Sprint Delivery – 1

Project	IoT Enabled Smart Farming Application
Team ID	PNT2022TMID11481
Date	12 November 2022

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 presencein 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.

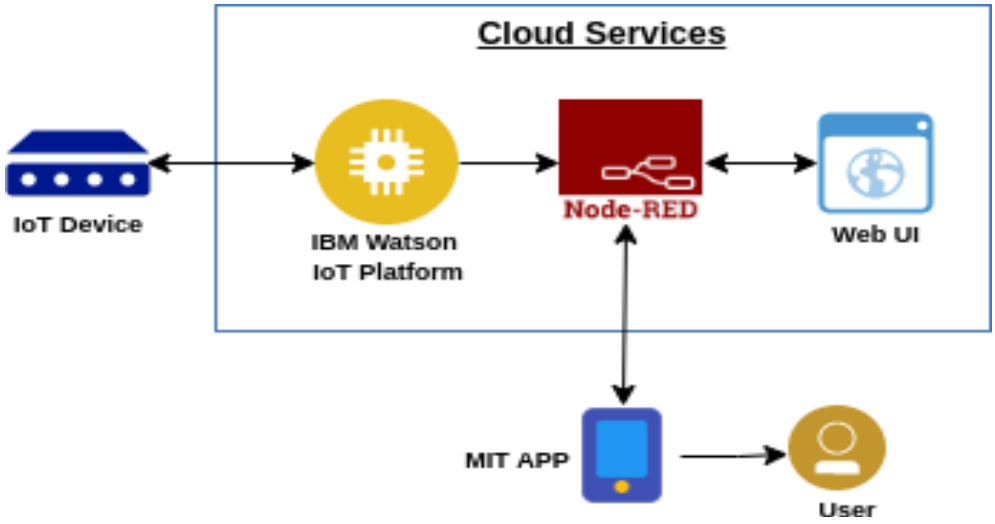
3. Proposed Solution

   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

1. Theoretical Analysis

   Block Diagram

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

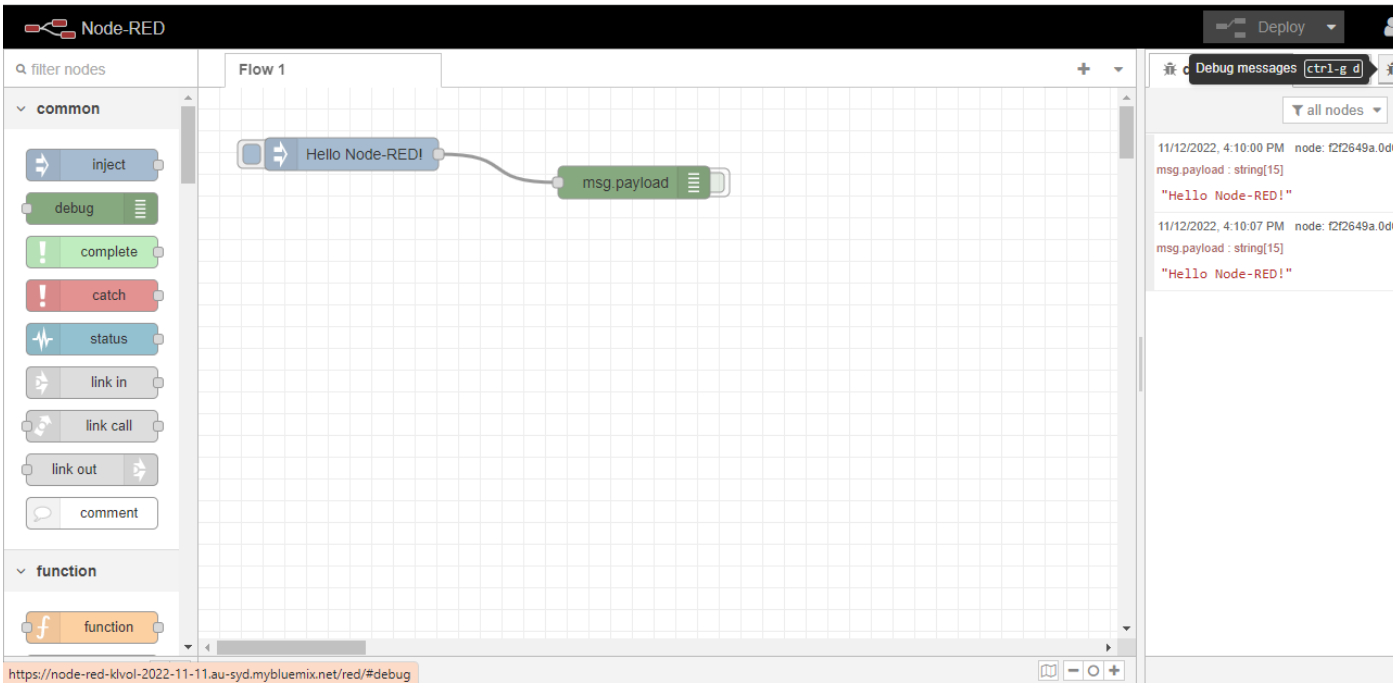


Required Software Installation

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

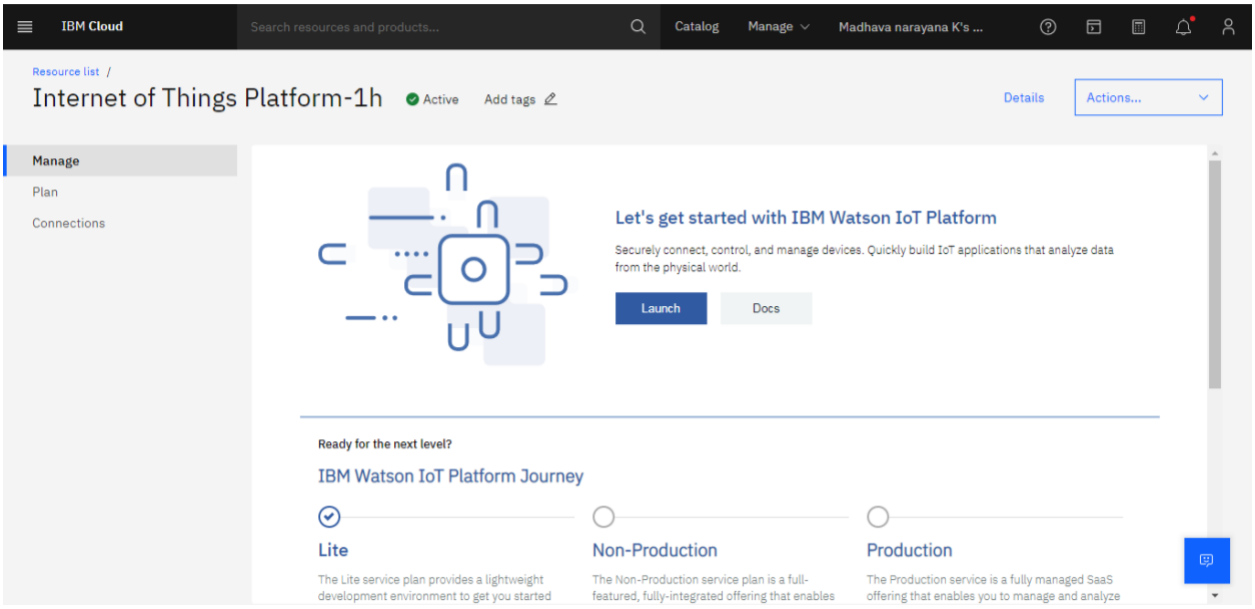
2. Dashboard node

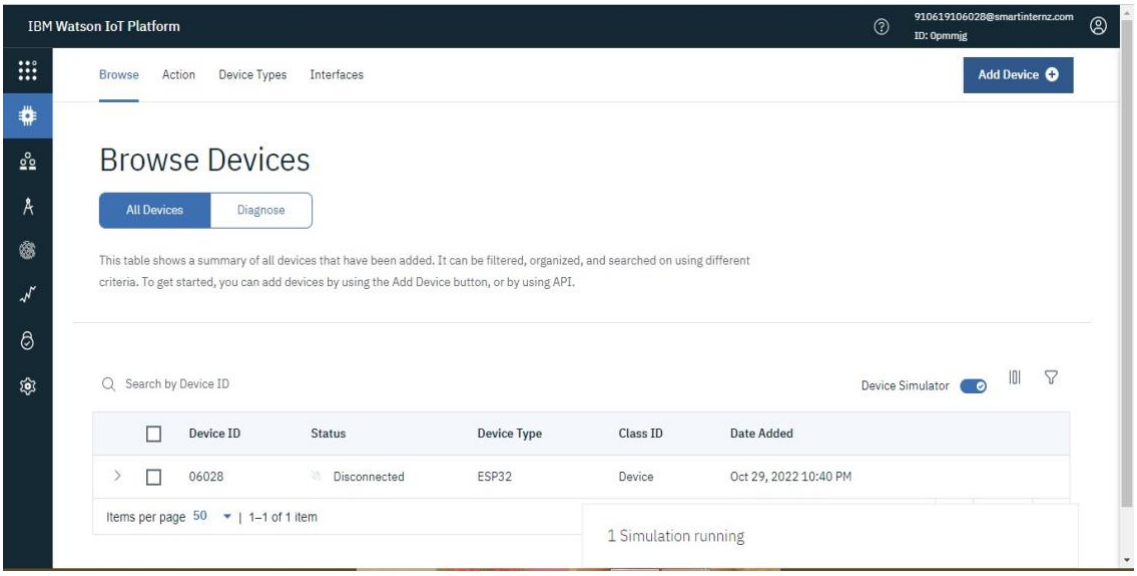
IBM Watson IoT Platform

A fully managed, cloud-hosted service with capabilities for device registration, connectivity, control, rapid visualization and data storage. IBM Watson IoT Platformis 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 Python3.8compiler

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

Code:

```
import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig = {
"identity": {
"orgId" :"0pmmjg",
"typeId":"ESP32",
"deviceId":"1234"
},
"auth": {
"token":"12345678"
}
}
client=wiotp.sdk.device.DeviceClient(config=myConfig,logHandlers=None)
client.connect()

def myCommandCallback(cmd):
print("Message received from IBM IoT platform: %s" % cmd.data ['command'])
```

```
m=cmd.data['command']
if(m=="motoron"):
print("motor is switched on")
elif(m=="motoroff"):
print("motor is switched off")
print(" ")
while True:
soil=random.randint(0,100)
temp=random.randint(-20,125)
hum=random.randint(0,100)
myData={'soil_moisture':soil,'temperature':temp,'humidity':hum}
client.publishEvent(eventId="status",msgFormat="json",data=myData
,qos=0,onPublish=None)
print("published data successfully: %s",myData)
time.sleep(2)
client.commandCallback=myCommandCallback
client.disconnect()
```

Arduino code for C :

```
String ssid     = "Simulator Wifi";  // SSID to connect to
String password = "";
String host     = "api.thingspeak.com";
const int httpPort   = 80;
String url     = "/update?api_key=6YDIQZLVKXPQN7GL&field1=";

int setupESP8266(void) {
  // Start our ESP8266 Serial Communication
  Serial.begin(115200);   // Serial connection over USB to computer
  Serial.println("AT");   // Serial connection on Tx / Rx port to ESP8266
  delay(10);        // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 1;

  // Connect to 123D Circuits Simulator Wifi
  Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");
  delay(10);        // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 2;

  // Open TCP connection to the host:
  Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
  delay(50);        // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 3;
```

```
  return 0;
}

#include<LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);

#include<Servo.h>;
Servo servo;

int air;
int motor=7;
int buzz=6;
int sprinkler=10;
int led=8;
int sensor=9;
int temp;
int pir;
float mois;
byte degree[8]={
  B00110,
  B01001,
  B01001,
  B00110,
  B00000,
  B00000
};

void setup(){
  lcd.begin(16,2);
  setupESP8266();
  Serial.begin(9600);
  pinMode(sensor,INPUT);
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  pinMode(A2,INPUT);
  pinMode(buzz,OUTPUT);
  pinMode(sprinkler,OUTPUT);
  pinMode(motor,OUTPUT);
  pinMode(led,OUTPUT);

}

void senddata(void) {
  int temp = map(analogRead(A0),20,358,-40,125);
  // Construct our HTTP call
  String httpPacket = "GET " + url + String(temp) + " HTTP/1.1\r\nHost: " + host + "\r\n\r\n";
  int length = httpPacket.length();

  // Send our message length
  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(10); // Wait a little for the ESP to respond if (!Serial.find(">")) return -1;

  // Send our http request
```

```
  Serial.print(httpPacket);
  delay(10); // Wait a little for the ESP to respond
  if (!Serial.find("SEND OK\r\n")) return;
}


void loop() {
  senddata();
  delay(20);
  air=map(analogRead(A1),0,358,0,125);
  temp=map(analogRead(A0),20,358,-40,125);
  mois=map(analogRead(A2),0,5,0,1);

  if(mois<0.5)
  {
    digitalWrite(motor,HIGH);
    lcd.setCursor(0,1);
    lcd.print("low moisture, Motor on");
    delay(10);
  }
  else if(temp>=75)
  {
    digitalWrite(sprinkler,HIGH);
    digitalWrite(led,HIGH);
    delay(10);
  }
  else
  {
    digitalWrite(sprinkler,LOW);
    digitalWrite(led,LOW);
    digitalWrite(motor,LOW);
  }
  pir=digitalRead(sensor);
  if(pir==1)
  {
    digitalWrite(buzz,HIGH);
  }
  else if(pir==0)
  {
    digitalWrite(buzz,LOW);
  }
  //Temperature:
  lcd.createChar(0,degree);
  lcd.clear();
  lcd.print("Temp:");
  lcd.print(temp);
  lcd.write(byte(0));
  lcd.print("C");
  if(mois<0.5)
  {
    lcd.setCursor(0,1);
    lcd.print("low moisture, Motor on");
  }
  else if(temp>=75)
```

```
  {
    lcd.setCursor(0,1);
    lcd.print("FIRE! EVACUATE!!");
  }
  delay(1000);
  //Air Quality:
  lcd.clear();
  lcd.print("AirQ:");
  lcd.print(air);
  lcd.print("ppm");
  lcd.setCursor(0,1);
  //Door
  if(pir==1)
  {
  lcd.print("Intruder");
  }
}
```

https://www.tinkercad.com/things/9YLynkNdia5