# P PROJECT BASED EXPERIENTIAL LEARNING PROGRAM (NALAIYA THIRAN)

# IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

**A PROJECT REPORT**

***Submitted by***

**SANTHOSH KUMAR U (111719106137)**

**SARAN V (111719106138)**

**SYED NAWAZ HUSSIAN (111719106156)**

**RITHESH CHARAN RAJ (111719106127)**

# TEAM ID : PNT2022TMID16166

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**R.M.K.ENGINEERING COLLEGE**

(An Autonomous Institution)

# R.S.M. Nagar, Kavaraipettai-601 206

****



**NOVEMBER 2022**

1. **INTRODUCTION**
   1. OVERVIEW
   2. PURPOSE
2. **LITERATURE SURVEY**
   1. EXISTING PROBLEM
   2. PROPOSED SOLUTION
   3. BRAINSTORMING
3. **THEORITICAL ANALYSIS**
   1. BLOCK DIAGRAM
   2. HARDWARE / SOFTWARE DESIGNING
4. **FLOWCHART**
5. **REQUIREMENT ANALYSIS**
   1. FUNCTIONAL REQUIREMENT
   2. NON FUNCTIONAL REQUIREMENT
6. **ADVANTAGES & DISADVANTAGES**
   1. ADVANTAGES
   2. DISADVANTAGES
7. **APPLICATIONS**
8. **CONCLUSION**
9. **FUTURE SCOPE 10 BIBLIOGRAPHY 11 APPENDIX**
10. SOURCE CODE
11. NODE-RED FLOW

# INTRODUCTION:

* 1. **OVERVIEW**

This is a Smart Agriculture System project based on Internet Of Things (IoT), that can measure soil moisture and temperature conditions for agriculture using Watson IoT services. IoT is network that connects physical objects or things embedded with electronics, software and sensors through network connectivity that collects and transfers data using cloud for communication. Data is transferred through internet without human to human or human to computer interaction.

In this project we have not used any hardware.

Instead of real soil and temperature conditions, sensors IBM IoT Simulator is used which can transmit soil moisture temperature as required.

* + - **Project requirements**: Node-RED, IBM Cloud, IBM Watson IoT, Node.js, IBM Device, IBM IoT Simulator, Python 3.7, Open Weather API platform.
    - **Project Deliverables**: Application for IoT based Smart Agriculture System

# PURPOSE :

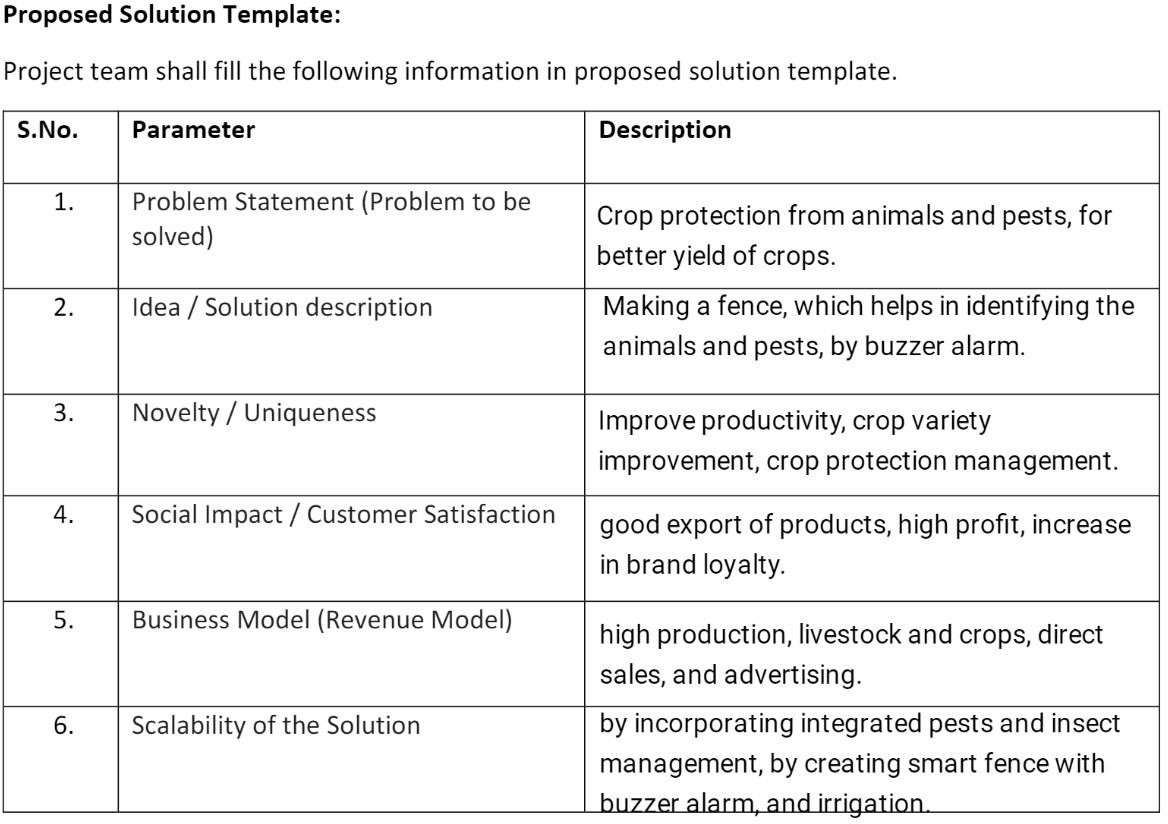
Our main purpose of the project is to develop intruder alert to the farm, to avoid losses due to animal and fire. These intruder alert protect the crop that damaging that indirectly increase yield of the crop. The develop system will not harmful and injurious to animal as well as human beings. Theme of project is to design a intelligent security system for farm protecting by using embedded system.

# LITERATURE SURVEY:

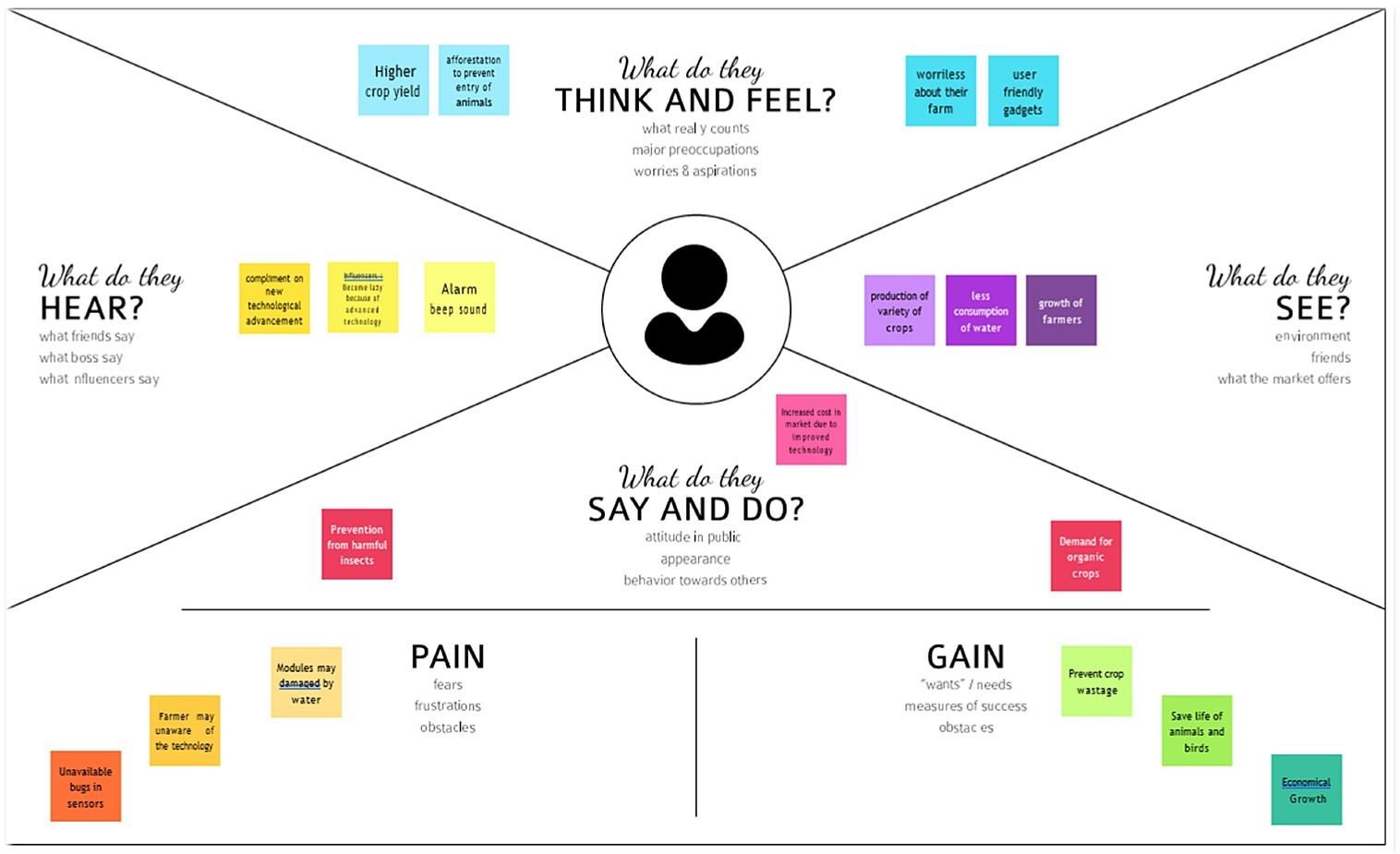
* 1. **EXISTING PROBLEM:**

The existing system mainly provide the surveillance functionality. Also these system don’t provide protection from wild animals, especially in such an application area. They also need to take actions based on the type of animal that tries to enter the area, as different methods are adopted to prevent different animals from entering restricted areas. The other commonly used method by farmer in order to prevent the crop vandalization by animals include building physical barriers, use of electric fences and manual surveillance and various such exhaustive and dangerous method. Sometimes if the weather/temperature changes suddenly it is necessary to take certain actions.

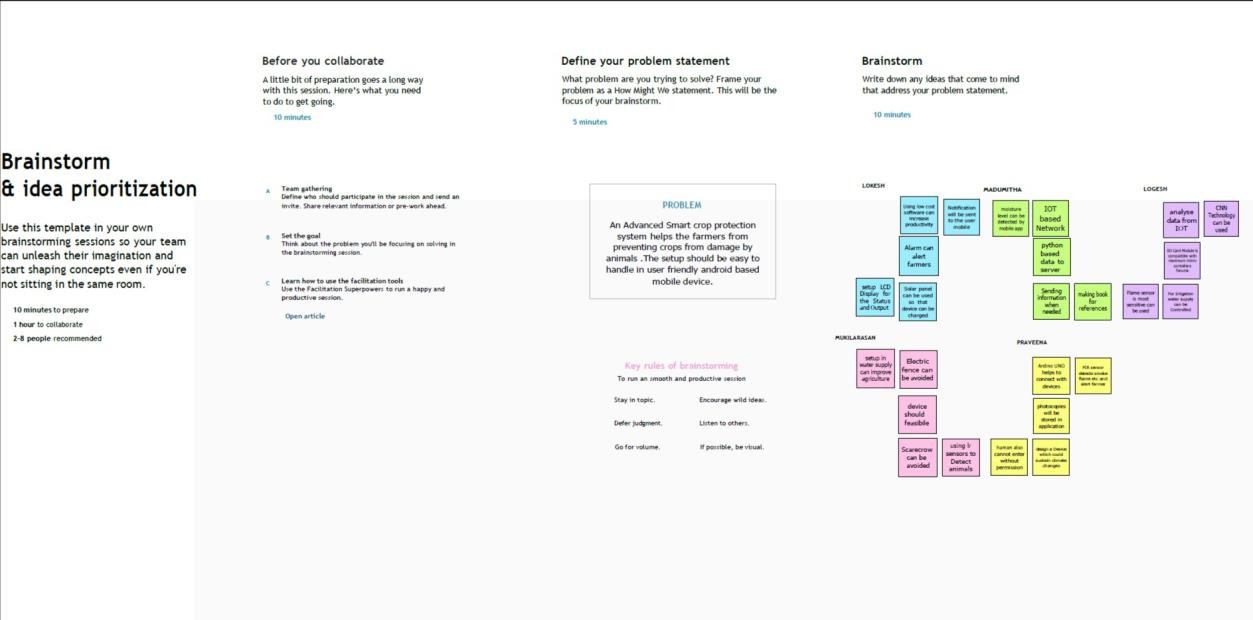
# PROPOSED SOLUTION



* Soil Moisture can be checked by using the sensors that can sense the soil condition and send the moisture content in the soil over the cloud services to the web application.
* The supply of water can be controlled from anywhere by controlling the motor state (ON/OFF), using web application.
* Surrounding temperature can also be sensed by the sensors and displayed on the application.
* Real time weather conditions can also be known by using different weather APIs from different websites and displayed on our application.



# BRAINSTORMING:



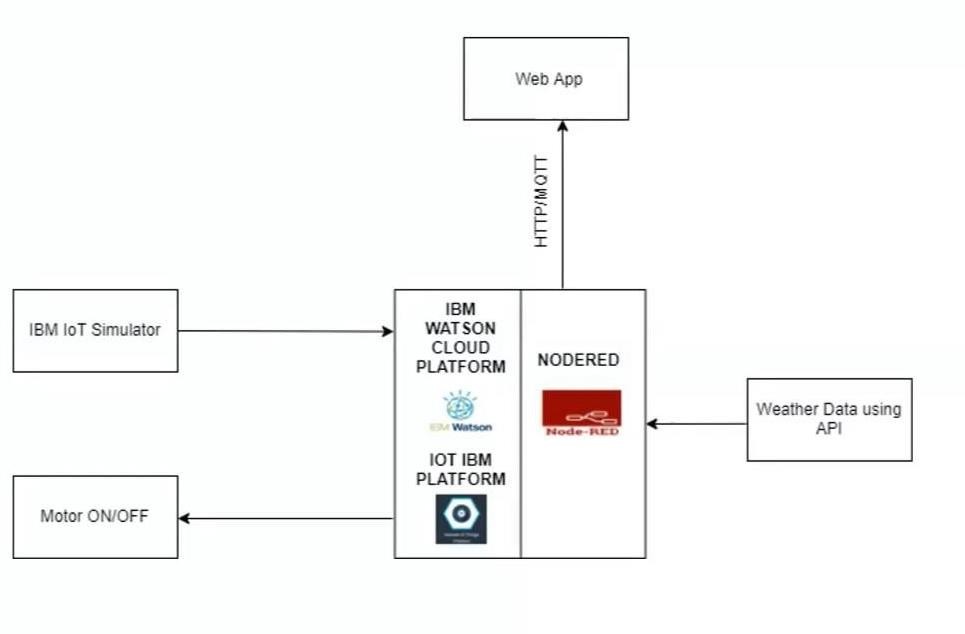
**DEFINITION STATEMENT:**

In the world economy of many Countries dependent upon the agriculture.

In spite of economic development agriculture is the backbone of the economy. Crops in forms are many times ravaged by local animals like buffaloes, cows, goats, birds and fire etc. this leads to huge loss for the farmers.it is not possible for farmers to blockade to entire fields or stay 24 hours and guard it. Agriculture meets food requirements of the people and produces several raw materials for industries. But because of animal interference and fire in agricultural lands, there will be huge loss of crops. Crops will be totally getting destroyed.

# THEORITICAL ANALYSIS:

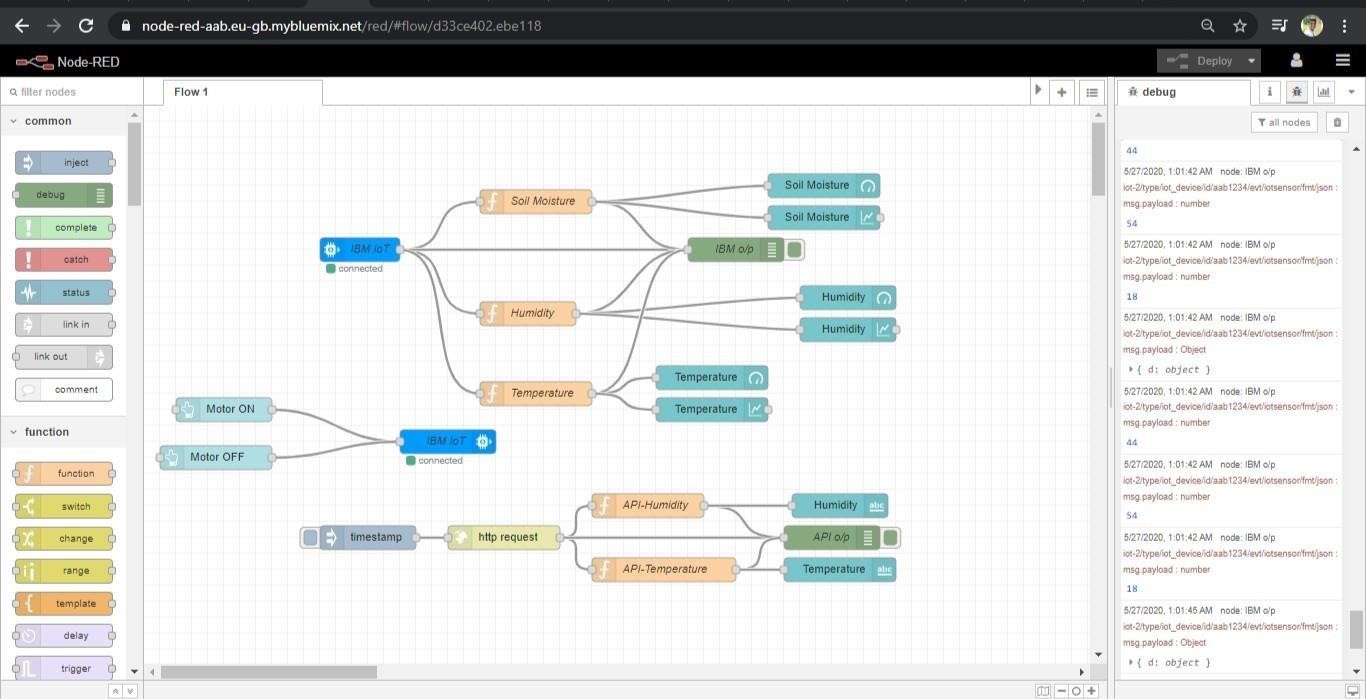
* 1. **BLOCK DIAGRAM**



# Hardware / Software Designing

1. Create a device in IBM Cloud.
2. Connect the device to IBM Simulator to get the weather conditions.
3. Build Node-RED flow to build a web application to display the weather conditions and control the devices.
4. Get the real time weather condition data from open weather map and integrate it in the Node-RED.
5. Control the working of the web application to the devices by python coding.

# 5 FLOWCHART:

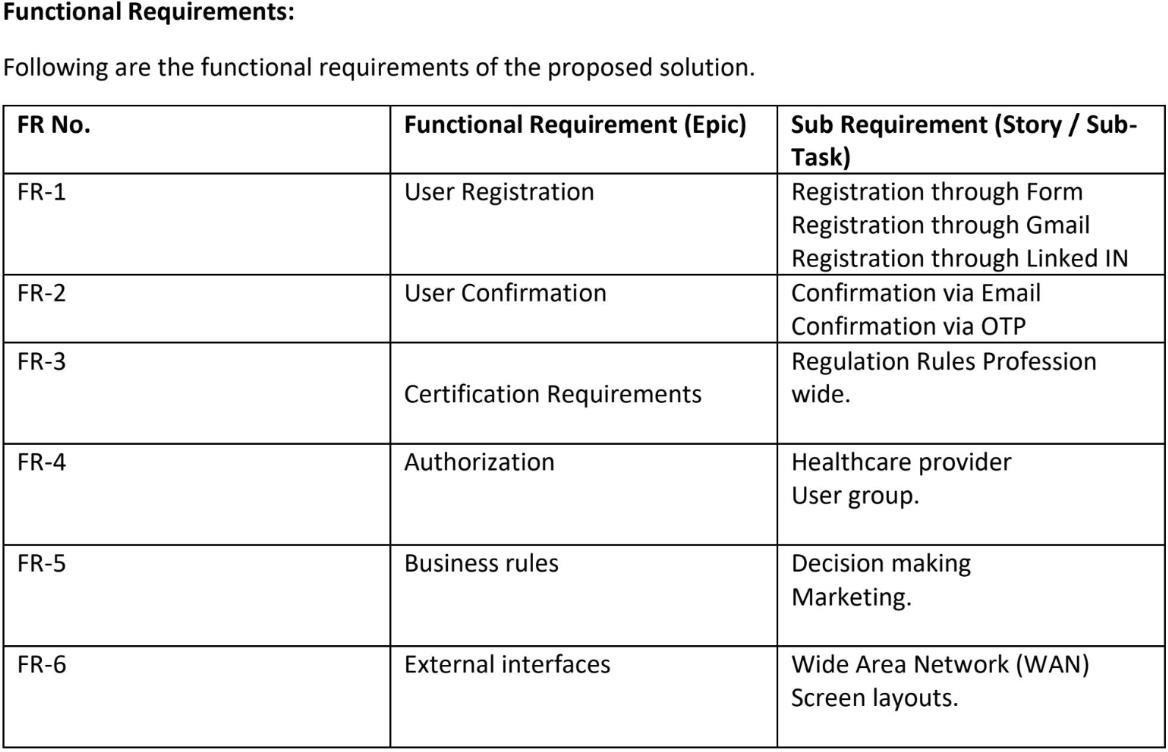


**Following are the nodes used in the project in the Web Application:**

1. IBM IoT: IN and OUT Nodes.
2. function Nodes.
3. Gauge Nodes.
4. Chart Nodes.
5. Debug Nodes.
6. Button Nodes.

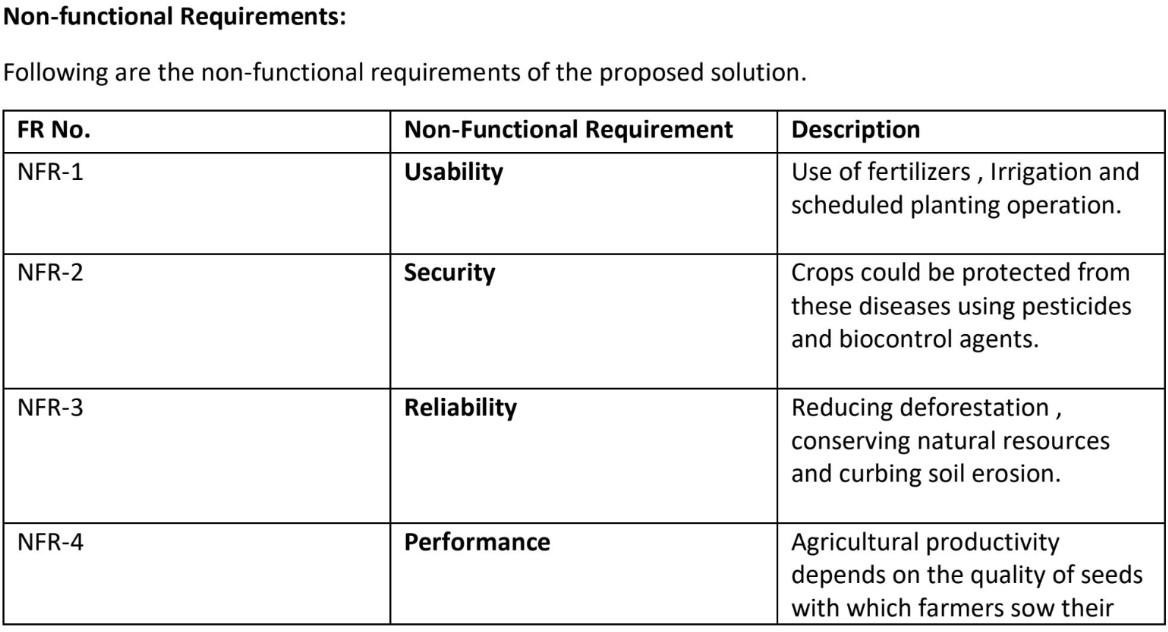
# REQUIREMENT ANALYSIS

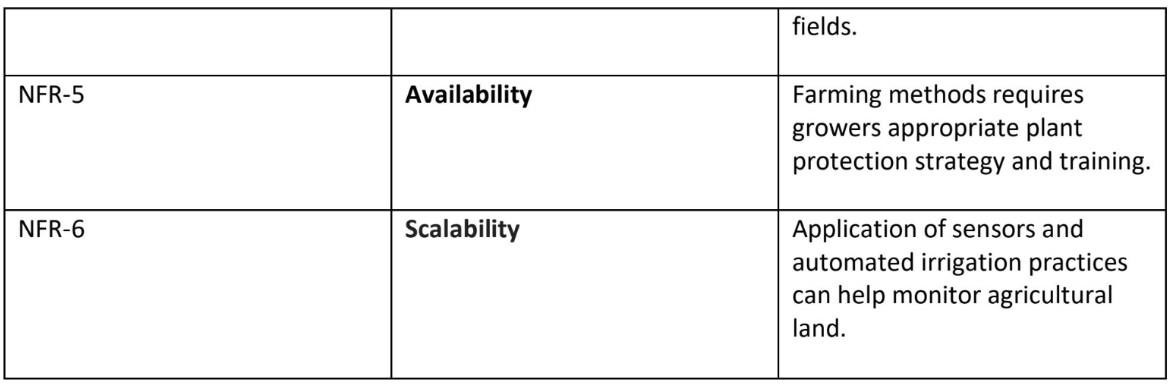
* 1. **FUNCTIONAL REQUIREMENT**



Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describe all the cases where the system uses the functional requirements, these are captured in use cases.

# NON-FUNCTIONAL REQUIREMENT





1. **ADVANTAGES & DISADVANTAGES:**

# ADVANTAGES

* + - All the data like climatic conditions and changes in them, soil or crop conditions everything can be easily monitored.
    - Risk of crop damage can be lowered to a greater extent.
    - Many difficult challenges can be avoided making the process automated and the quality of crops can be maintained.
    - The process included in farming can be controlled using the web applications from anywhere, anytime.
    - They are easy to operate and use and easy to maintain.
    - Sensors are cheaper in price and best in quality.

# DISADVANTAGES:

* + - Smart Agriculture requires internet connectivity continuously, but rural parts cannot fulfil this requirement.
    - Any faults in the sensors can cause great loss in the agriculture, due to wrong records and the actions of automated processes.
    - IoT devices need much money to implement.
      * High Cost: Equipment needed to implement IoT in agriculture is expensive.

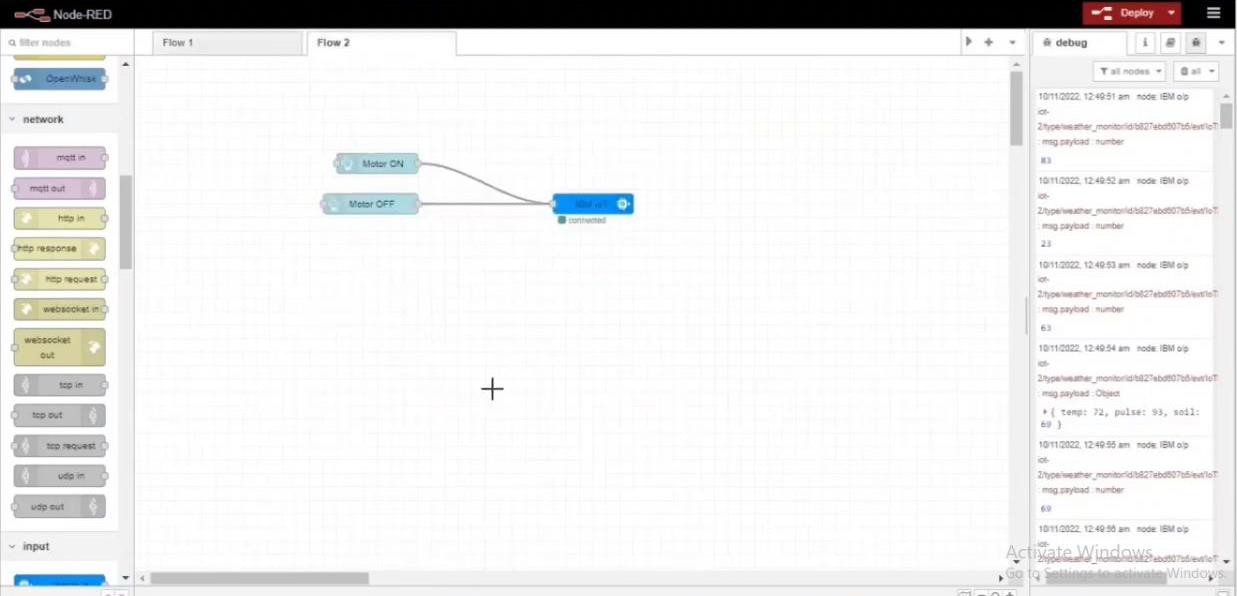
# 8 APPLICATIONS:

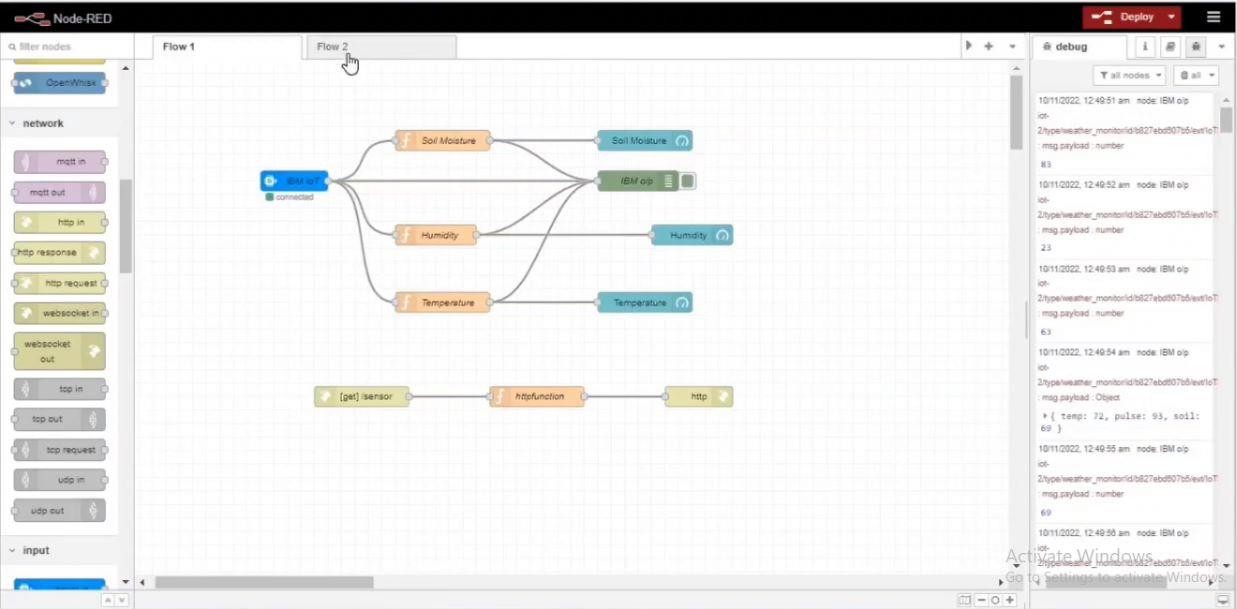
* Precision Farming that is farming processes can be made more controlled and accurate.
* Live monitoring can be done of all the processes and the conditions on the agricultural field.
* All the controls can be made just on the click.
* Quality can be maintained.

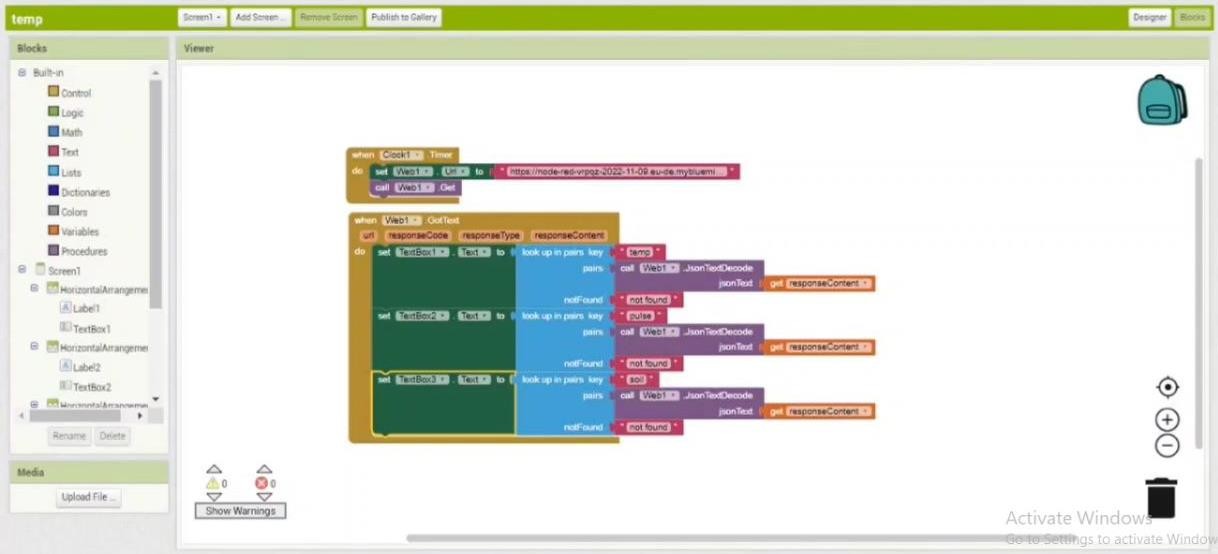
# RESULT

The problem of crop vandalization by wild animals and fire has become a major social problem in current time.

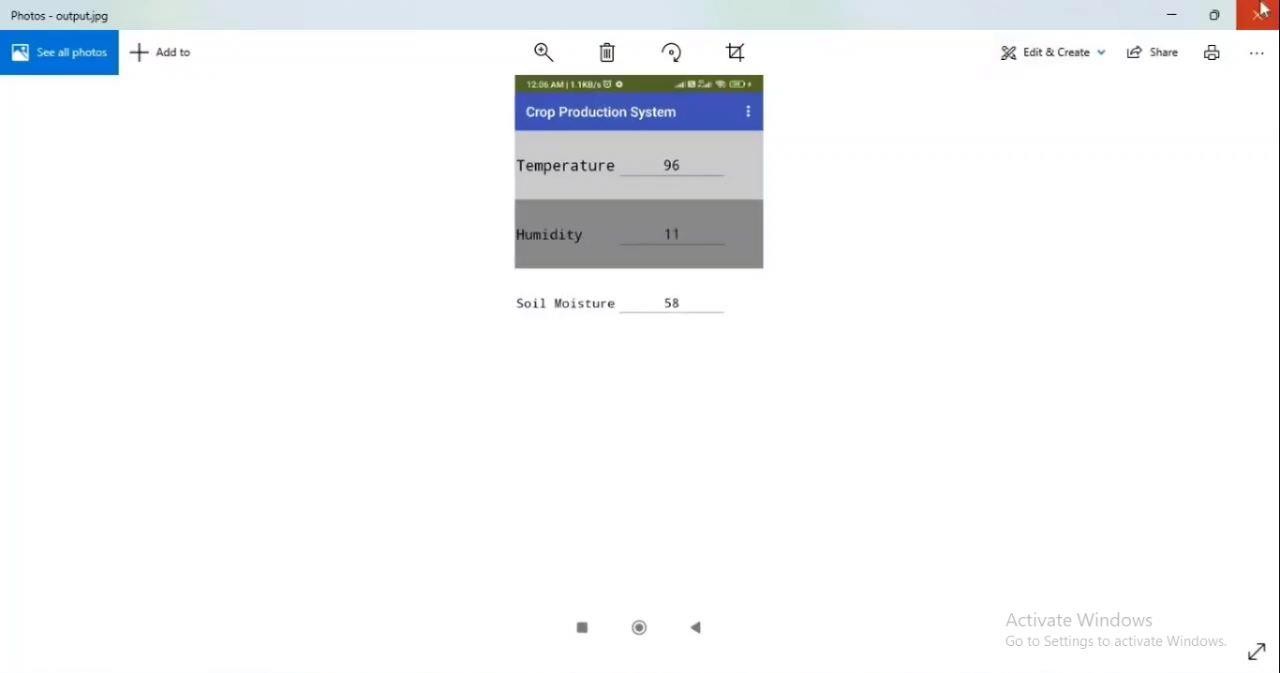
It requires urgent attention as no effective solution exists till date for this problem. Thus, this project carries a great social relevance as it aims to address this problem. This project will help farmers in protecting their orchards and fields and save them from significant financial losses and will save them from the unproductive efforts that they endure for the protection their fields. This will also help them in achieving better crop yields thus leading to their economic wellbeing.

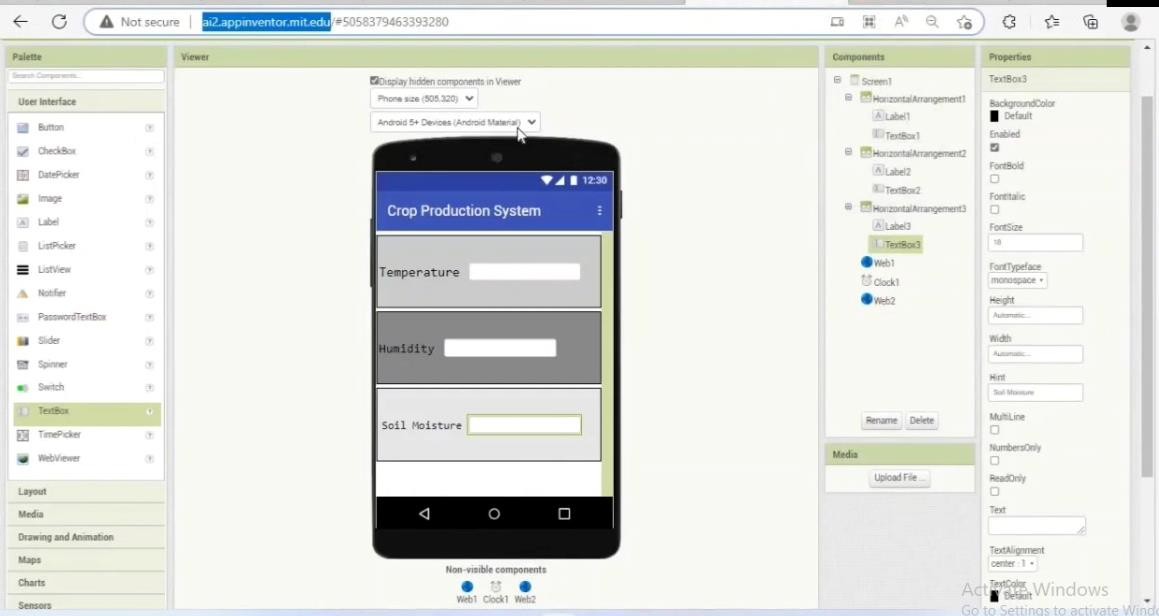


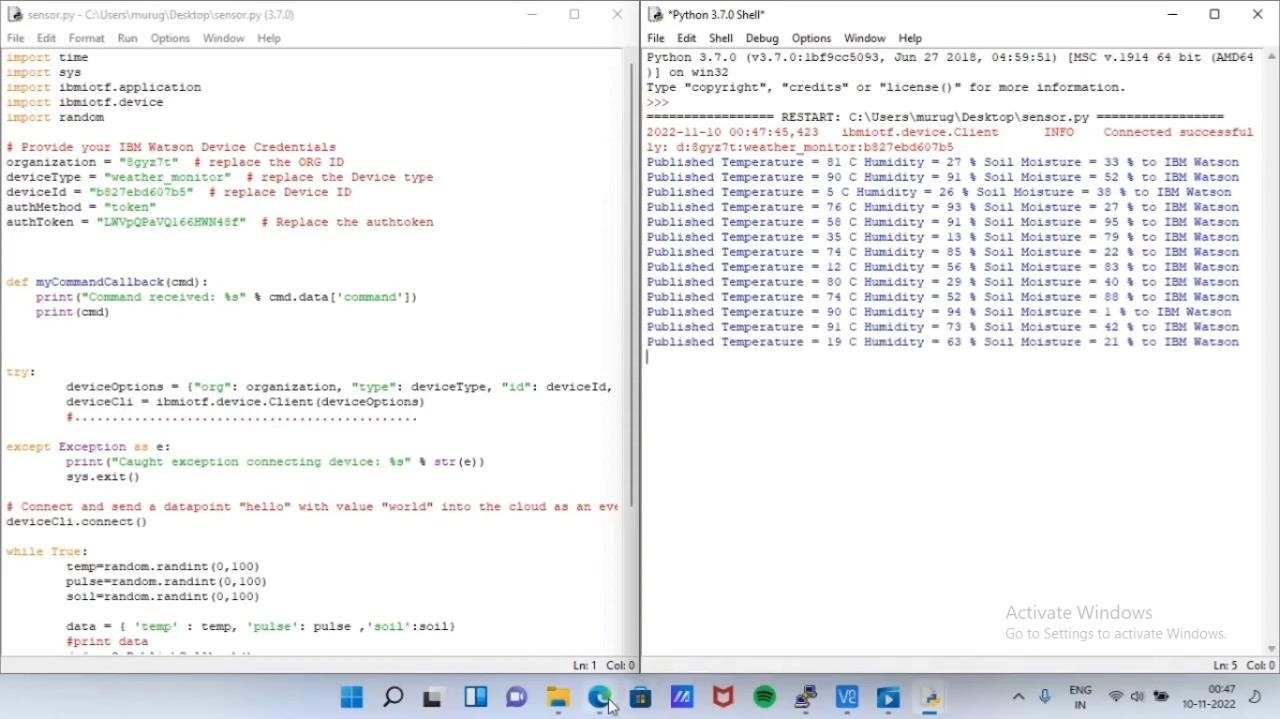




# INTERFACE:







1. **CONCLUSION**

A IoT Web Application is built for smart agricultural system using Watson IoT platform, Watson simulator, IBM cloud and Node- RED.

# FUTURE SCOPE

In future due to more demand of good and more farming in less time, for betterment of the crops and reducing the usage of extravagant resources like electricity and water IoT can be implemented in most of the places.

# BIBLIOGRAPHY

* + IBM Cloud:

[https://cloud.ibm.com/docs/overview?topic=overview-](https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform) [whatis-platform](https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform)

* + Watson IoT:

[https://www.iotone.com/software/ibm-watson-iot-](https://www.iotone.com/software/ibm-watson-iot-platform/s62) [platform/s62](https://www.iotone.com/software/ibm-watson-iot-platform/s62)

* + Node-RED:

[https://nodered.org/docs/getting-started/windows#3-run-](https://nodered.org/docs/getting-started/windows#3-run-node-red)

[node-red](https://nodered.org/docs/getting-started/windows#3-run-node-red)

<https://www.youtube.com/watch?v=cicTw4SEdxk>

* + Open weather map: <https://openweathermap.org/>

# APPENDIX

1. **SOURCE CODE**

## MOTOR.PY

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

# Provide your IBM Watson Device Credentials organization

= "8gyz7t" # replace the ORG ID deviceType = "weather\_monitor" # replace the Device type deviceId = "b827ebd607b5" # replace Device ID authMethod = "token" authToken = "LWVpQPaVQ166HWN48f" # Replace the authtoken

def myCommandCallback(cmd): # function for Callback if cmd.data['command'] == 'motoron':

print("MOTOR ON IS RECEIVED")

elif cmd.data['command'] == 'motoroff': 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, "authmethod": 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()

## SENSOR.PY

import time import sys

import ibmiotf.application

import ibmiotf.device import random

# Provide your IBM Watson Device Credentials organization

= "8gyz7t" # replace the ORG ID deviceType = "weather\_monitor" # replace the Device type deviceId = "b827ebd607b5" # replace Device ID authMethod = "token" authToken = "LWVpQPaVQ166HWN48f" # Replace the authtoken

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data['command']) print(cmd)

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:

temp=random.randint(0,100) pulse=random.randint(0,100)

soil=random.randint(0,100)

def

data = { 'temp' : temp, 'pulse': pulse ,'soil':soil} #print data

myOnPublishCallback()

:

print ("Published Temperature = %s C" % temp, "Humidity = %s %%"

% pulse,"Soil Moisture = %s %%" % soil,"to IBM Watson")

success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on\_publish=myOnPublishCallback) if not success: print("Not connected to IoTF")

time.sleep(1)

deviceCli.commandCallback = myCommandCallback

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

## Node-RED FLOW :

[

{ "id":"625574ead9839b34",

"type":"ibmiotout", "z":"630c8601c5ac3295",

"authentication":"apiKey", "apiKey":"ef745d48e395ccc0", "outputType":"cmd", "deviceId":"b827ebd607b5", "deviceType":"weather\_monitor", "eventCommandType":"data", "format":"json",

"data":"data", "qos":0, "name":"IBM IoT",

"service":"registered", "x":680,

"y":220,

"wires":[]

},

{

"id":"4cff18c3274cccc4", "type":"ui\_button", "z":"630c8601c5ac3295",

"name":"",

"group":"716e956.00eed6c", "order":2,

"width":"0",

"height":"0", "passthru":false, "label":"MotorON",

"tooltip":"",

"color":"",

"bgcolor":"",

"className":"",

"icon":"", "payload":"{\"command\":\"motoron\"}", "payloadType":"str",

"topic":"motoron",

"topicType":"str", "x":360,

"y":160, "wires":[["625574ead9839b34"]]},

{

"id":"659589baceb4e0b0", "type":"ui\_button", "z":"630c8601c5ac3295",

"name":"", "group":"716e956.00eed6c", "order":3,

"width":"0",

"height":"0", "passthru":true,

"label":"MotorOFF",

"tooltip":"",

"color":"",

"bgcolor":"",

"className":"",

"icon":"", "payload":"{\"command\":\"motoroff\"}", "payloadType":"str",

"topic":"motoroff",

"topicType":"str", "x":350,

"y":220, "wires":[["625574ead9839b34"]]},

{"id":"ef745d48e395ccc0", "type":"ibmiot", "name":"weather\_monitor", "keepalive":"60",

"serverName":"", "cleansession":true, "appId":"",

"shared":false},

{"id":"716e956.00eed6c",

"type":"ui\_group",

"name":"Form", "tab":"7e62365e.b7e6b8", "order":1,

"disp":true, "width":"6",

"collapse":false},

{"id":"7e62365e.b7e6b8",

"type":"ui\_tab",

"name":"contorl",

"icon":"dashboard", "order":1, "disabled":false, "hidden":false}

]

[

{

"id":"b42b5519fee73ee2", "type":"ibmiotin", "z":"03acb6ae05a0c712", "authentication":"apiKey", "apiKey":"ef745d48e395ccc0", "inputType":"evt", "logicalInterface":"", "ruleId":"", "deviceId":"b827ebd607b5", "applicationId":"",

"deviceType":"weather\_monitor", "eventType":"+",

"commandType":"",

"format":"json",

"name":"IBMIoT",

"service":"registered", "allDevices":"", "allApplications":"", "allDeviceTypes":"", "allLogicalInterfaces":"", "allEvents":true, "allCommands":"",

"allFormats":"", "qos":0,

"x":270,

"y":180,

"wires":[["50b13e02170d73fc","d7da6c2f5302ffaf","a949797028158f3f","a71f16 4bc3 78bcf1"]]

},

{ "id":"50b13e02170d73fc",

"type":"function", "z":"03acb6ae05a0c712", "name":"Soil Moisture",

"func":"msg.payload = msg.payload.soil;\nglobal.set('s',msg.payload);\nreturn msg;", "outputs":1,

"noerr":0, "initialize":"",

"finalize":"",

"libs":[], "x":490,

"y":120,

"wires":[["a949797028158f3f","ba98e701f55f04fe"]]

},

{

"id":"d7da6c2f5302ffaf", "type":"function", "z":"03acb6ae05a0c712", "name":"Humidity",

"func":"msg.payload = msg.payload.pulse;\nglobal.set('p',msg.payload)\nreturn msg;", "outputs":1,

"noerr":0, "initialize":"",

"finalize":"",

"libs":[], "x":480,

"y":260, "wires":[["a949797028158f3f","70a5b076eeb80b70"]]

},

{ "id":"a949797028158f3f",

"type":"debug", "z":"03acb6ae05a0c712", "name":"IBMo/p", "active":true, "tosidebar":true,

"console":false, "tostatus":false,

"complete":"payload", "targetType":"msg",

"statusVal":"",

"statusType":"auto", "x":780,

"y":180,

"wires":[]

},

{

"id":"70a5b076eeb80b70", "type":"ui\_gauge", "z":"03acb6ae05a0c712", "name":"", "group":"f4cb8513b95c98a4", "order":6,

"width":"0",

"height":"0",

"gtype":"gage",

"title":"Humidity",

"label":"Percentage(%)",

"format":"{{value}}", "min":0, "max":"100",

"colors":["#00b500","#e6e600","#ca3838"], "seg1":"", "seg2":"",

"className":"", "x":860,

"y":260,

"wires":[]

},

{

"id":"a71f164bc378bcf1", "type":"function", "z":"03acb6ae05a0c712", "name":"Temperature",

"func":"msg.payload=msg.payload.temp;\nglobal.set('t',msg.payload);\nreturn msg;", "outputs":1,

"noerr":0, "initialize":"",

"finalize":"",

"libs":[], "x":490,

"y":360,

"wires":[["8e8b63b110c5ec2d","a949797028158f3f"]]

},

{

"id":"8e8b63b110c5ec2d", "type":"ui\_gauge", "z":"03acb6ae05a0c712", "name":"", "group":"f4cb8513b95c98a4", "order":11,

"width":"0",

"height":"0",

"gtype":"gage", "title":"Temperature", "label":"DegreeCelcius", "format":"{{value}}", "min":0,

"max":"100", "colors":["#00b500","#e6e600","#ca3838"], "seg1":"",

"seg2":"",

"className":"", "x":790,

"y":360,

"wires":[]

},

{

"id":"ba98e701f55f04fe", "type":"ui\_gauge", "z":"03acb6ae05a0c712", "name":"", "group":"f4cb8513b95c98a4", "order":1,

"width":"0",

"height":"0",

"gtype":"gage", "title":"Soil Moisture", "label":"Percentage(%)",

"format":"{{value}}", "min":0, "max":"100",

"colors":["#00b500","#e6e600","#ca3838"], "seg1":"",

"seg2":"",

"className":"",

"x":790,

"y":120,

"wires":[]

},

{

"id":"a259673baf5f0f98", "type":"httpin", "z":"03acb6ae05a0c712", "name":"",

"url":"/sensor",

"method":"get", "upload":false, "swaggerDoc":"", "x":370,

"y":500,

"wires":[["18a8cdbf7943d27a"]]

},

{

"id":"18a8cdbf7943d27a", "type":"function", "z":"03acb6ae05a0c712", "name":"httpfunction",

"func":"msg.payload{\"pulse\":global.get('p'),\"temp\":global.get('t'),\"soil\":glob al.get( 's')};\nreturn msg;",

"outputs":1,

"noerr":0, "initialize":"",

"finalize":"",

"libs":[], "x":630,

"y":500, "wires":[["5c7996d53a445412"]]

},

{ "id":"5c7996d53a445412",

"type":"httpresponse", "z":"03acb6ae05a0c712", "name":"",

"statusCode":"",

"headers":{}, "x":870,

"y":500,

"wires":[]

},

{

"id":"ef745d48e395ccc0", "type":"ibmiot", "name":"weather\_monitor", "keepalive":"60",

"serverName":"", "cleansession":true, "appId":"",

"shared":false},

{

"id":"f4cb8513b95c98a4", "type":"ui\_group",

"name":"monitor", "tab":"1f4cb829.2fdee8", "order":2,

"disp":true, "width":"6", "collapse":false, "className":""

},

{

"id":"1f4cb829.2fdee8",

"type":"ui\_tab",

"name":"Home",

"icon":"dashboard", "order":3, "disabled":false, "hidden":false }

Github link of our project

# 

# [IBM-EPBL/IBM-Project-10200-1659112865: IoT Based Smart Crop Protection System for Agriculture (github.com)](https://github.com/IBM-EPBL/IBM-Project-10200-1659112865)

# 

# Youtube link of our video:

# 

# <https://youtu.be/Oeap2zWzidc>

# Thanks for giving this opportunity

# 