IOT ENABLED SMART FARMER SMART FARMING APPLICATION.

Sprint Delivery – 1

TEAMID : PNT2022TMID40009

1. Introduction

The main aim of this project is to help farmers automate their farms by providing them with a Web App through which they can monitor the parameters of the field like Temperature, soil moisture, humidity and etc and control the equipment like water motor and other devices remotely via internet without their actual presence in the field.

2. Problem Statement

Farmers are to be present at farm for its maintenance irrespective of the weather conditions. They have to ensure that the crops are well watered and the farm status is monitored by them physically. Farmer have to stay most of the time in field in order to get a good yield. In difficult times like in the presence of pandemic also they have to work hard in their fields risking their lives to provide food for the country.

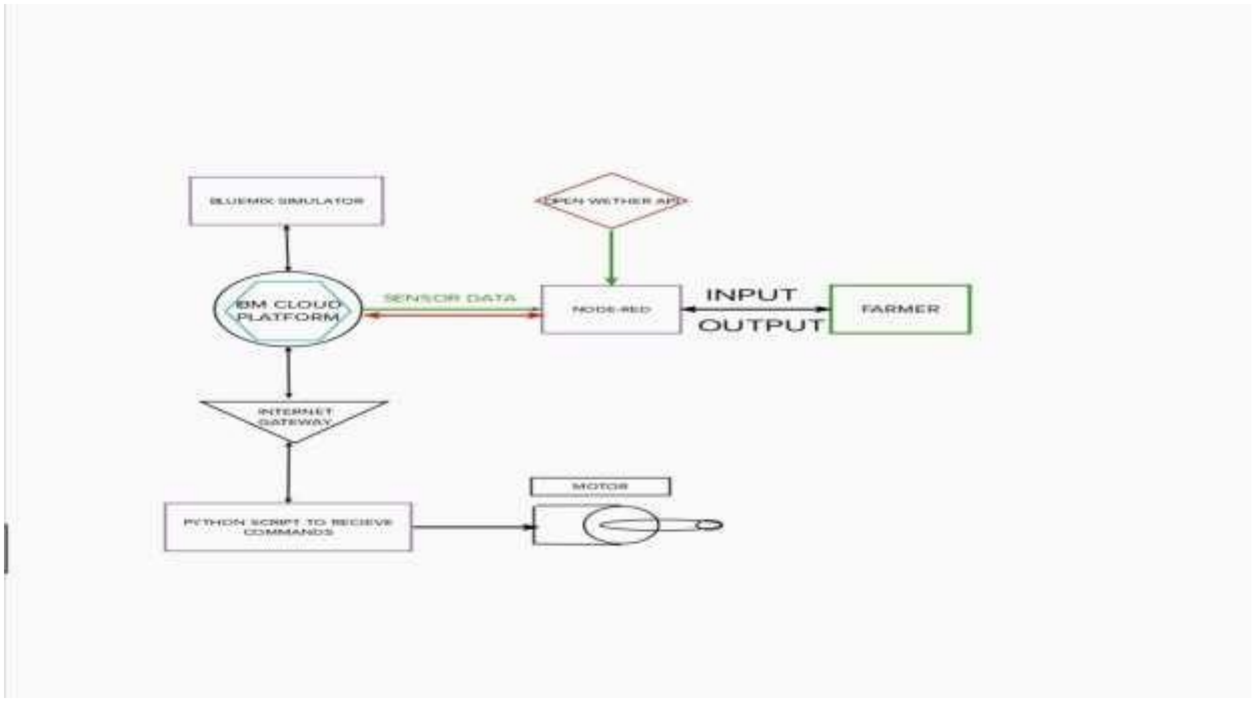
3. Proposed Solution

In order to improve the farmer's working conditions and make them easier, we introduce IoT services to him in which we use cloud services and internet to enable farmer to continue his work remotely via internet. He can monitor the field parameters and control the devices in farm.

4. Theoretical Analysis

4.1 Block Diagram

In order to implement the solution , the following approach as shown in the block diagram is used



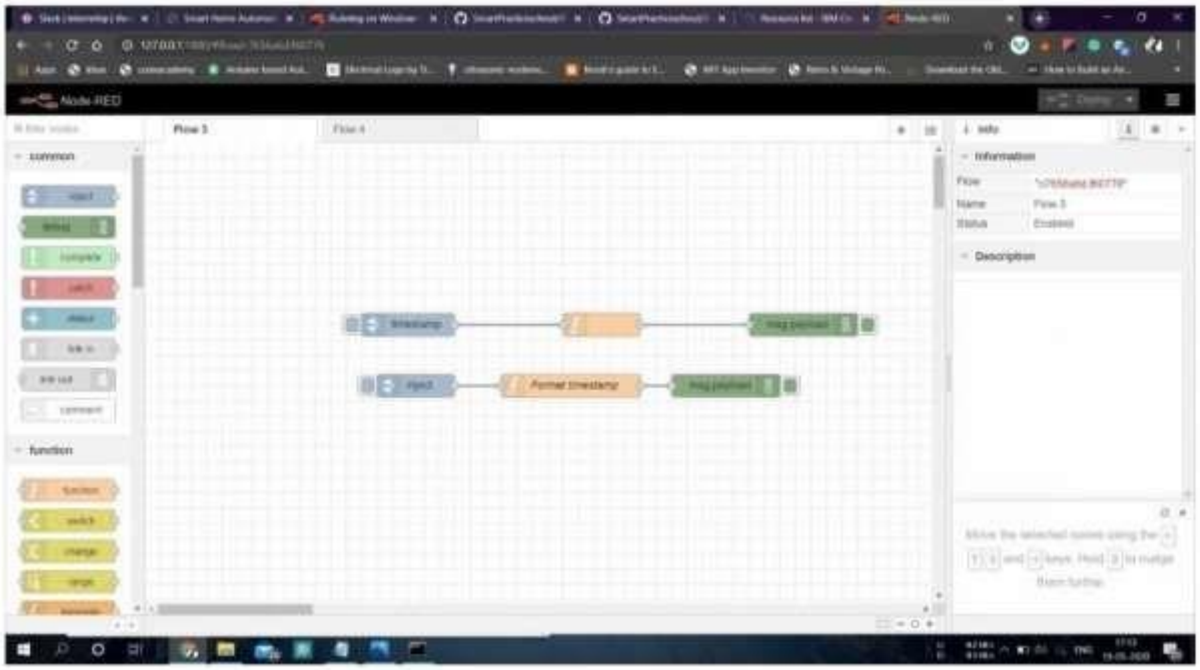
4.2 Required Software Installation

4.2.A Node-Red

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.

Installation :

- First install npm/node.js
- Open cmd prompt
- Type => npm install node-red



To run the application :

- Open cmd prompt
- Type=>node-red
- Then open http://localhost:1880/ in browser

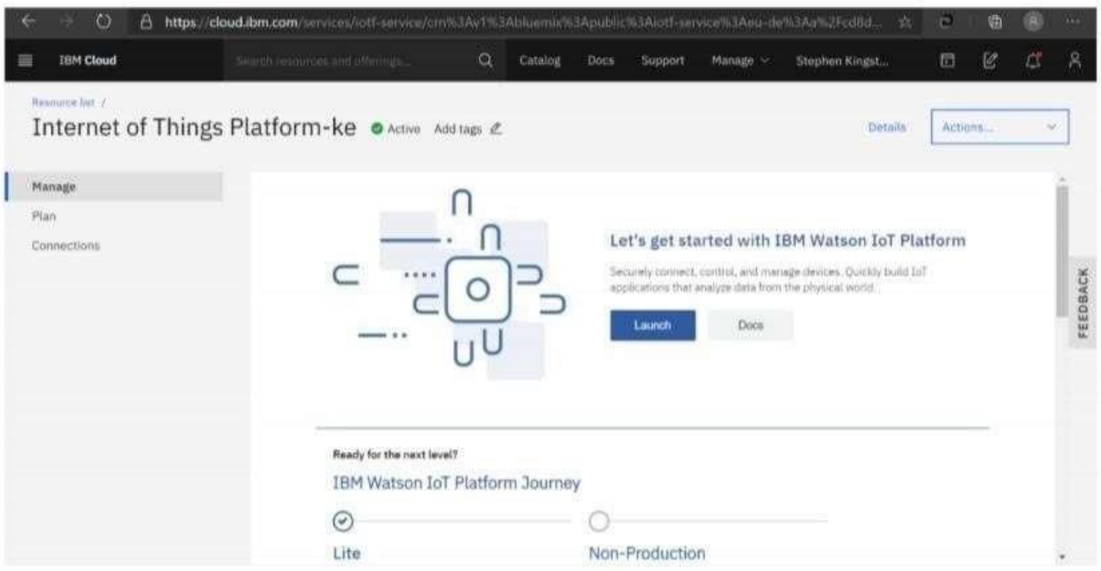
Installation of IBM IoT and Dashboard nodes for Node-Red

In order to connect to IBM Watson IoT platform and create the Web App UI these nodes are required

1. IBM IoT node
2. Dashboard node

4.2.B 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.


Internet of Things Platform-ke

Let's get started with IBM Watson IoT Platform

Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.

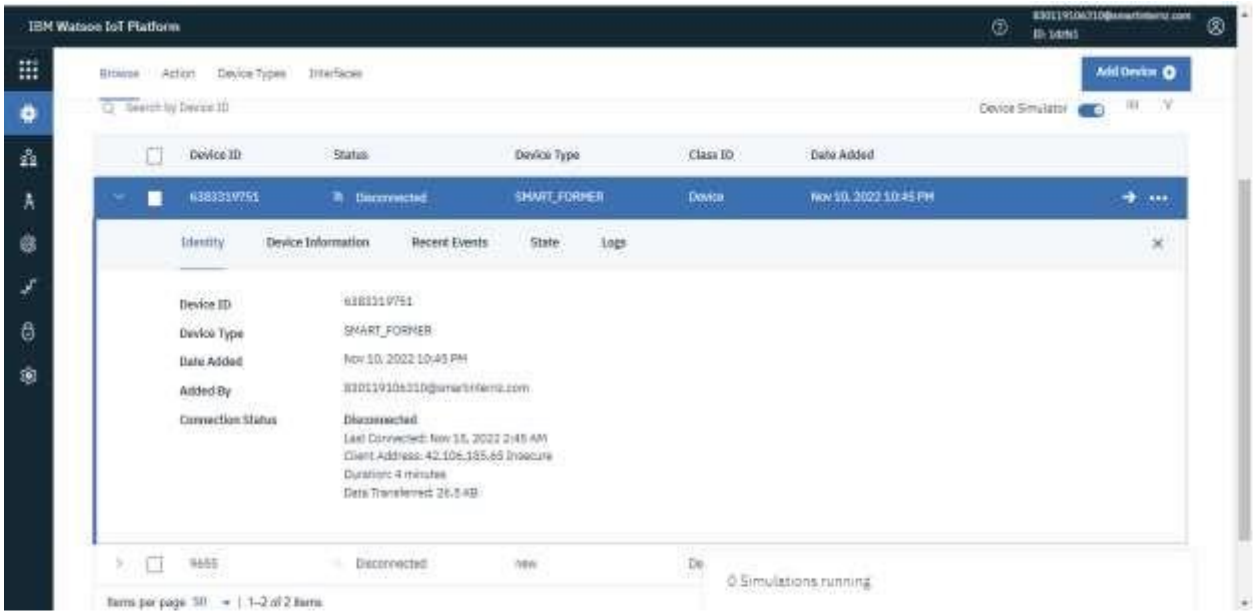
Ready for the next level?

IBM Watson IoT Platform Journey

Lite — Non-Production

Steps to configure:

- Create an account in IBM cloud using your email ID
- Create IBM Watson Platform in services in your IBM cloud account
- Launch the IBM Watson IoT Platform
- Create a new device
- Give credentials like device type, device ID, Auth. Token
- Create API key and store API key and token elsewhere.


IBM Watson IoT Platform

Device ID	Status	Device Type	Class ID	Date Added
6383319751	Disconnected	SMART_FORMER	Device	Nov 10, 2022 10:45 PM
9655	Disconnected	new	De	

Device ID	6383319751
Device Type	SMART_FORMER
Date Added	Nov 10, 2022 10:45 PM
Added By	830119106310@smartinternz.com
Connection Status	Disconnected Last Connected: Nov 15, 2022 2:45 AM Client Address: 42.106.185.65 Insecure Duration: 4 minutes Data Transferred: 26.5 KB

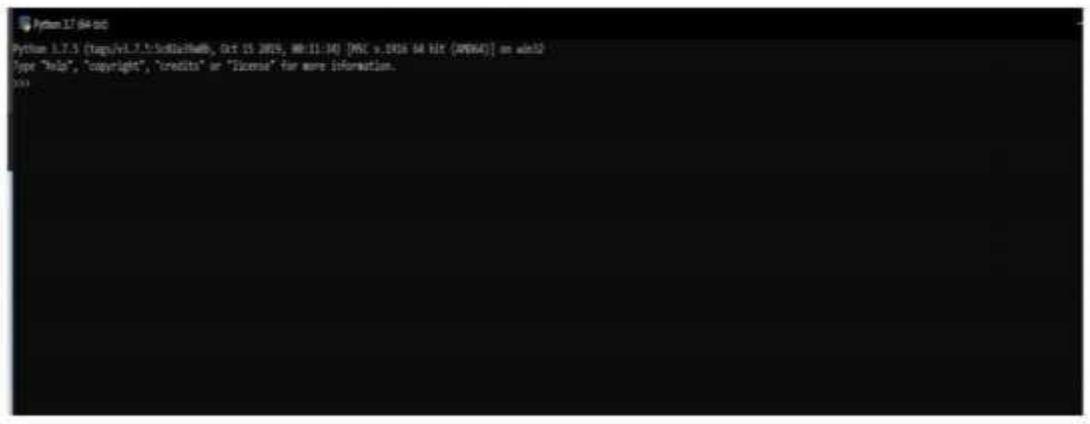
0 Simulations running

Items per page 50 | 1–2 of 2 items

4.2.C Python IDE

Install Python3 compiler

Install any python IDE to execute python scripts, in my case I used Spyder to execute the code.



```
Python 3.7.5 (tags/v3.7.5:5c02a39a0b, Oct 15 2019, 00:11:34) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Code: import time import

```
sys import
ibmiotf.application import
ibmiotf.device import
random
#Provide your IBM Watson Device
Credentials organization = "157uf3"
deviceType = "abcd" deviceId = "7654321"
authMethod = "token" authToken =
"87654321"


# Initialize GP 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, "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:
     #Get Sensor Data from DHT11
     temp=random.randint(90,110)
     Humid=random.randint(60,100)
     Mois=random.randint(20,120)

     data = { 'temp' : temp, 'Humid': Humid, 'Mois'
     :Mois} #print data   def
myOnPublishCallback():
print ("Published Temperature =
%s C" % temp, "Humidity = %s
%%" % Humid, "Moisture
=%s deg c" %Mois, "to IBM
Watson") success =     deviceCli.publishEvent("IoTSensor",  "json",     data,
      qos=0, on_publish=myOnPublishCallback)     if not success:   print("Not
connected to IoTF") time.sleep(10)

     deviceCli.commandCallback = myCommandCallback
```

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

Aurdino code for C :

```
//#include <ESP8266WiFi.h>
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT11   // define type of sensor DHT 11
#define LED 2
#define pot 34
#define swat 27
#define twat 14
#define pone 12
#define ptwo 13
DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht connected
int moister = 0; int swatv = 0; int twatv = 0; int ponev = 0; int ptwov = 0;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength); //-------credentials
of IBM Accounts------

#define ORG "1dzfs1"//IBM ORGANITION ID
#define DEVICE_TYPE "SMART_FORMER"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "6383319751"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "DQIhkT2xKA-Xk*Ztau"     //Token
String data3;
float h, t;
//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which
data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT command type AND
COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method char
token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id char
ssid[] = "MY WIFI";
char pass[] = "96559655";

//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback , wifiClient); //calling the predefined client id by passing
parameter like server id,portand wificredential
```

```
void setup()// configureing the ESP32 {
  Serial.begin(115200); dht.begin();
  pinMode(LED, OUTPUT);
pinMode(pot, INPUT);   pinMode(swat,
INPUT);   pinMode(twat, INPUT);
pinMode(pone, INPUT);
pinMode(ptwo, INPUT);
  delay(10);
Serial.println();
wificonnect();
mqttconnect();
}

void loop()// Recursive Function
{   moister =
analogRead(pot);   swatv =
digitalRead(swat);   twatv =
digitalRead(twat);   ponev =
digitalRead(pone);
  ptwov = digitalRead(ptwo);

  h = dht.readHumidity();   t
= dht.readTemperature();
  Serial.print("temp:");
  Serial.println(t);
  Serial.print("Humid:");
  Serial.println(h);

  PublishData(t, h, moister);
delay(1000);   if
(!client.loop()) {
    mqttconnect();
  } } void PublishData(float temp, float humid, float
moister) {
  mqttconnect();;//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"temp\":";
payload += temp;   payload +=
"," "\"Humid\":";   payload +=
humid;   payload += ","
"\"moister\":";   payload +=
moister;   payload += ","
"\"swat\":";   payload +=
swatv;   payload += ","
"\"twat\":";   payload += twatv;
payload += "," "\"pone\":";
payload += ponev; payload +=
"," "\"ptwo\":";
  payload += ptwov;
payload += "}";
  Serial.print("Sending payload: ");
Serial.println(payload);
  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish ok in
Serial monitor or else it will print publish failed
  } else {
    Serial.println("Publish failed");
  }}
void mqttconnect() {   if
(!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
delay(500);
    }

    initManagedDevice();
    Serial.println();
  }}
void wificonnect() //function defination for wificonnect {
  Serial.println();
  Serial.print("Connecting to ");

  WiFi.begin(ssid, pass);//passing the wifi credentials to establish the connection
while (WiFi.status() != WL_CONNECTED) {     delay(500);
Serial.print(".");   }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }}
```

```
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength) {

  Serial.print("callback invoked for topic: ");
  Serial.println(subscribetopic); for (int i
= 0; i < payloadLength; i++) {
//Serial.print((char)payload[i]);
    data3 += (char)payload[i];
  }
  Serial.println("data: " + data3);
  if (data3 == "motoron")
  {
    Serial.println(data3);
    digitalWrite(LED, HIGH);
  }
else
  {
    Serial.println(data3);
    digitalWrite(LED, LOW);
  }   data3 =
"";
        }
```

4.3 IoT Simulator

In our project in the place of sensors we are going to use IoT sensor simulator which give random readings to the connected cloud.

The link to simulator:

https://watson-iot-sensor-simulator.mybluemix.net/

We need to give the credentials of the created device in IBM Watson IoT Platform to connect cloud to simulator.

4.4 OpenWeather API

OpenWeatherMap is an online service that provides weather data. It provides current weather data, forecasts and historical data to more than 2 million customer.

Website link: https://openweathermap.org/guide **Steps to**

configure:

o Create account in OpenWeather o Find the name of your city by searching o Create API key to your account

o Replace “city name” and “your api key” with your city and API key in below red text

api.openweathermap.org/data/2.5/weather?q={city name}&appid={your api key}