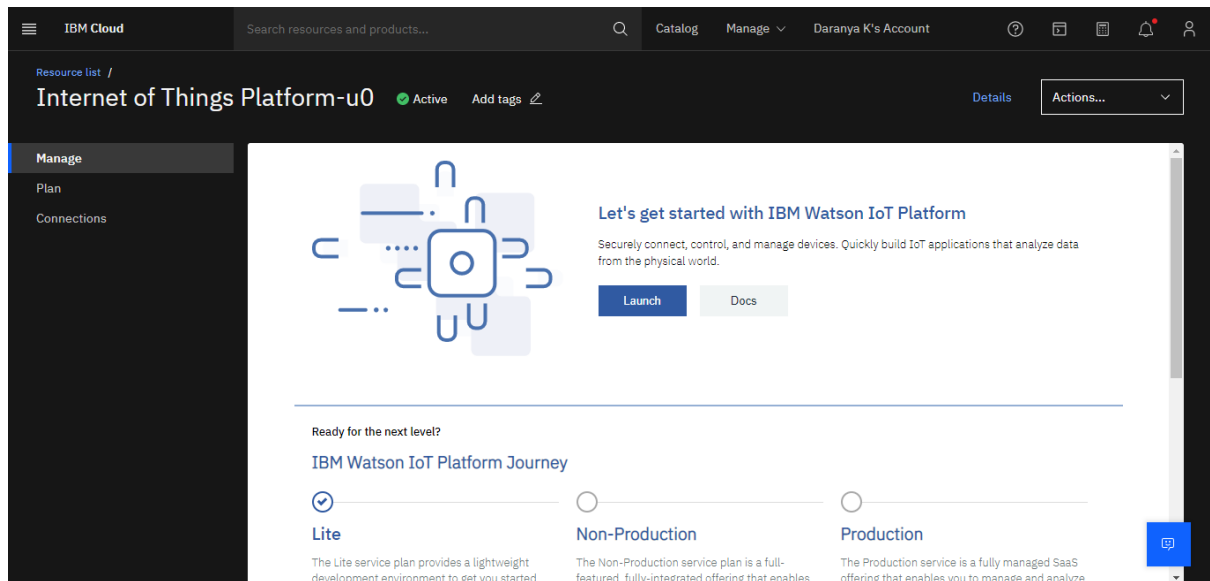
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status = cmd.data['command']
    if status=="motoron":
        print ("Motor is on")
    elif status == "motoroff":
        print ("Motor is off")
    else :
        print ("please send proper command")
try:
    deviceOptions = {"org" : Organization_ID, "type": Device_Type, "id" : Device_ID}
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....
except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()
deviceCli.connect()

while True:
    temp=random.randint(90,110)
    hum=random.randint(60,100)
    moist=random.randint(20,60)
    data = {'temperature':temp, 'Humidity':hum, 'SoilMoisture':moist}
    #print data
    def myOnPublishCallback():
        print("Published Temperature = %s C" %temp, "Humidity = %s %" %hum,
```

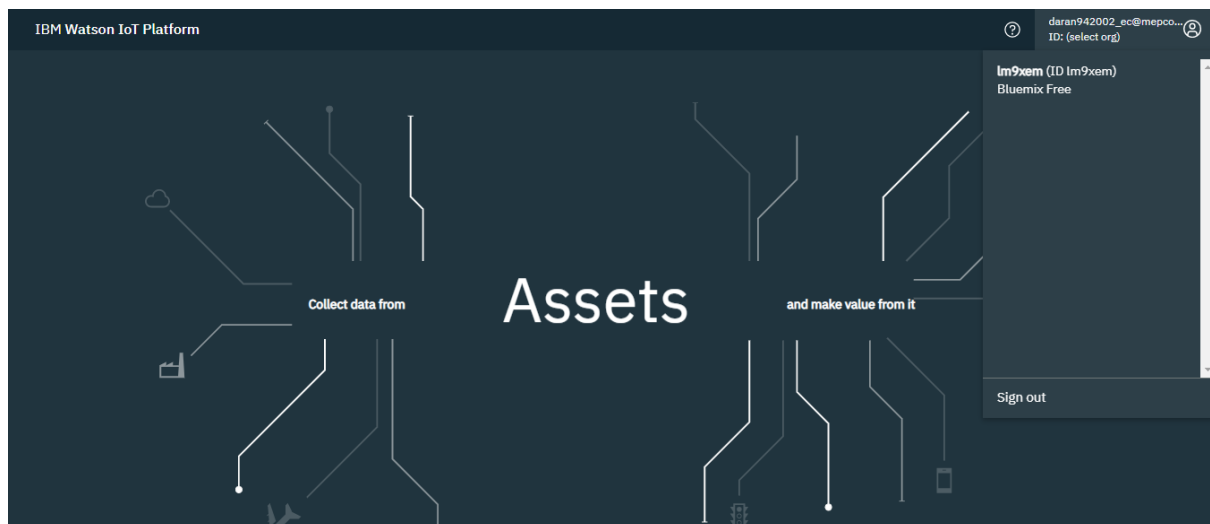
Ln: 17 Col: 25

## 5.2.CONNECTING IBM IoT WATSON AND THE PYTHON CODE

- Login into the IBM account with the given Smartinternz mail ID.
- Once logged in, select IoT in resource list.
- The IoT platform will be launched.



- Once the IoT is launched, IBM IoT Watson will be opened.
- We are going to use the BlueMix Simulator to connect the sensors.



- The sensors can be installed by giving Add Devices in the right corner.

IBM Watson IoT Platform

daran942002\_ec@mepcoeng.ac.in  
ID: lm9xem

Browse Action Device Types Interfaces

**Browse Devices**

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator ☒ ☐ ☐

<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added
> <input type="checkbox"/>	1234	Disconnected	project	Device	Nov 17, 2022 5:43 PM
> <input type="checkbox"/>	9182	Disconnected	sensor	Device	Nov 15, 2022 12:17 PM

- Fill in the needed details such as Device type and Device ID in order to link it to the python code.

### Add Device

Identity Device Information Security Summary

Select a device type for the device that you are adding and give the device a unique ID.

Device Type

Device ID

- After providing all the details, the device will get added.

### Add Device

Identity Device Information Security Summary

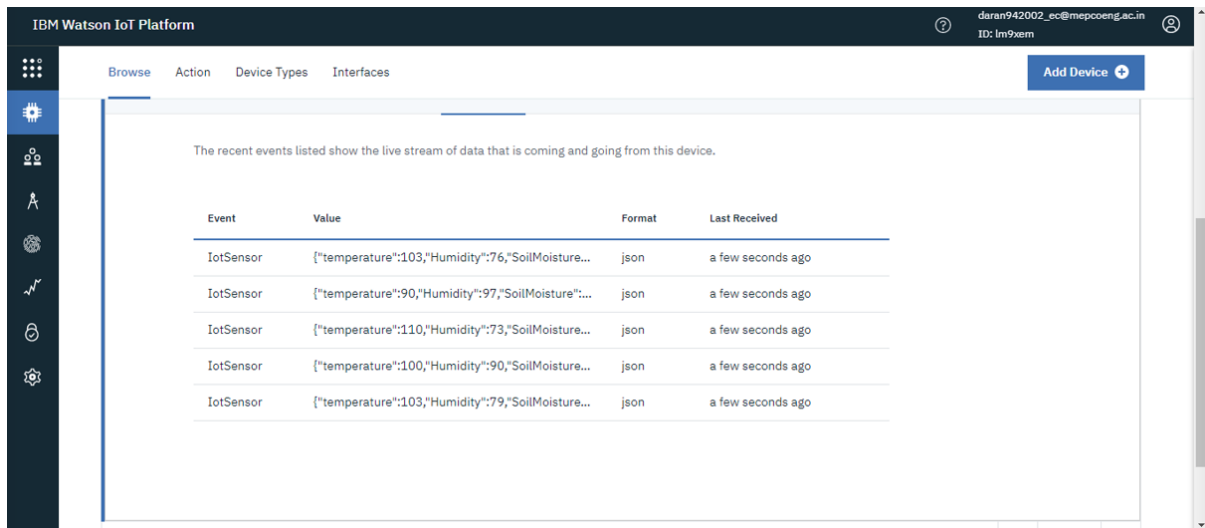
Verify that the following information is correct then select Finish

Device Type  
sensor

Device ID  
91825

Security Token  
12345678

- Once the device is added, run the python code.
- On running the python code, the random values for the Temperature, Humidity and soil moisture sensors are displayed in the IBM IoT Watson platform.

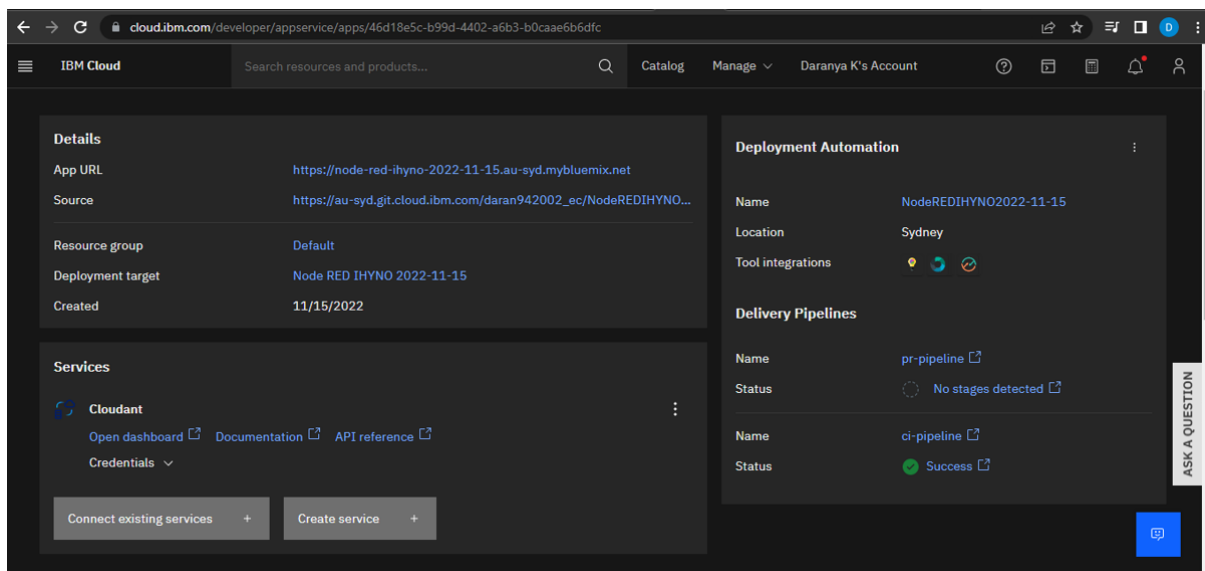


The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons. The main content area displays a table of recent events from an IoT sensor.

Event	Value	Format	Last Received
IotSensor	["temperature":103,"Humidity":76,"SoilMoisture"...	json	a few seconds ago
IotSensor	["temperature":90,"Humidity":97,"SoilMoisture"...	json	a few seconds ago
IotSensor	["temperature":110,"Humidity":73,"SoilMoisture"...	json	a few seconds ago
IotSensor	["temperature":100,"Humidity":90,"SoilMoisture"...	json	a few seconds ago
IotSensor	["temperature":103,"Humidity":79,"SoilMoisture"...	json	a few seconds ago

### 5.3. CONFIGURING DEVICE IN THE NODE-RED

- Once the random values are generated in the Iot Watson platform, select the Node-red in the catalog under IBM cloud.



The screenshot shows the IBM Cloud Developer console. The left sidebar contains 'Details' and 'Services' sections. The 'Details' section shows the App URL, Source, Resource group, Deployment target, and Created date. The 'Services' section shows the Cloudant service with links to the dashboard, documentation, and API reference. The right sidebar contains 'Deployment Automation' and 'Delivery Pipelines' sections. The 'Deployment Automation' section shows the Name, Location, and Tool integrations. The 'Delivery Pipelines' section shows the Name, Status, and a 'Success' message.

**Details**

- App URL: <https://node-red-ihyno-2022-11-15-au-syd.mybluemix.net>
- Source: [https://au-syd.git.cloud.ibm.com/daran942002\\_ec/NodeREDIHYN022-11-15](https://au-syd.git.cloud.ibm.com/daran942002_ec/NodeREDIHYN022-11-15)
- Resource group: Default
- Deployment target: Node RED IHYN0 2022-11-15
- Created: 11/15/2022

**Services**

- Cloudant
  - Open dashboard
  - Documentation
  - API reference
  - Credentials

**Deployment Automation**


- Name: NodeREDIHYN022-11-15
- Location: Sydney
- Tool integrations: [Icons]

**Delivery Pipelines**

- Name: pr-pipeline
- Status: No stages detected
- Name: ci-pipeline
- Status: Success



- Select Go compute in the resource list and open the node-red .

▼ Name	↑ Group	Location	Product	Status	Tags
Filter by name or IP address...	Filter by group or org...	Filter...	Filter...	Filter...	Filter...
^ Compute (1)					
 Node RED IHYNO 2022-11-15	z9y8x7 / ibm	Sydney	Node.js	Started	—

- Go to visit app URL.

Resource list / **Node RED IHYNO 2022-11-15** Running [Visit App URL](#) [Add tags](#) [Details](#) [Actions...](#)

Getting started

**Overview**

Runtime

Connections

Logs

API Management

Autoscaling

**Instances** [Edit](#)

Health **100%**  
1/1 instance(s) are running

Instances  - +

MB memory per instance

0  2048 256

**Runtime**

Node.js

**256**  
Total MB allocation

1.75 GB still available

Free Used

**Runtime cost**

Current and estimated cost excludes connected services.

**Connections (1)**

[Go to your Node-RED flow editor](#)

- Go to node-red flow editor

# Node-RED

Flow-based programming for the Internet of Things

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

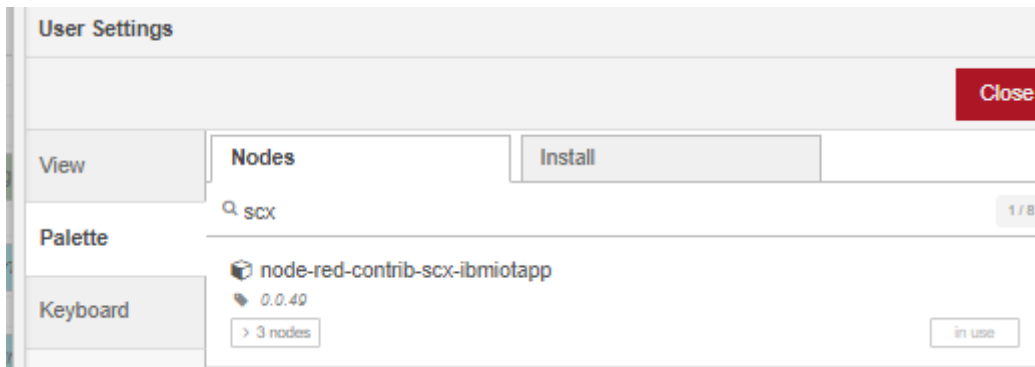
This instance is running as an IBM Cloud application, giving it access to the wide range of services available on the platform.

More information about Node-RED, including documentation, can be found at [nodered.org](https://nodered.org).

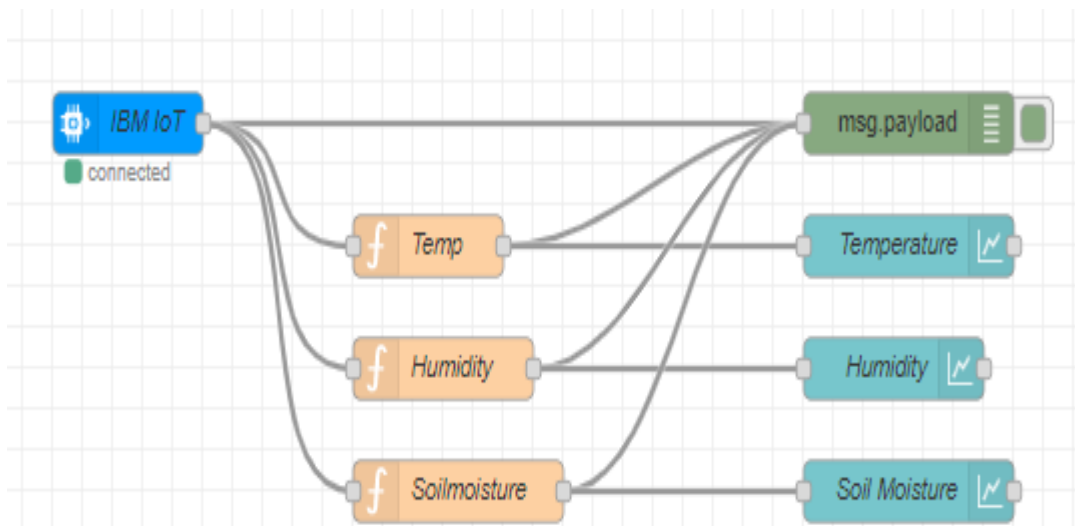
[Go to your Node-RED flow editor](#)

[Learn how to customise Node-RED](#)

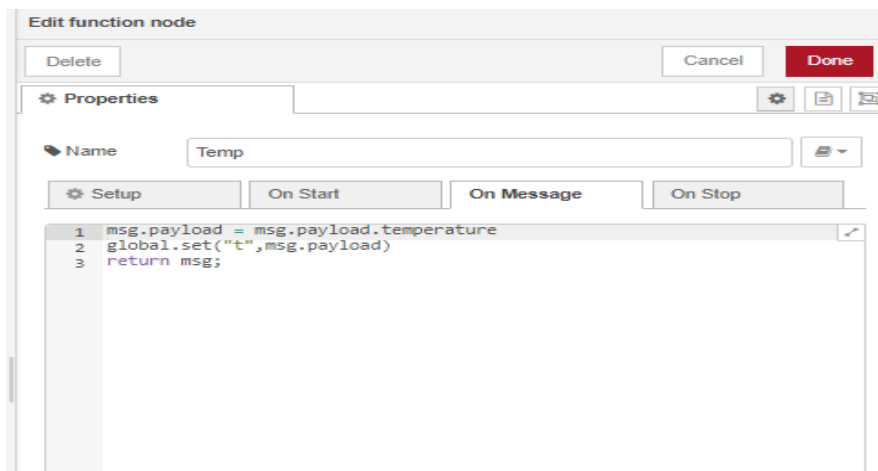
- Once enter inside install the scx-ibmiotapp under the manage palette in the right corner.



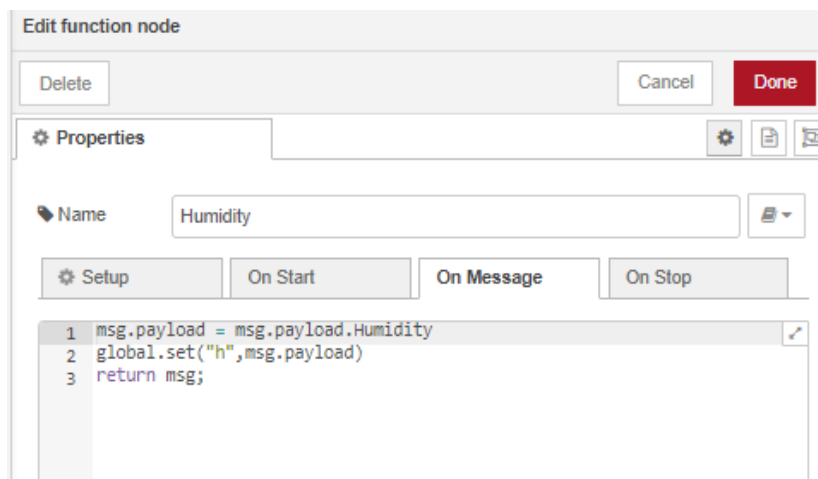
- Once installed, we able to use ibmiot which shows connected and retrieve the data from the python code which is linked with the IBM Watson platform.
- The functions for Temperature sensors, Humidity Sensors and Soil moisture sensors should be written in Javascript.
- The flow diagram in the node-red is given below:



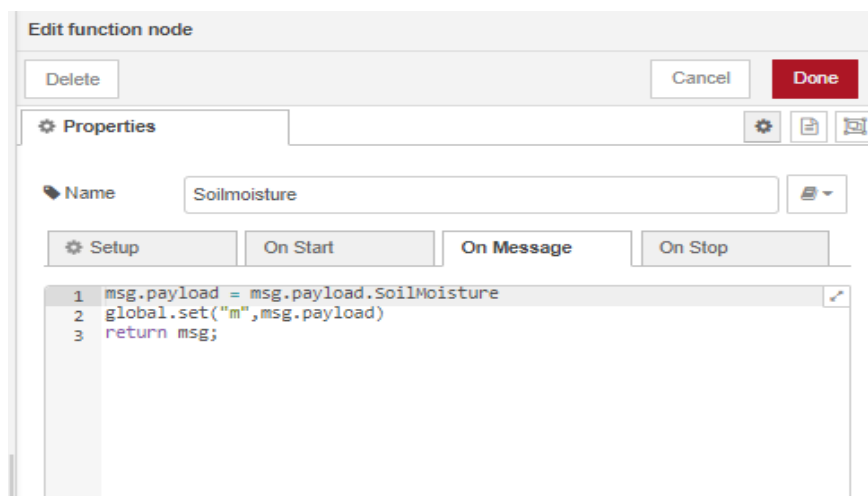
- The javascript for the temperature sensor in the node-red application:



- The javascript for the Humidity sensor in the node-red application:



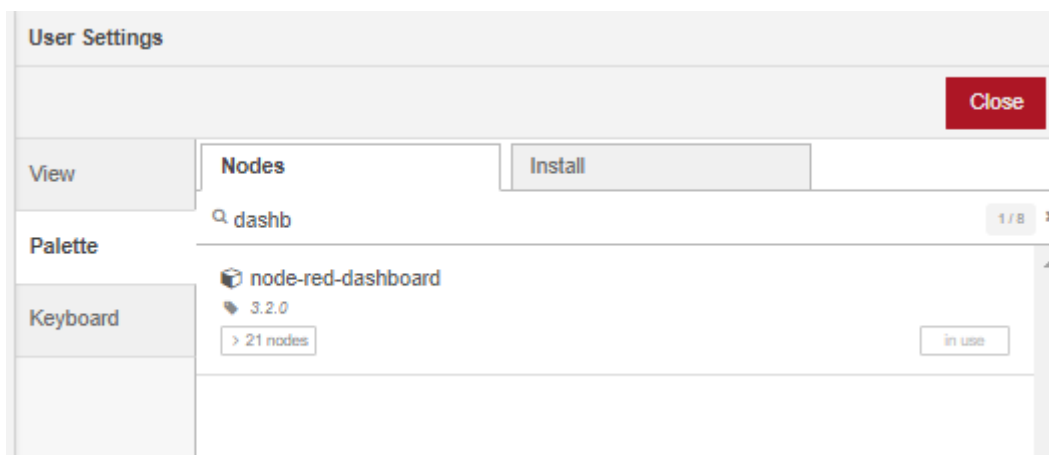
- The javascript for the Soil moisture sensor in the node-red application:



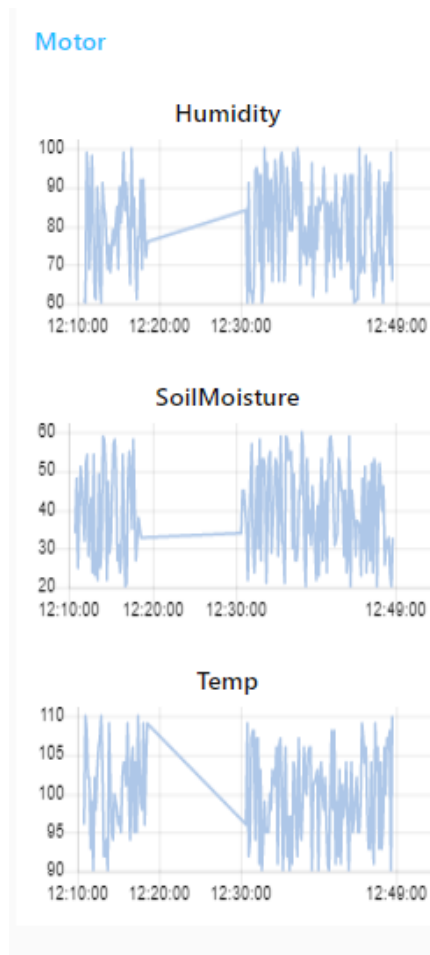
- Once the Program simulated, the readings are shown in the node-red as follows:



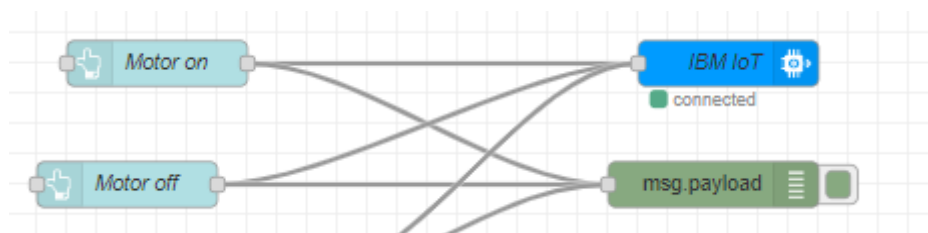
- We need to install the node-red dashboard by searching in the dashboard.



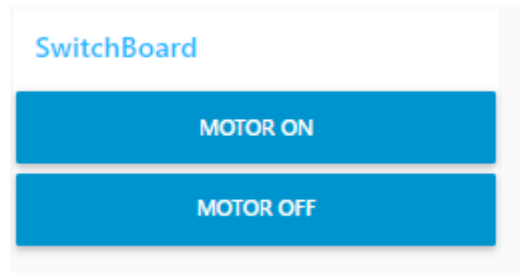
- Once the dashboard is installed, the readings for the temperature sensors, Humidity sensors and the soil moisture sensors can be viewed in the dashboard.



- After that connect the Motor on and off with IBM IoT in order to control the motor according to the soil moisture sensor values,



- A Switch Board is created with the label MOTOR ON and MOTOR OFF.



- For creating the labels, the edit button option is used. Here the requirements for the button can be given.
- One thing should be noted is that command for the motor on and off should be given as the same of the IBM IoT Watson Platform.

Edit button node

Delete
Cancel
Done

Properties

Group
[SmartFarming] SwitchBoard

Size
auto

Icon
optional icon

Label
Motor On

Tooltip
optional tooltip

Color
optional text/icon color

Background
optional background color

When clicked, send:

Payload
{"command":"motoron"}

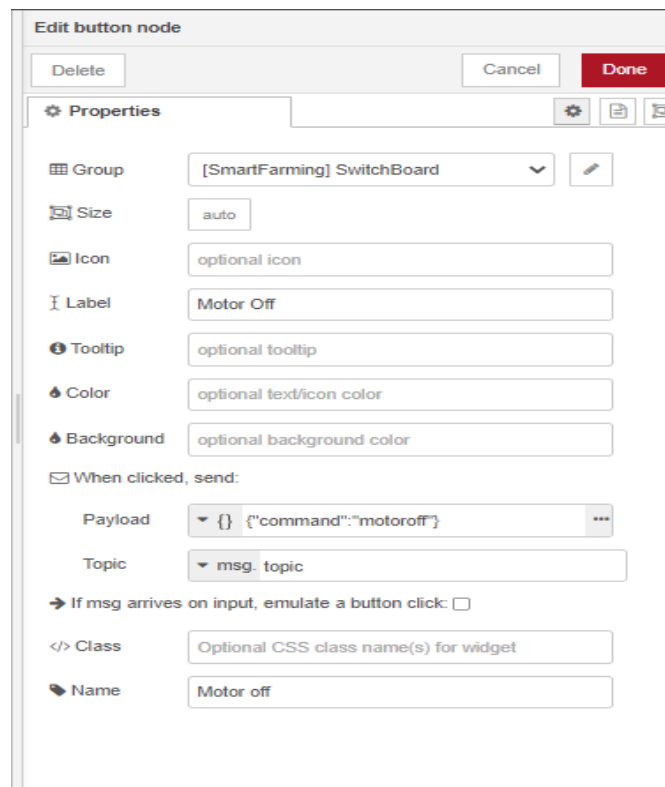
Topic
msg. topic

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

Class
Optional CSS class name(s) for widget

Name
Motor on

- The label for motor off is given as:



**Edit button node**

Delete Cancel Done

**Properties**

Group [SmartFarming] SwitchBoard

Size auto

Icon optional icon

Label **Motor Off**

Tooltip optional tooltip

Color optional text/icon color

Background optional background color

When clicked, send:

Payload {} {"command":"motoroff"}

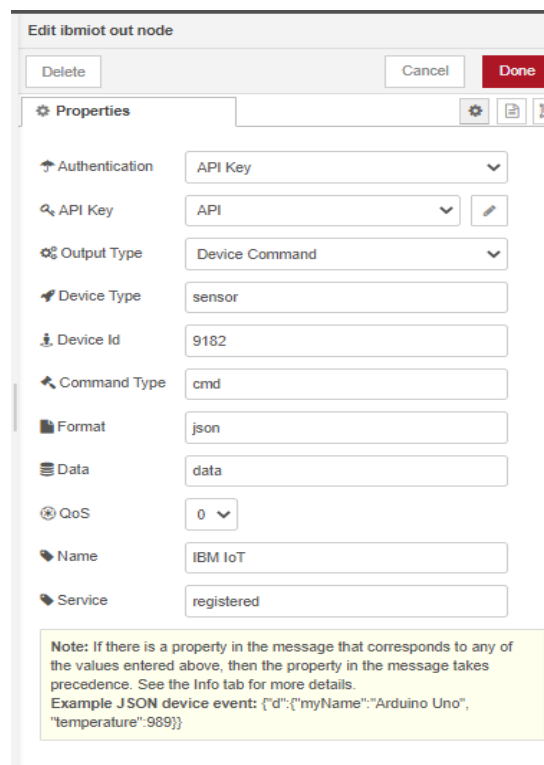
Topic msg.topic

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

Class Optional CSS class name(s) for widget

Name Motor off

- Once the commands are given and the switch board is created, the device i.e.) motor details should be given in ibmiot.



**Edit ibmiot out node**

Delete Cancel Done

**Properties**

Authentication API Key

API Key API

Output Type Device Command

Device Type sensor

Device Id 9182

Command Type cmd

Format json

Data data

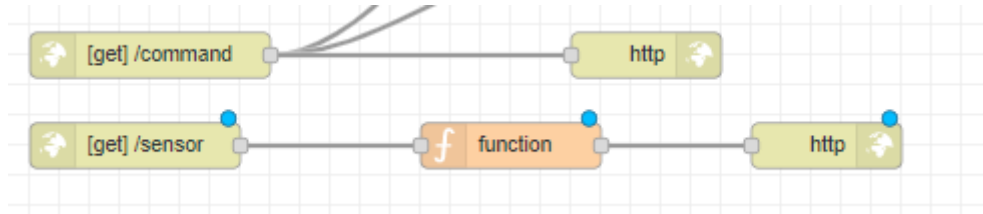
QoS 0

Name IBM IoT

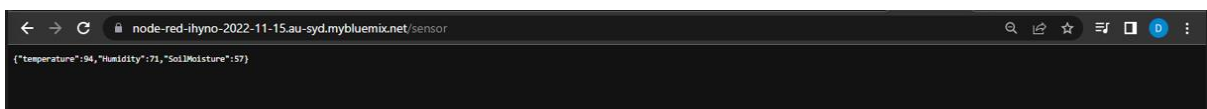
Service registered

**Note:** If there is a property in the message that corresponds to any of the values entered above, then the property in the message takes precedence. See the Info tab for more details.  
Example JSON device event: {"d":{"myName":"Arduino Uno", "temperature":989}}

- Once all the devices are configured, the webpage should be created for displaying the values from the sensors and to display the status of the motor whether it is on or off.

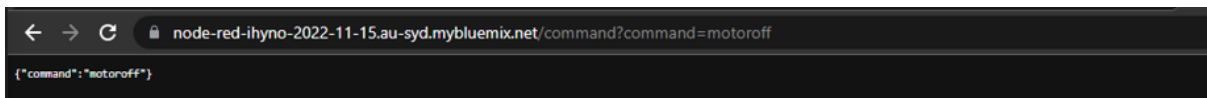


- The webpage URL showing the sensors values:

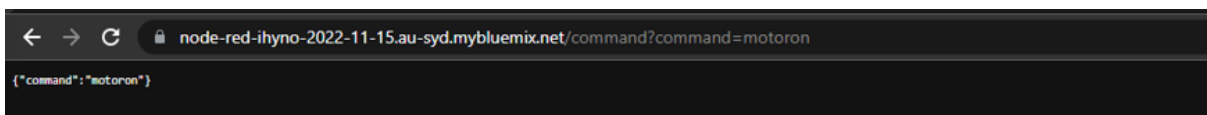


- The webpage URL showing the status of the motor :

When the motor is off:

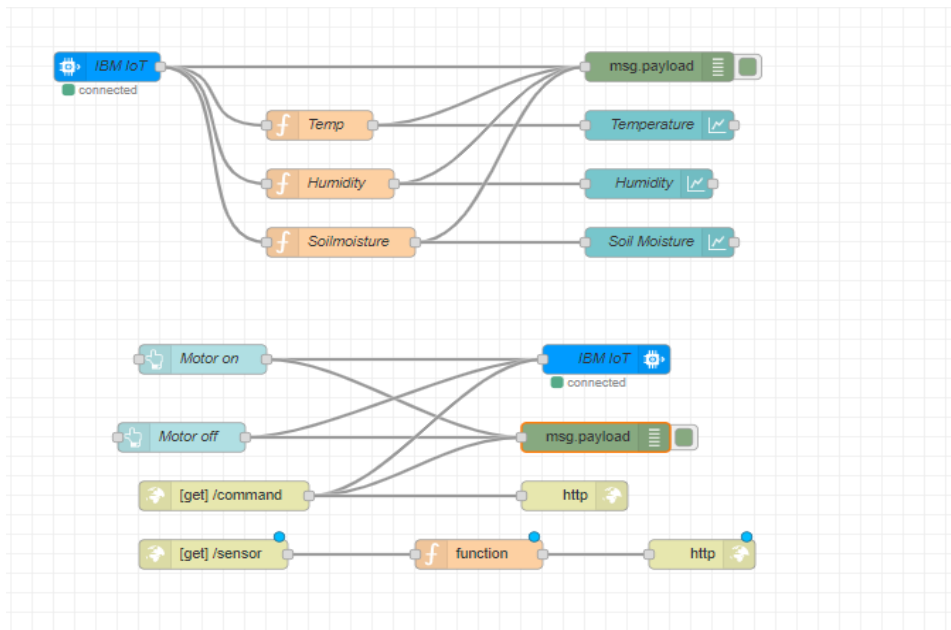


When the motor is on:



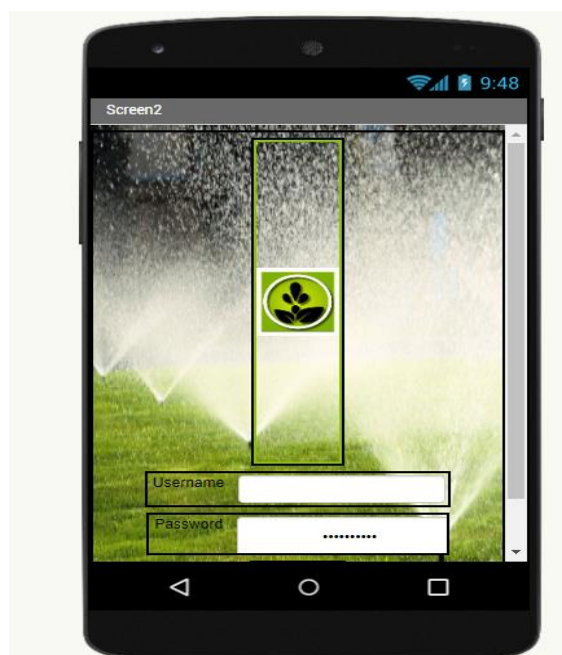


- The final flow diagram in the node-red is given below:

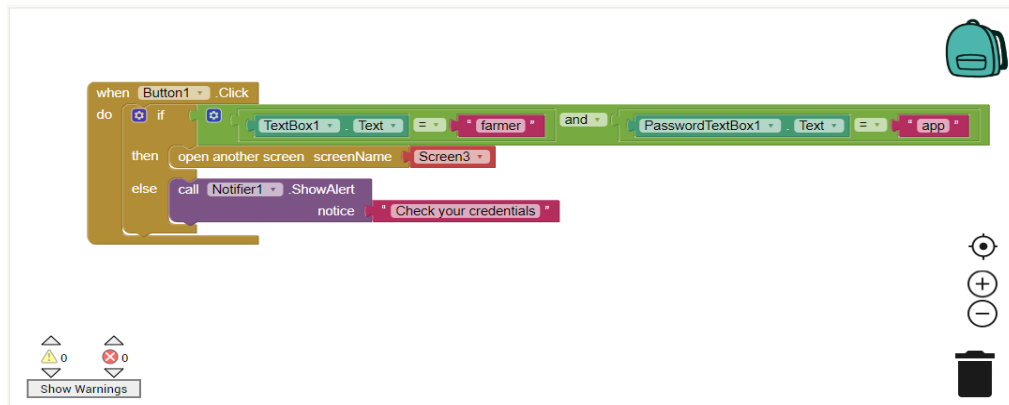


## 5.4. DEVELOPING APP IN MIT APP INVENTOR

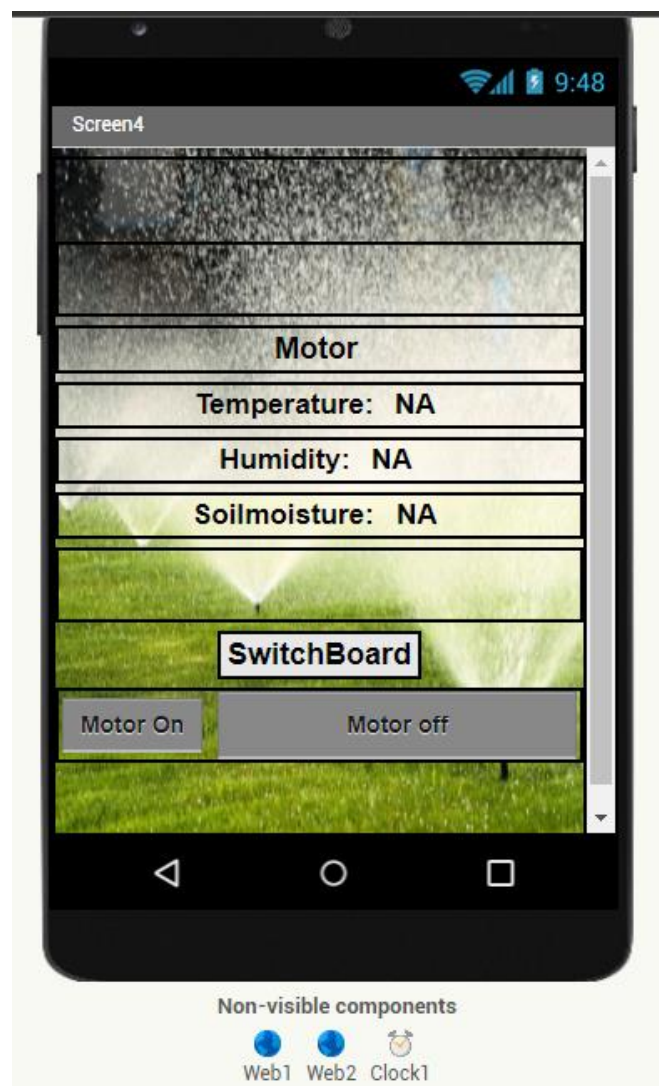
- In the MIT app inventor, the mobile app is developed by dragging the desired blocks.
- The main screen is developed as the home page for the app which is asking username and password.



- The above screen is designed using the password box and the text box. The design is given by the block diagram given below:



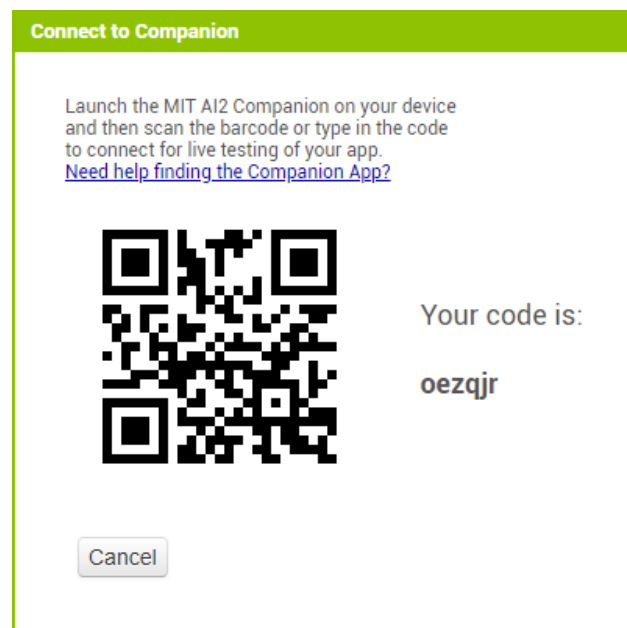
- After that screen 2 is developed i.e., the screen which displays the sensors values and contains the switch board which is used to control the motor via the web application.



- Here we can see there are invisible components: Web1, Web2, Clock1. The webpage we created in the node-red platforms which shows the value of the sensors and the status of the motor is linked with the MIT app inventor.
- The design for the above URL page link is given by the block diagram is given below:



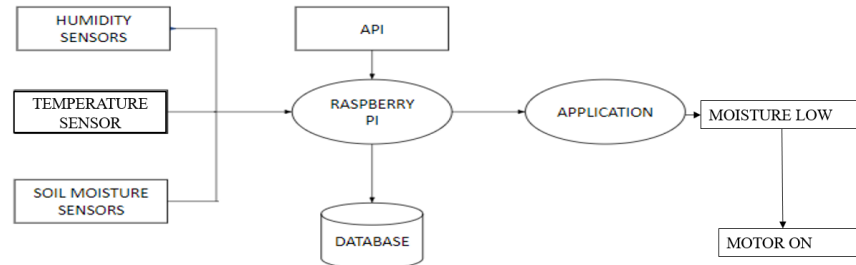
- Once the screens are designed, the QR code will scanned in order to access the app at the smartphone.
- The QR code for the app we developed is shown below:



- Once the app is shown, the measured sensors values are displayed and we can control the motor from the smartphone itself.

- Whenever we are giving command as turn on and off, the change can be noted in the Python code.

## 6. FLOW DIAGRAM



## 7. OBSERVATIONS AND RESULTS

Once all the IBM IoT Watson, Node-red and python code is linked together, we can control the motor from the app itself and can view the information. The web page 1 and 2 is connected to blocks of MIT app inventor . Once the button with motor on is pressed in MIT app, the corresponding command is sent which is received by the python code will be shown in the compiler and vice versa. The farmer can know about the fields through the sensor values.

```

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Published Temperature = 100 C Humidity = 81 % Soil Moisture = 41 % to IBM Watson
Published Temperature = 96 C Humidity = 71 % Soil Moisture = 49 % to IBM Watson
Published Temperature = 91 C Humidity = 80 % Soil Moisture = 44 % to IBM Watson
Published Temperature = 94 C Humidity = 91 % Soil Moisture = 42 % to IBM Watson
Published Temperature = 103 C Humidity = 85 % Soil Moisture = 37 % to IBM Watson
Published Temperature = 92 C Humidity = 73 % Soil Moisture = 55 % to IBM Watson
Published Temperature = 109 C Humidity = 71 % Soil Moisture = 59 % to IBM Watson
Published Temperature = 98 C Humidity = 64 % Soil Moisture = 35 % to IBM Watson
Published Temperature = 90 C Humidity = 74 % Soil Moisture = 41 % to IBM Watson
Published Temperature = 97 C Humidity = 66 % Soil Moisture = 36 % to IBM Watson
Published Temperature = 101 C Humidity = 94 % Soil Moisture = 60 % to IBM Watson
Published Temperature = 99 C Humidity = 70 % Soil Moisture = 34 % to IBM Watson
Published Temperature = 98 C Humidity = 75 % Soil Moisture = 45 % to IBM Watson
Published Temperature = 104 C Humidity = 95 % Soil Moisture = 48 % to IBM Watson
Published Temperature = 102 C Humidity = 89 % Soil Moisture = 55 % to IBM Watson
Published Temperature = 103 C Humidity = 61 % Soil Moisture = 57 % to IBM Watson
Published Temperature = 95 C Humidity = 89 % Soil Moisture = 37 % to IBM Watson
Published Temperature = 100 C Humidity = 74 % Soil Moisture = 23 % to IBM Watson
Published Temperature = 106 C Humidity = 65 % Soil Moisture = 24 % to IBM Watson
Published Temperature = 101 C Humidity = 61 % Soil Moisture = 41 % to IBM Watson
Published Temperature = 94 C Humidity = 83 % Soil Moisture = 48 % to IBM Watson
Published Temperature = 96 C Humidity = 96 % Soil Moisture = 59 % to IBM Watson
Published Temperature = 90 C Humidity = 87 % Soil Moisture = 52 % to IBM Watson
Published Temperature = 93 C Humidity = 94 % Soil Moisture = 60 % to IBM Watson
Published Temperature = 107 C Humidity = 97 % Soil Moisture = 59 % to IBM Watson
Command received: motoron
Motor is on
Published Temperature = 93 C Humidity = 72 % Soil Moisture = 56 % to IBM Watson
Command received: motoroff
Motor is off
Command received: motoron
Motor is on
Command received: motoroff
Motor is off
Command received: motoron
Motor is on
Command received: motoroff
Motor is off
Published Temperature = 93 C Humidity = 95 % Soil Moisture = 30 % to IBM Watson

```

## **8. ADVANTAGES AND DISADVANTAGES**

### **8.1.ADVANTAGES**

- The app allows the farmer to monitor the fields even though they are in distant .
- The labour cost can be saved.
- In the same time, the farmer can monitor multiple fields.

### **8.2.DISADVANTAGES**

- Internet is very essential for the information sharing but in remote areas it can be a question that if we get the uninterrupted connection.
- To ensure that the app is user-friendly so that the farmers will not face any difficulties while using the app .

## **9. CONCLUSION**

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.

## **10.BIBLIOGRAPHY**

- The link for the node-red: <https://node-red-ihyno-2022-11-15.au-syd.mybluemix.net/red/#flow/5b453cec53400c10>
- The link for the node-red dashboard : [https://node-red-ihyno-2022-11-15.au-syd.mybluemix.net/ui/#!/0?socketid=JZ9A7ZsgwZ2dOx3\\_AAAP](https://node-red-ihyno-2022-11-15.au-syd.mybluemix.net/ui/#!/0?socketid=JZ9A7ZsgwZ2dOx3_AAAP)
- The MIT app inventor link:  
<http://ai2.appinventor.mit.edu/#5442614409953280>