

IOT ENABLED SMART FARMING APPLICATION.

Sprint Delivery – 1

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.

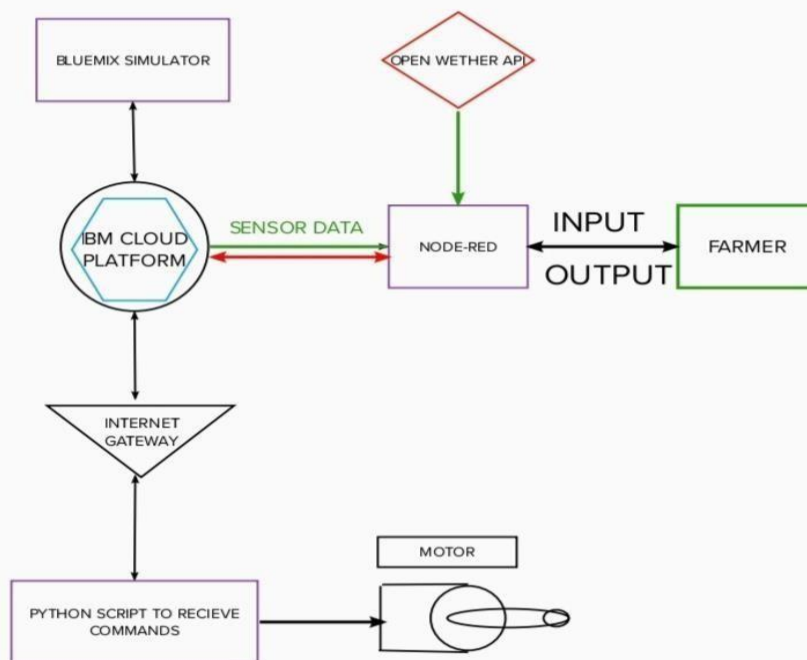
3. Proposed Solution

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

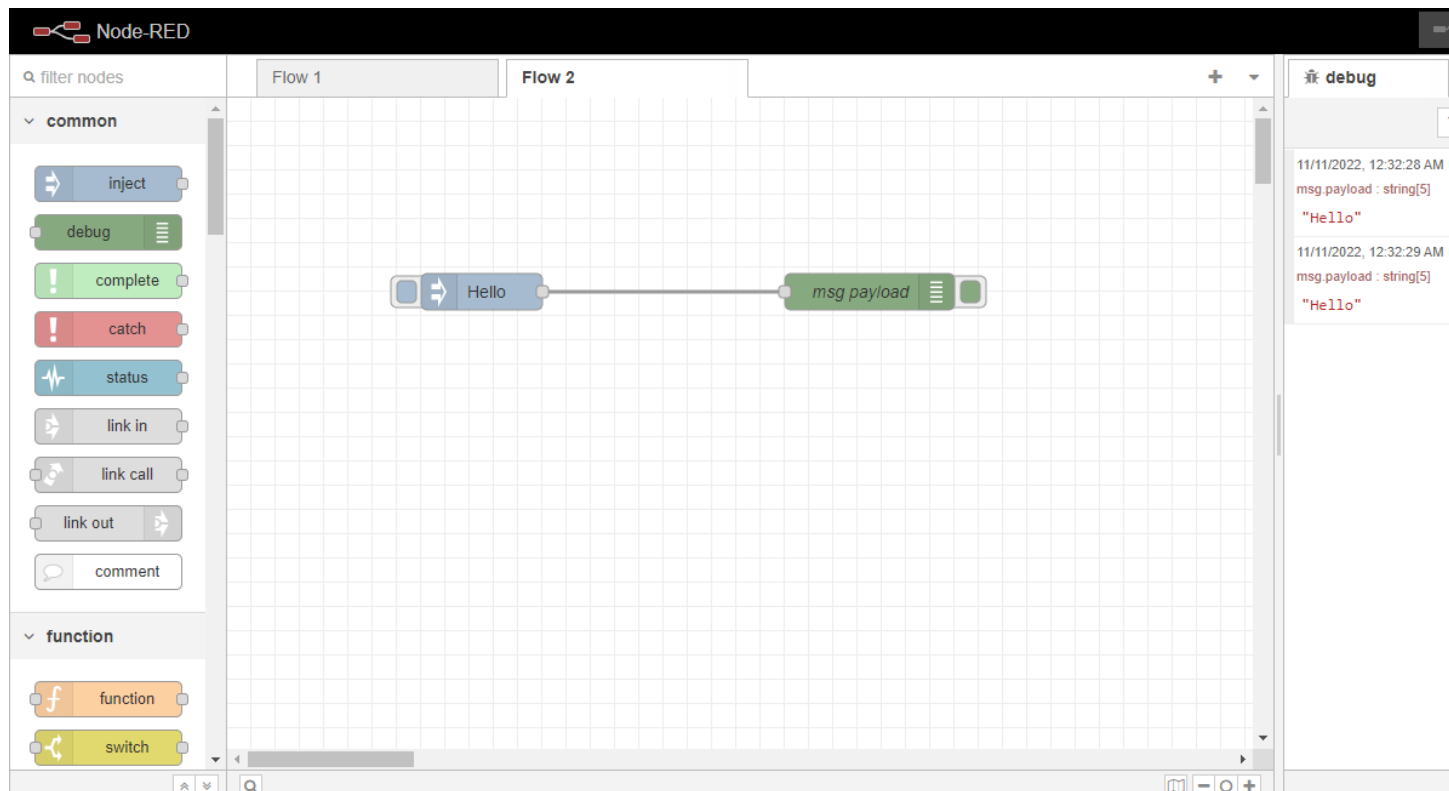
In order to implement the solution , the following approach as shown in the block diagram is used



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 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.

IBM Cloud Search resources and products... Catalog Manage Aiswarya Sree's Account

Resource list / Internet of Things Platform-q4 Active Add tags

Manage Plan Connections

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.

Launch Docs

Ready for the next level?

IBM Watson IoT Platform Journey

☒ Lite ☐ Non-Production ☐ Production

The Lite service plan provides a lightweight development environment for experimentation.

The Non-Production service plan is a full-featured, fully integrated offering that enables development and testing of IoT applications.

The Production service is a fully managed SaaS offering that enables monitoring, management, and control of IoT applications.

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

Browse Action Device Types Interfaces

Search by Device ID

Device ID	Status	Device Type	Class ID	Date Added
PNTID34083	Disconnected	PNT2022TMID34083	Device	30 Oct 2022 08:24

Identity Device Information Recent Events State Logs

Device ID PNTID34083

Device Type PNT2022TMID34083

Date Added 30 Oct 2022 08:24

Added By sreeaishu56@gmail.com

Connection Status **Disconnected**
 Last Connected: 16 Nov 2022 21:59
 Client Address: 145.40.93.209 Insecure
 Duration: 2 minutes
 Data Transferred: 8.1 KB

Items per page 50 | 1-1 of 1 item

1 Simulation running

4.2.C Python IDE

Install Python3 compiler

python1.py - C:\Users\ADHARSH\AppData\Local\Programs\Python\Python37-32\python1.py (3.7.0)

File Edit Format Run Options Window Help

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

#Provide your IBM Watson Device Credentials
organization = "b4hkg6"
deviceType = "12345"
deviceId = "54321"
authMethod = "token"
authToken = "cJG?h2d?IkxL&ZO*b"

# Initialize GPIO

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motor on":
        print ("motor is on")
    elif status=="motor off":
        print ("motor is off")
    else :
        print("please send proper comand")

    #print(cmd)

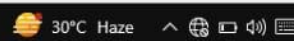
try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....

except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()
```

Connect and send a device's "Hello" with value "Hello" into the cloud as an event of type "Event1" 10 times



Type here to search



python1.py - C:\Users\ADHARSH\AppData\Local\Programs\Python\Python37-32\python1.py (3.7.0)

File Edit Format Run Options Window Help

```
print("hello to cli")
else:
    print("please send proper comand")

#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:
    #Get Sensor Data from DHT11

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

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

Type here to search

30°C Haze

```
*Python 3.7.0 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:/Users/ADHARSH/AppData/Local/Programs/Python/Python37-32/pythn1.py
2022-11-18 15:04:50,897 ibmiotf.device.Client INFO Connected successfully: d:b4hkg6:12345:54321
Published Temperature = 79 C Humidity = 72 % to IBM Watson
Published Temperature = 48 C Humidity = 43 % to IBM Watson
Published Temperature = 19 C Humidity = 57 % to IBM Watson
Published Temperature = 21 C Humidity = 70 % to IBM Watson
Published Temperature = 79 C Humidity = 33 % to IBM Watson
Published Temperature = 17 C Humidity = 49 % to IBM Watson
Published Temperature = 98 C Humidity = 60 % to IBM Watson
Published Temperature = 96 C Humidity = 16 % to IBM Watson
Published Temperature = 18 C Humidity = 36 % to IBM Watson
Published Temperature = 91 C Humidity = 93 % to IBM Watson
Published Temperature = 41 C Humidity = 89 % to IBM Watson
Published Temperature = 74 C Humidity = 71 % to IBM Watson
Published Temperature = 85 C Humidity = 4 % to IBM Watson
Published Temperature = 12 C Humidity = 40 % to IBM Watson
Published Temperature = 93 C Humidity = 90 % to IBM Watson
Published Temperature = 37 C Humidity = 62 % to IBM Watson
|
```


Code:

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

#Provide your IBM Watson Device Credentials
organization = "bxobbs"
deviceType = "b5ibm"
deviceId = "b5device"
authMethod = "token"
authToken = "b55mleibm"

# Initialize GPIO

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("motor is on")
    else :
        print ("motor is off")

    #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:
    #Get Sensor Data from DHT11

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

    data = { 'temp' : temp, 'Humid': Humid }
```

```

        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %" %
Humid, "to IBM Watson")

            success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
            if not success:
                print("Not connected to IoT")
                time.sleep(1)

            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);
    }

```

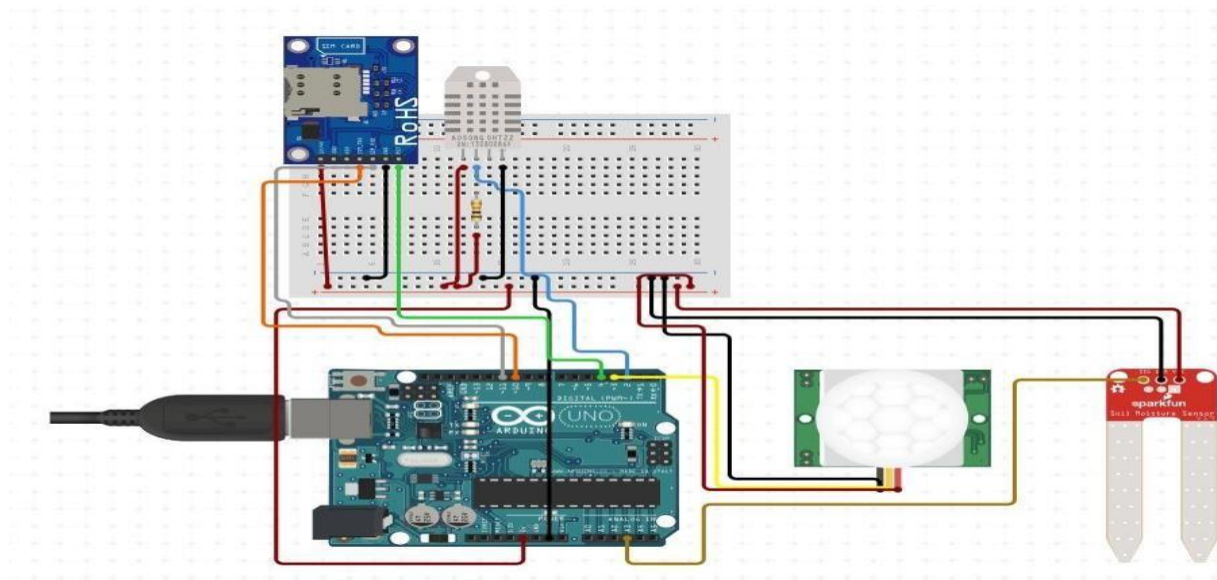
```

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++;

}

}

```



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}