SPRINT 1-REPORT

SMART FARMER – IOT ENABLED SMART FARMING APPLICATION

Date	19 November 2022
Team ID	PNT2022TMID04701
Project Name	Smart Farmer-IOT Enabled smart farming application

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 control the equipment like water motor and other devices remotely via internet without their actual presence in the field.

Problem statement:

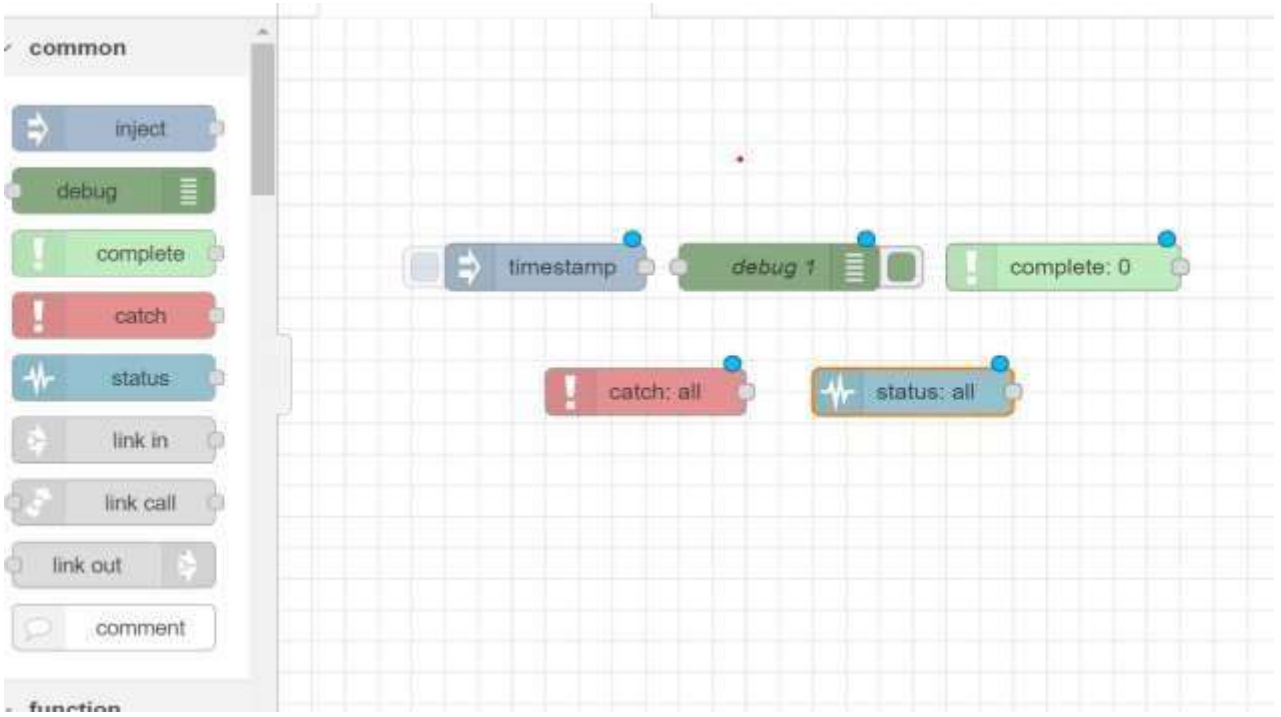
Farmers need to be present at farm for its maintenance irrespective of the weather conditions. They have to ensure that the crops are well watered and the farm status is monitored by them physically. Farmer have to stay most of the time in field in order to get a good yield. In difficult times like in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

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.

Prerequisite:

1.Node-red:



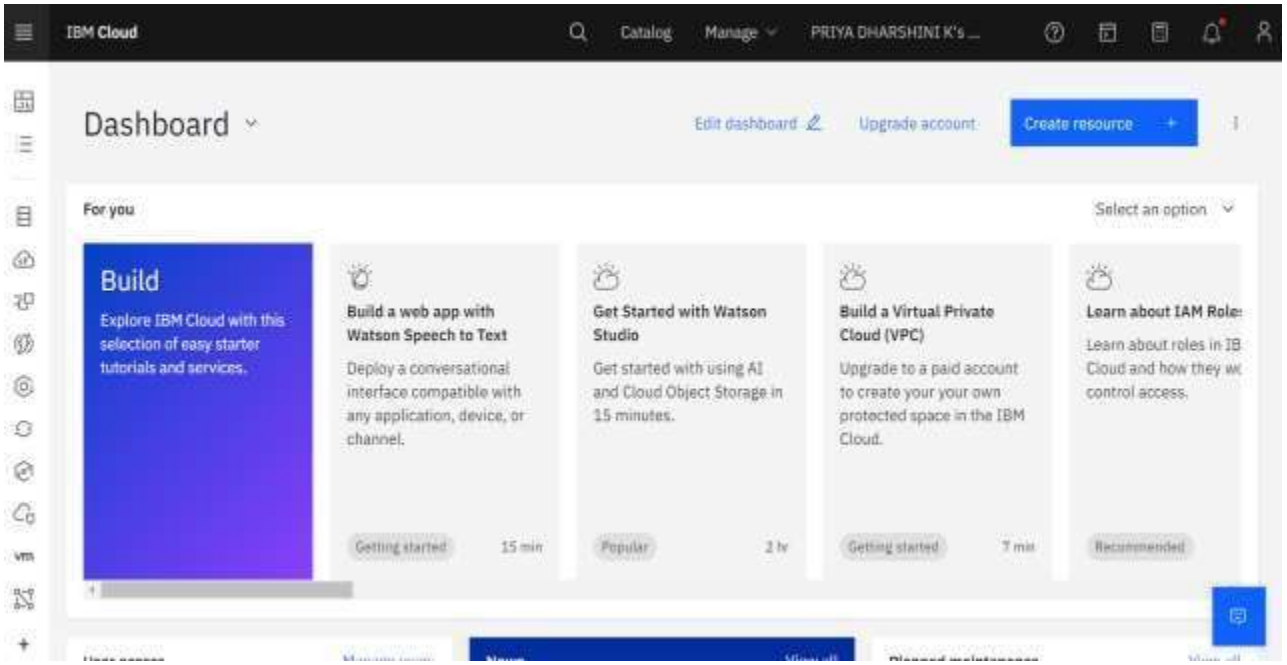
Steps to install:

1. Download Node.js
2. Do the installation process
3. Open command prompt and run the command “node –version && npm --version”
4. Install Node-red by running the command “npm install --g –unsafe-perm node-red”
5. Run Node red by simply typing “node-red” in command prompt
6. In any web browser can access node-red by http://localhost:1880

3.IBM cloud services:

Steps:

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



PYTHON CODE:

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

#Provide your IBM Watson Device Credentials
Organization = "639sac"
deviceType = "Nodemcu"
deviceId = "12345"
authMethod = "token"
authToken="1234567890"

# 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
deviceCli.connect()

while True:
        temp=random.randint(0,100) # Temperature value
        Humid=random.randint(0,100) # Humidity value
        moisture = random.randint(0,100) # Soil moisture value

        data = { 'temp' : temp, 'Humid': Humid, 'Moisture' : moisture }
#print data
        def myOnPublishCallback():
                print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %
        Humid,    "Soil Moisture = %s %%" % moisture, "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()
```

ARDUINO CODE:

```
#include "Arduino.h"
#include "dht.h"
#include "SoilMoisture.h"
#include "Pump.h"

#define  DHT_PIN 2
#define SOILMOISTURE_PIN A3
#define WATERPUMP_PIN 5
dht DHT;
int c=0;

void setup()
{
      Serial.begin(9600);
      pinMode(5, OUTPUT);// Output for Pump

      delay(1000);
}


void loop()

{
   DHT.read11(DHT_PIN);
   float h=DHT.humidity;
   float t=DHT.temperature;
   delay(1000);


   float moisture_percent;
   int moisture_analog;
   moisture_analog = analogRead(SOILMOISTURE_PIN);

   moisture_percent = ( 100 - ( (moisture_analog/1023.00) *100 ) );
```

```
  float moist= moisture_percent;
   delay(1000);

  if(moist<40)// Pump functions
 {
    while(moist<40)
   {
            digitalWrite(5 ,HIGH); // Pump ON
            moisture_analog = analogRead(SOILMOISTURE_PIN);
            moisture_percent = ( 100 - ( (moisture_analog/1023.00) *100 ) );
            moist=moisture_percent;
            delay(1000);
    }
     digitalWrite(5 ,LOW);       // Pump OFF
 }
 if(c>=0)
 {
      Serial.print("\r");
      delay(1000);
      Serial.print((String)"update>"+(String)"Temprature="+t+(String)"Humidity
      ="+h+(String)"Moisture="+moist);
      delay(1000);
      c++;
 }
}
```

CIRCUIT DIAGRAM: