

# **IOT ENABLED SMART FARMING APPLICATION.**

Sprint Delivery – 1

TEAMID : **PNT2022TMID34723**

## **Introduction**

Most important factors for the quality and productivity of plant growth are temperature, humidity and light. Continuous monitoring of these environmental variables provides valuable information to the grower to better understand, how each factor affects growth and how to maximize crop productivity. WSN composed of hundreds of nodes which have ability of sensing, actuation and communicating, has great advantages in terms of high accuracy, fault tolerance, flexibility, cost, autonomy and robustness compared to wired ones. IOT is a general term, covering a number of technologies that allows devices to communicate with each other, with or without human intervention. The wireless sensor network (WSN) is one of the most significant technologies in the 21st century and they are very suitable for distributed data collecting and monitoring in tough environments such as greenhouses. The other most significant technologies in the 21st century is the Internet of Things (IOT) which has rapidly developed covering hundreds of applications in the civil, health, military and agriculture areas.

## **Problem Statement**

Develop affordable app-based solution for soil health monitoring and suggest which crop to be sown based on it.

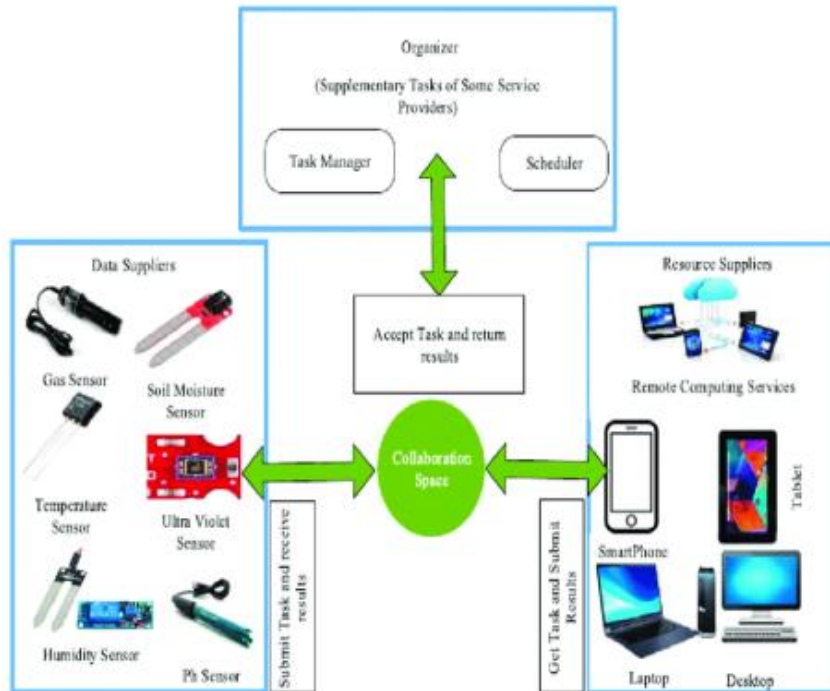
## **Proposed Solution**

The system can also help avoid over irrigation, thus protecting crops from diseases, saving water, electricity, predict early onset of diseases and other advisories. In case of erratic electricity farmers can switch on the irrigation equipment (sprinklers, drip, pumps) with the leaching of nutrients from soil.

## Theoretical Analysis

### Block Diagram

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

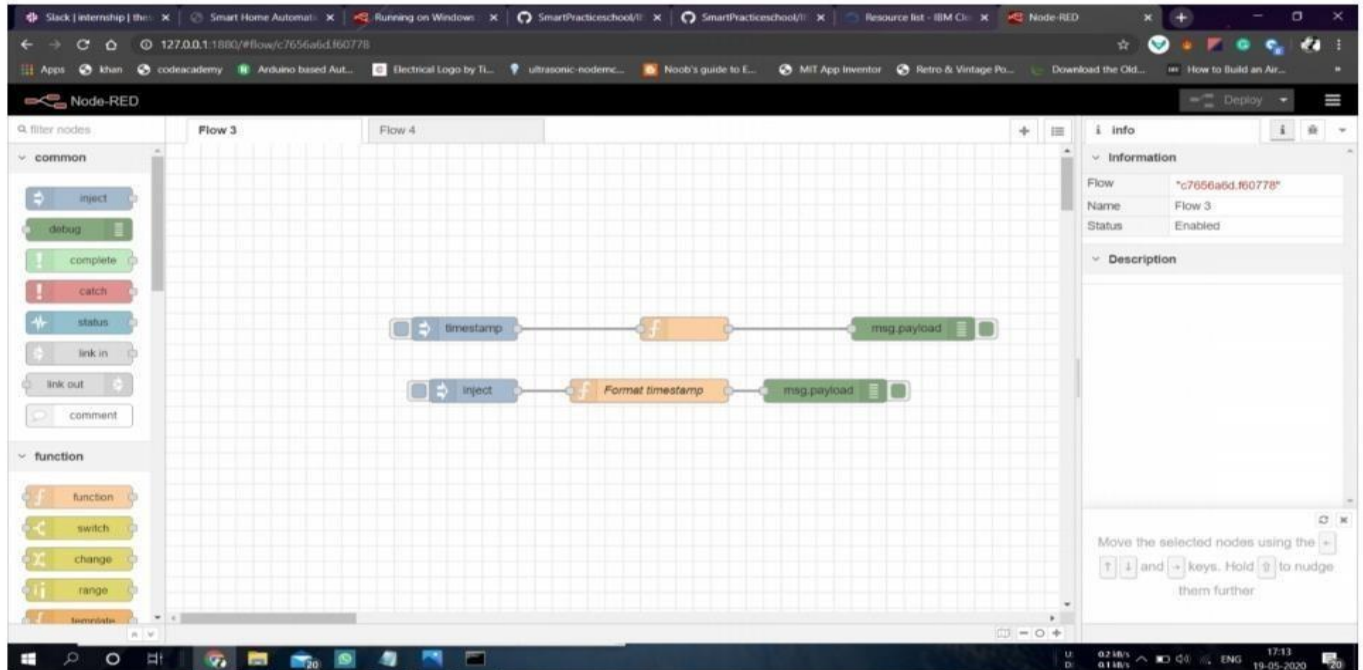


## Required Software Installation

### 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 :

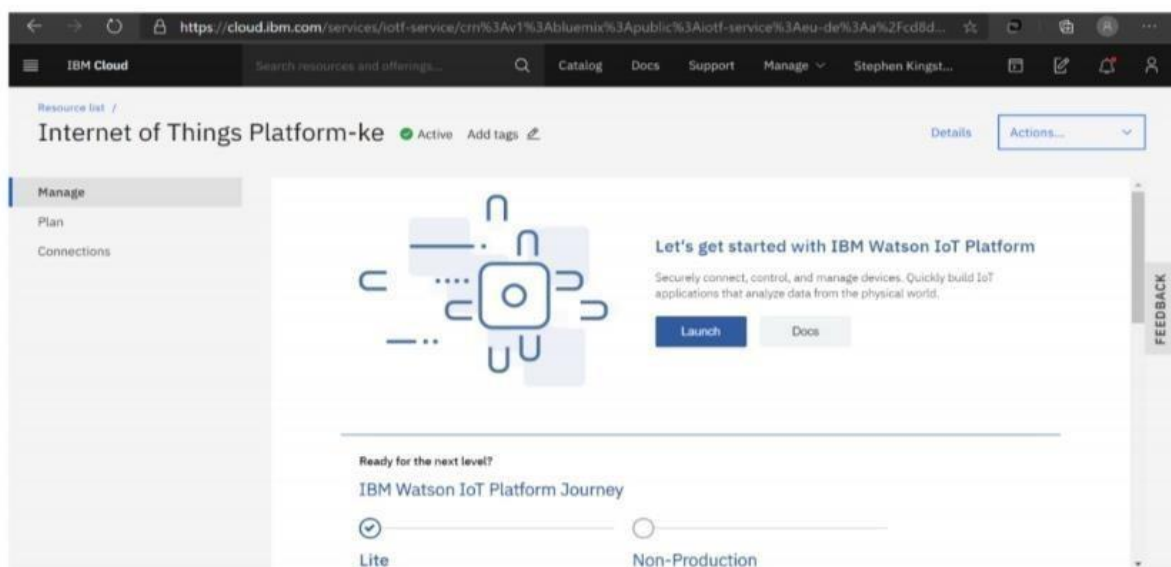
- Open cmd prompt
- Type=>node-red
- Then open <http://localhost:1880/> in browser

### 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

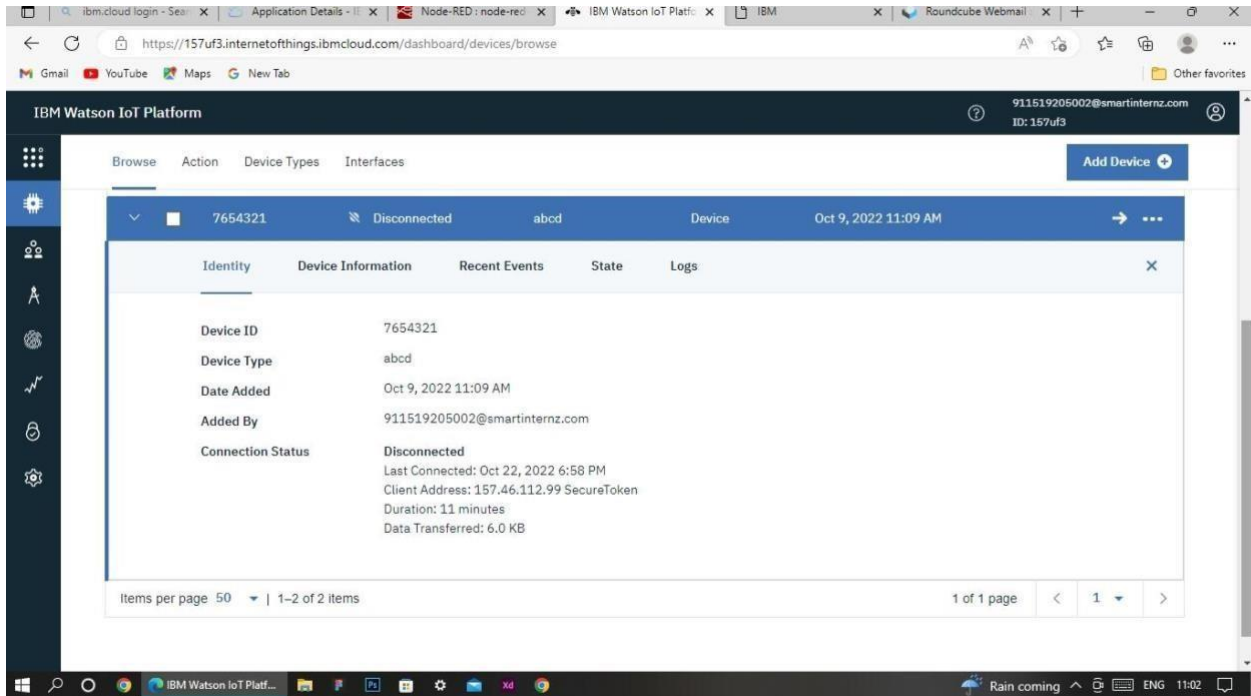
## IBM Watson IoT Platform

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 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.



## 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 Credentials

organization = "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 =%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 IoT")    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 connected
SoftwareSerial mySerial(7,8);//serial port of gsm  const

```

```

int sensor_pin = A1; // Soil moisture sensor O/P pin  int
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 OUTPUT
  pinMode(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 OFF
    delay(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;//moisture
  int 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
  {

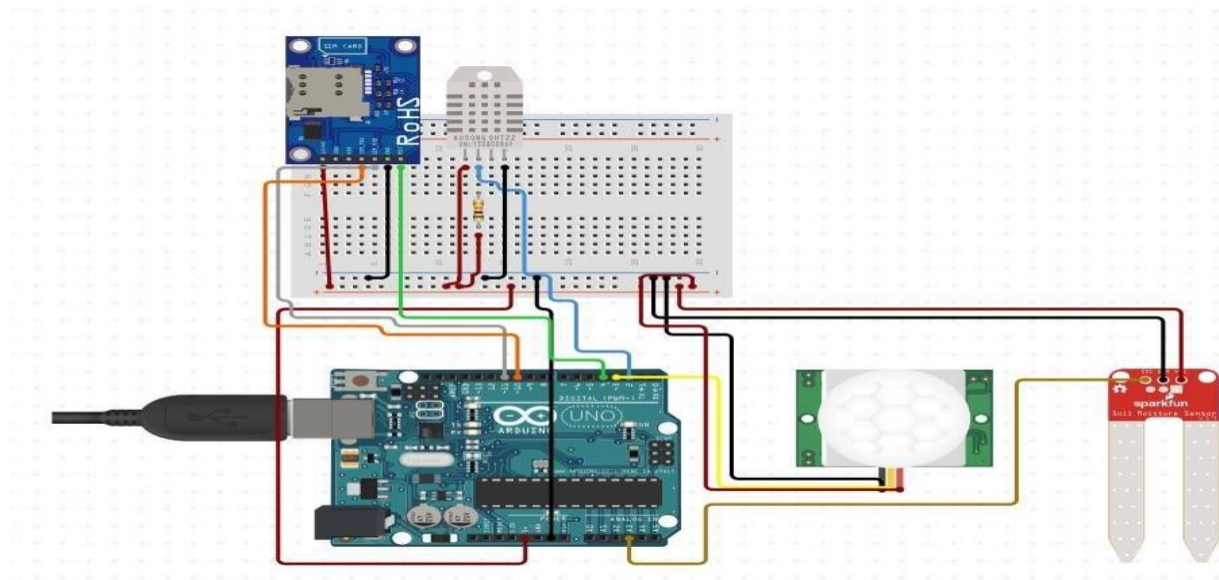
```

```

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 mobil
e 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 Mode
  delay(1000);
  mySerial.println("AT+CMGS=\"+XXXXXXXXXX\"\\r"); //replace X with 10 digit
mobile 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++;  
  
}  
  
}
```



## 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.

## 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}`