IOT ENABLED SMART FARMING APPLICATION

Sprint Delivery – 1

TEAMID :PNT2022TMID18018

1. Introduction:

The main goal of this project is to assist farmers in automating their farms by giving them access to a Web App that allows them to remotely control equipment like water motors and other devices without being physically present in the field while also monitoring field parameters like temperature, soil moisture, and humidity.

2. Problem Statement:

Regardless of the weather, farmers must be on site to maintain their farms. They need to physically check on the farm's condition and make sure the crops are getting enough water. To get a good yield, the farmer must spend the majority of his time on the field. They must toil in their fields and risk their lives to provide food for the nation during difficult times, such as when a pandemic is present.

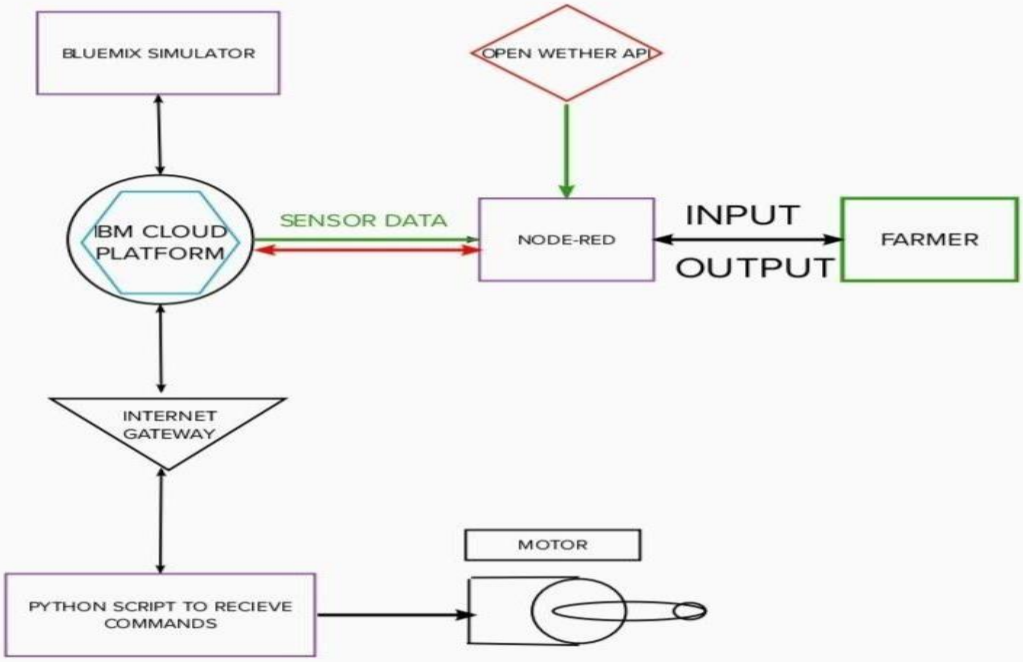
3. Proposed Solution:

We introduce IoT services to the farmer in order to enhance and facilitate his working conditions. These services make it possible for the farmer to work remotely using the internet and cloud services. He has the ability to manage farm equipment and monitor field parameters.

4. Theoretical Analysis

4.1 Block Diagram:

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

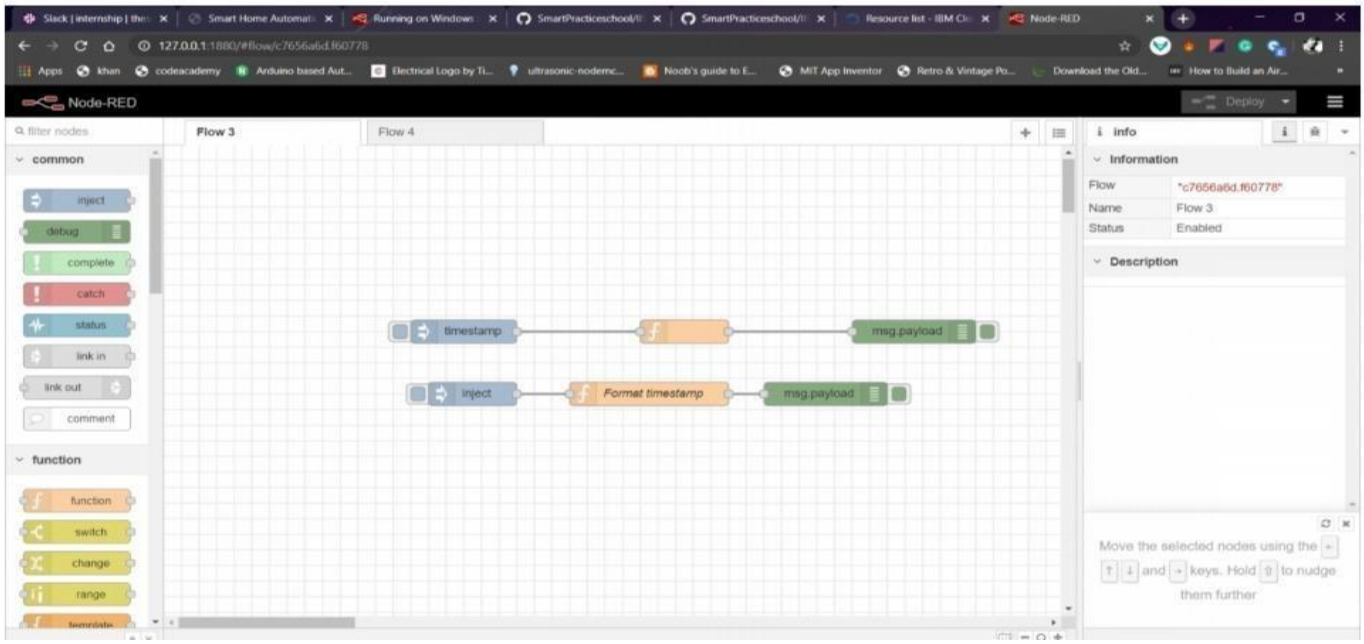


4.2 Required Software Installation

4.2 A Node-Red

Node-RED is a flow-based visual programming tool created initially by IBM for connecting physical components, APIs, and web applications such

One of the Things on the Internet. An online flow editor powered by Node-RED is available for developing 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

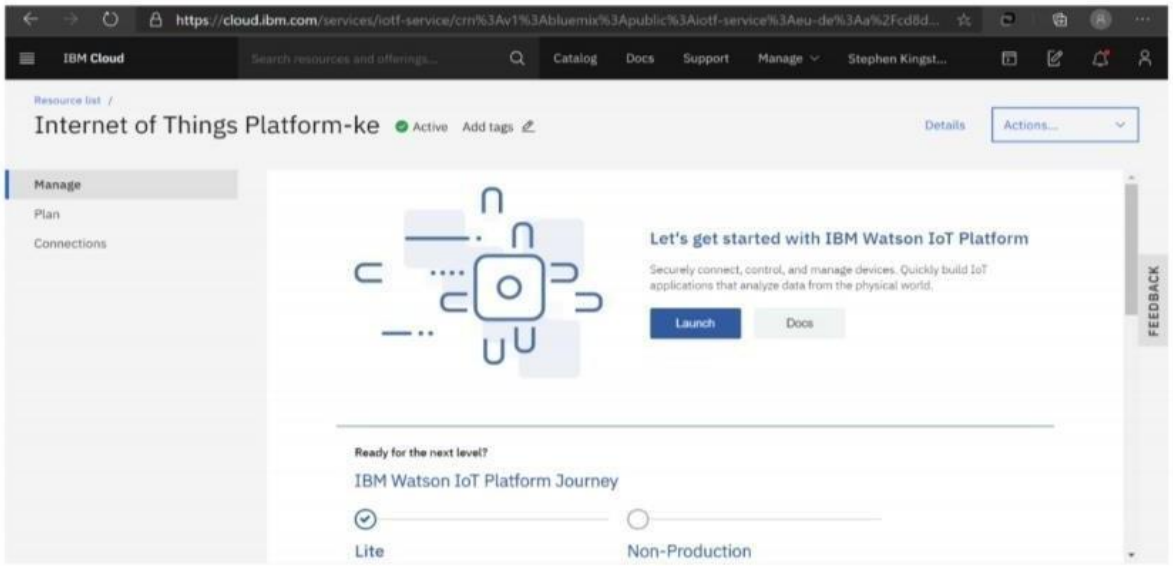
Installation of IBM IoT and Dashboard nodes for Node-Red

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

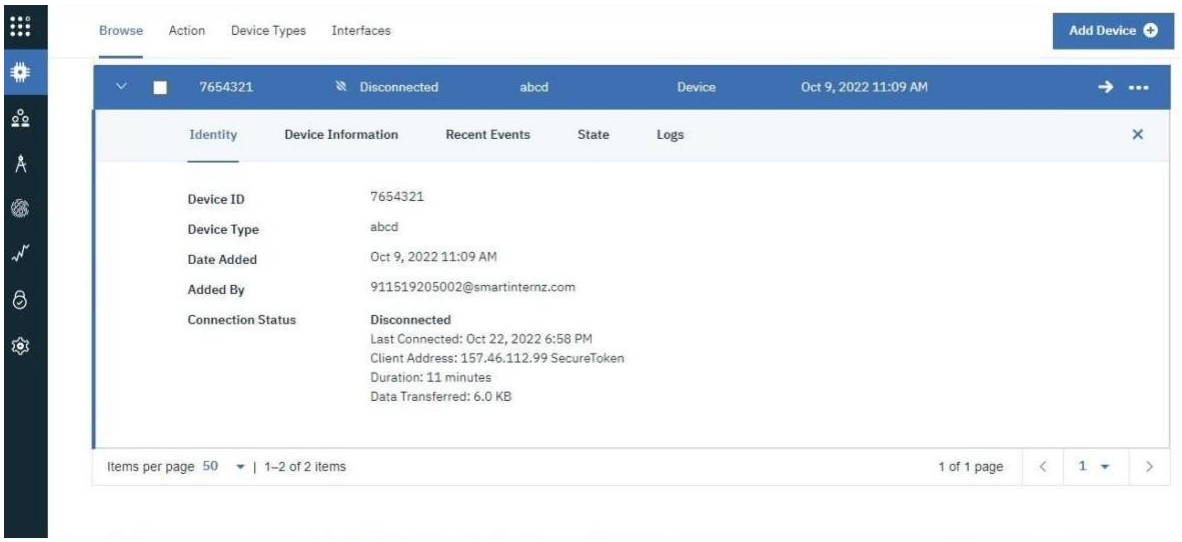
4.2.1 IBM Watson IoT Platform

A fully managed, cloud-hosted solution that has data storage, communication, control, and device registration features. The managed, cloud-hosted IBM Watson IoT Platform is a solution that makes it easy to get value out of your IoT devices.



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.



4.2.2 Python IDE

Install Python3 compiler

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

Code: import time

import sys import

ibmiotf.application import

ibmiotf.device import

random

#Provide your IBM Watson Device

Credentialsorganization = "157uf3"

deviceType = "abcd" deviceId = "7654321"

authMethod = "token" authToken =

"87654321"

Initialize GPIO

```
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
=%sdeg 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()
```

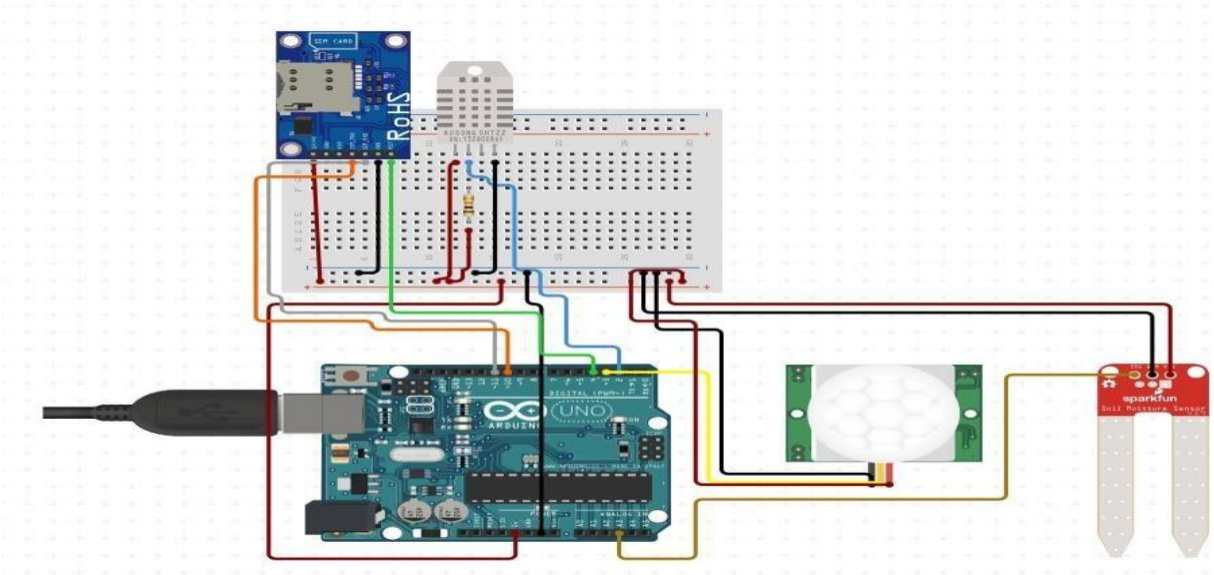
Aurdino code for C :

```
//include libraries
 #include <dht.h>
 #include <SoftwareSerial.h>
```

```
//define pins
#define dht_apin A0 // Analog Pin sensor is connectedSoftwareSerial
mySerial(7,8);//serial port of gsm
const int sensor_pin = A1; // Soil moisture sensor O/P pinint
pin_out = 9;
//allocate variables
dht DHT;
int c=0;

void setup()
{
pinMode(2, INPUT); //Pin 2 as INPUT
pinMode(3, OUTPUT); //PIN 3 as
OUTPUTpinMode(9, OUTPUT);//output
for pump
}
void loop()
{
 if (digitalRead(2) == HIGH)
 {
 digitalWrite(3, HIGH); // turn the LED/Buzz ON
 delay(10000); // wait for 100 msecond digitalWrite(3,
 LOW);  // turn the LED/Buzz OFFdelay(100);
 }
  Serial.begin(9600);
    delay(1000);
   DHT.read11(dht_apin); //temprature
 float h=DHT.humidity;
 float t=DHT.temperature;
   delay(5000);
   Serial.begin(9600);
    float moisture_percentage;//moistureint
 sensor_analog;
 sensor_analog = analogRead(sensor_pin);
 moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
```

```
  float m=moisture_percentage;
  delay(1000);
  if(m<40)//pump
  {
  while(m<40)
  {
  digitalWrite(pin_out,HIGH);//open pump
  sensor_analog = analogRead(sensor_pin);
  moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
  m=moisture_percentage;
  delay(1000);
  }
  digitalWrite(pin_out,LOW);//closepump
  }
  if(c>=0)
  {
  mySerial.begin(9600);
  delay(15000);
  Serial.begin(9600);
  delay(1000); Serial.print("\r");
  delay(1000);
  Serial.print("AT+CMGF=1\r"
  );delay(1000);
  Serial.print("AT+CMGS=\"+XXXXXXXXXX\"\r"); //replace X with 10 digit
mobile number
  delay(1000);
  Serial.print((String)"update-
>"+(String)"Temprature="+t+(String)"Humidity="+h+(String)"Moisture="+m);
  delay(1000);
  Serial.write(0x1A);
  delay(1000);
  mySerial.println("AT+CMGF=1");//Sets the GSM Module in Text
  Modedelay(1000);
```

```
  mySerial.println("AT+CMGS=\"+XXXXXXXXXX\"\r"); //replace X with 10
digitmobile number
  delay(1000);
  mySerial.println((String)"update-
>"+(String)"Temprature="+t+(String)"Humidity="+h+(String)"Moisture="+m);//
message format
  mySerial.println();
  delay(100);
  Serial.write(0x1A);
  delay(1000);
  c++;

    }


}
```



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 Platformto 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

Stepsto configure:

- Create account in OpenWeather o Find the name of

your city by searching o Create API key to your account

- 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}