FINAL DELIVERABLES

FINAL CODE

Team ID	PNT2022TMID52856
Project Name	SmartFarmer - IoT Enabled Smart Farming Application

ESP-32 PROGRAMMING

```
#include <string.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <ArduinoJson.h>
#include <PubSubClient.h>
#include "DHT.h"

float distance=44;
#define sound_speed 0.034
int trigpin=18;
int echopin=19;
int led=5;
int LED=9;
long duration;
String message;
int ph;
int temp;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//***IBM Account**

#define ORG "94ab7c"//IBM ORGANITION ID
#define DEVICE_TYPE "Node"//Device type in ibm watson IOT Platform
#define DEVICE_ID "esp2"//Device ID in ibm watson IOT Platform
#define TOKEN "ChVhYc0Dz(AD*rSw9A"
```

```
String data3;
float h, t;

//** Formatting the values**

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/Motor/fmt/json";// cmd  REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id

WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing parameter like server id,portand wificredential

const int BUTTON_PIN = 19; // Arduino pin connected to button's pin
const int RELAY_PIN  = 27; // Arduino pin connected to relay's pin
#define DHTpin 4    //D15 of ESP32 DevKit
#define DHTTYPE DHT11
DHT dht(DHTpin,DHTTYPE);

#define LIGHT_SENSOR_PIN 33 // ESP32 pin GIOP36 (ADC0)

#define rainAnalog 35
#define rainDigital 34

#define AOUT_PIN 39 // ESP32 pin GIOP36 (ADC0) that connects to AOUT pin of moisture sensor
void setup()
```

```
{
  Serial.begin(115200);
  wificonnect();
  mqttconnect();
  client.subscribe(subscribetopic);
  client.setCallback(callback);

  randomSeed(42);
  pinMode(BUTTON_PIN, INPUT_PULLUP); // set arduino pin to input pull-up mode
  pinMode(RELAY_PIN, OUTPUT);        // set arduino pin to output mode

  Serial.println("Status\tHumidity (%)\tTemperature (C)\t(F)\tHeatIndex (C)\t(F)");
  dht.begin();
  //dht.setup(DHTpin, DHTesp::DHT11); //for DHT11 Connect DHT sensor to GPIO 17
  //dht.setup(DHTpin, DHTesp::DHT22); //for DHT22 Connect DHT sensor to GPIO 17

  pinMode(rainDigital,INPUT);

  delay(10);
  Serial.println();
}

void loop()
{

  client.loop();
  Serial.println("\n\n");
  float temp = dht.readTemperature();
  Serial.print("Temperature: ");
  Serial.println(temp);
  ph = random(5,9);//random value as sensor not available
  Serial.print("pH level: ");
```

```
 Serial.print(ph);

// reads the input on analog pin (value between 0 and 4095)
 int analogValue = analogRead(LIGHT_SENSOR_PIN);
 Serial.print("\nLDR reading = ");
 Serial.print(analogValue);   // the raw analog reading

 // We'll have a few threshholds, qualitatively determined
 if (analogValue < 400)
 {
   Serial.print(" => Dark condition");
 } else if (analogValue < 800)
   {
     Serial.print(" => Dim light");
   } else if (analogValue < 2000)
   {
     Serial.print(" => normal light");
   } else if (analogValue < 3200)
     {
       Serial.print(" => Bright");
     } else
     {
       Serial.print(" => Very bright");
     }

 int rainAnalogVal = analogRead(rainAnalog);
 int rainDigitalVal = digitalRead(rainDigital);
 Serial.print("\nRain gauge: ");
 Serial.print(rainAnalogVal);
 Serial.print("\t Raining?: ");
 if(rainDigitalVal==1)
 {
```

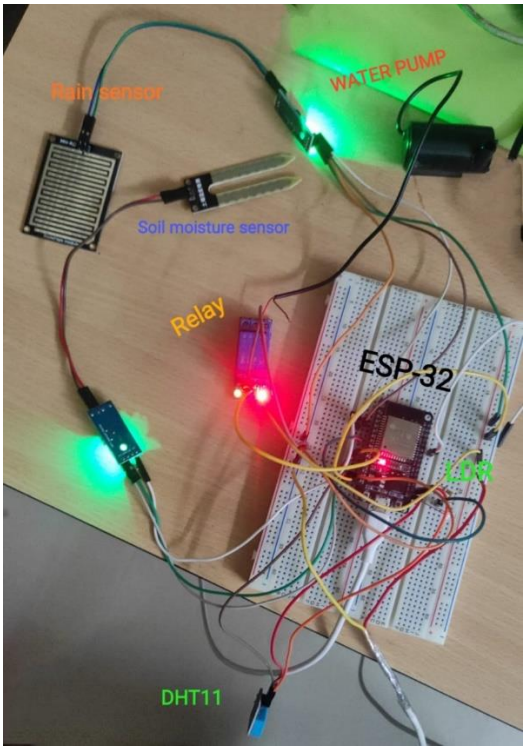
```
  Serial.print("No");
 }
 else
 {
  Serial.print("Yes");
 }

 int value = analogRead(AOUT_PIN); // read the analog value from sensor
 Serial.print("\nMoisture value: ");
 Serial.println(value);

 message="Normal";

 PublishData(temp,value,analogValue,rainAnalogVal,ph,message);
 callback;
 delay(1000);
}
//**Publish***

void PublishData(int temp,int sm,int ldr,int rain,int ph, String a)
{
 mqttconnect();

 //creating the String in in form JSon to update the data to ibm cloud

 DynamicJsonDocument doc(1024);
 String payload;
 doc["Temperature"]=temp;
 doc["Soil_moisture"]=sm;
 doc["Ambient_Light_LDR"]=ldr;
 doc["pH_sensor"]=ph;
 doc["Rain_sensor"]=rain;
```

```
  doc["message"]=a;
  serializeJson(doc, payload);
  Serial.print("Sending payload: ");
  Serial.println(payload);

  if (client.publish(publishTopic,(char*)  payload.c_str()))
  {
    Serial.println("Publish ok");// if upload sucessful
    client.subscribe(subscribetopic);
  }
  else
  {
    Serial.println("Publish failed");
  }
}

void mqttconnect()
{
  if (!client.connected())
  {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token))
    {
      Serial.print(".");
      delay(500);
    }

    initManagedDevice();
    Serial.println();
  }
}
```

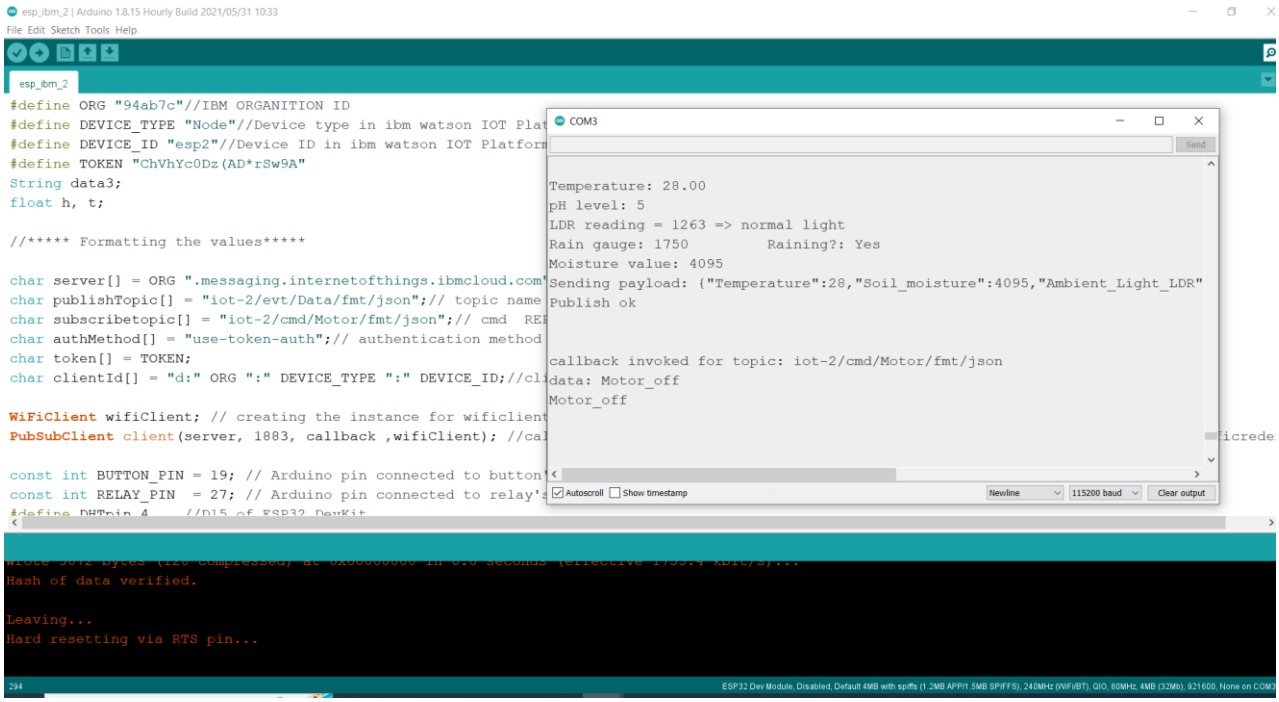
```
const char* ssid     = "POCO";
const char* password = "hpranga44";
const char* host = "192.168.6.129";
void wificonnect() //function defination for wificonnect
{
  Serial.println();
  Serial.print("Connecting to ");

  WiFi.begin(ssid, password);//passing the wifi credentials to establish the connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void initManagedDevice()
{
  if (client.subscribe(subscribetopic))
  {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  }
  else
  {
    Serial.println("subscribe to cmd FAILED");
  }

}
```

```
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{
  String comdata="";
  Serial.println("\n");
  Serial.print("callback invoked for topic: ");
  Serial.println(subscribetopic);
  for (int i = 12; i < payloadLength-2; i++)
  {
    //Serial.print((char)payload[i]);
    comdata += (char)payload[i];
  }
  Serial.println("data: "+ comdata);
  if(comdata=="Motor_on")
  {
     Serial.println(comdata);
    digitalWrite(RELAY_PIN,HIGH);
  }
  else
  {
    Serial.println(comdata);
    digitalWrite(RELAY_PIN,LOW);
  }
comdata="";
}
```

OUTPUT:



Sensors, relay and water pump integrated with ESP-32.



Serial monitor of Arduino IDE

PYTHON 3.7 - CODE:

Python script to publish and subscribe to IBM IoT platform

```
import wiotp.sdk.device
import time
import os
import datetime
import random

#IBM CREDENTIALS
myConfig = {
"identity": {
"orgId":"94ab7c",
"typeId":"Node",
"deviceId": "esp2"
},
"auth": {
"token": "ChVhYc0Dz(AD*rSw9A"
} }


client = wiotp.sdk.device.DeviceClient (config=myConfig,logHandlers=None)
client.connect ()


#Commands received through App/node red
def myCommandCallback (cmd) :
    print ("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']
    if (m=="Motor_on"):
        print ("Motor is switched on")
    elif (m=="Motor_off"):
        print ("Motor is switched OFF")
    print (" ")

while True:
    #Generate random sensor values
    soil=random.randint (1,100)
    temp=random.randint (-10,60)
    ldr=random.randint (0, 1023)
    rain=random.randint (0, 1023)
    ph=random.randint (5, 9)
```

```
#Publish and subscribe to IBM IoT platform
myData={'Temperature':temp,'Soil_moisture': soil ,'Ambient_Light_LDR' :ldr,'
      Rain_sensor':rain,'pH_sensor':ph}
client.publishEvent (eventId="status", msgFormat="json", data=myData, qos=0 ,
onPublish=None)
print ("Published data Successfully: ", myData)
time.sleep (2)
client.commandCallback = myCommandCallback
client.disconnect ()
```

PYTHON CODE EXECUTION

```
*Python 3.7.0 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/DELL/AppData/Local/Programs/Python/Python37/python_watson_publish.py
2022-11-18 00:30:29,592   wiotp.sdk.device.client.DeviceClient  INFO    Connected successfully: d:94ab7c:Node:esp2Published data Successfully
 {'Temperature': 13, 'Soil_moisture': 31, 'Ambient_Light_LDR': 162, 'Rain_sensor': 412, 'pH_sensor': 7}
Published data Successfully:  {'Temperature': -7, 'Soil_moisture': 82, 'Ambient_Light_LDR': 775, 'Rain_sensor': 649, 'pH_sensor': 9}
Published data Successfully:  {'Temperature': 32, 'Soil_moisture': 7, 'Ambient_Light_LDR': 383, 'Rain_sensor': 264, 'pH_sensor': 6}
Published data Successfully:  {'Temperature': 59, 'Soil_moisture': 3, 'Ambient_Light_LDR': 336, 'Rain_sensor': 893, 'pH_sensor': 5}
Published data Successfully:  {'Temperature': 16, 'Soil_moisture': 18, 'Ambient_Light_LDR': 260, 'Rain_sensor': 955, 'pH_sensor': 6}
Published data Successfully:  {'Temperature': 2, 'Soil_moisture': 5, 'Ambient_Light_LDR': 769, 'Rain_sensor': 1019, 'pH_sensor': 7}
Published data Successfully:  {'Temperature': 38, 'Soil_moisture': 71, 'Ambient_Light_LDR': 518, 'Rain_sensor': 57, 'pH_sensor': 8}
Published data Successfully:  {'Temperature': -3, 'Soil_moisture': 100, 'Ambient_Light_LDR': 770, 'Rain_sensor': 326, 'pH_sensor': 8}
Published data Successfully:  {'Temperature': 48, 'Soil_moisture': 9, 'Ambient_Light_LDR': 304, 'Rain_sensor': 776, 'pH_sensor': 7}
Published data Successfully:  {'Temperature': 33, 'Soil_moisture': 24, 'Ambient_Light_LDR': 581, 'Rain_sensor': 158, 'pH_sensor': 6}
Published data Successfully:  {'Temperature': 53, 'Soil_moisture': 80, 'Ambient_Light_LDR': 618, 'Rain_sensor': 874, 'pH_sensor': 6}
Published data Successfully:  {'Temperature': 25, 'Soil_moisture': 54, 'Ambient_Light_LDR': 746, 'Rain_sensor': 537, 'pH_sensor': 6}
Published data Successfully:  {'Temperature': -5, 'Soil_moisture': 89, 'Ambient_Light_LDR': 482, 'Rain_sensor': 738, 'pH_sensor': 9}
Published data Successfully:  {'Temperature': 12, 'Soil_moisture': 25, 'Ambient_Light_LDR': 290, 'Rain_sensor': 953, 'pH_sensor': 9}
Published data Successfully:  {'Temperature': -9, 'Soil_moisture': 44, 'Ambient_Light_LDR': 583, 'Rain_sensor': 637, 'pH_sensor': 8}
Published data Successfully:  {'Temperature': 9, 'Soil_moisture': 64, 'Ambient_Light_LDR': 609, 'Rain_sensor': 330, 'pH_sensor': 7}
Published data Successfully:  {'Temperature': 11, 'Soil_moisture': 69, 'Ambient_Light_LDR': 261, 'Rain_sensor': 445, 'pH_sensor': 8}
Published data Successfully:  {'Temperature': 48, 'Soil_moisture': 73, 'Ambient_Light_LDR': 966, 'Rain_sensor': 135, 'pH_sensor': 5}
Message received from IBM IoT Platform: Motor_on
Motor is switched on

Published data Successfully:  {'Temperature': -10, 'Soil_moisture': 41, 'Ambient_Light_LDR': 429, 'Rain_sensor': 296, 'pH_sensor': 7}
Published data Successfully:  {'Temperature': 53, 'Soil_moisture': 77, 'Ambient_Light_LDR': 465, 'Rain_sensor': 152, 'pH_sensor': 5}
Published data Successfully:  {'Temperature': 35, 'Soil_moisture': 48, 'Ambient_Light_LDR': 446, 'Rain_sensor': 152, 'pH_sensor': 5}
Published data Successfully:  {'Temperature': 42, 'Soil_moisture': 20, 'Ambient_Light_LDR': 894, 'Rain_sensor': 547, 'pH_sensor': 5}
Published data Successfully:  {'Temperature': 26, 'Soil_moisture': 41, 'Ambient_Light_LDR': 212, 'Rain_sensor': 529, 'pH_sensor': 8}
Published data Successfully:  {'Temperature': 58, 'Soil_moisture': 43, 'Ambient_Light_LDR': 709, 'Rain_sensor': 256, 'pH_sensor': 6}
Message received from IBM IoT Platform: Motor_off
Motor is switched OFF

Published data Successfully:  {'Temperature': 7, 'Soil_moisture': 33, 'Ambient_Light_LDR': 897, 'Rain_sensor': 976, 'pH_sensor': 8}
Published data Successfully:  {'Temperature': 60, 'Soil_moisture': 11, 'Ambient_Light_LDR': 157, 'Rain_sensor': 841, 'pH_sensor': 9}
Published data Successfully:  {'Temperature': 26, 'Soil_moisture': 83, 'Ambient_Light_LDR': 1018, 'Rain_sensor': 207, 'pH_sensor': 9}
Published data Successfully:  {'Temperature': 30, 'Soil_moisture': 39, 'Ambient_Light_LDR': 836, 'Rain_sensor': 311, 'pH_sensor': 5}
```

EVENT LOGGED IN IBM IoT WATSON PLATFORM