COLLEGE CODE: 3114

COLLEGE NAME: MEENAKSHI COLLEGE OF
ENGINEERING

DEPARTMENT: INFORMATION TECHNOLOGY

STUDENT NM-ID: autit1062

ROLL NO: 311423205062

DATE: 16-05-2025

TECHNOLOGY-PROJECT NAME:

IOT-CROP AND PEST MONITORING

SUBMITTED BY:

SIVA R

SELVAKUMAR S

SATHRIYAN U

SANJAY J

PHASE 5:

PROJECT DEMONSTRATION & DOCUMENTATION

Title: Crop and Pest Monitoring using IoT

Abstract:

The Crop and Pest Monitoring project leverages IoT (Internet of Things) to provide real-time monitoring of crop health and pest activity. In its final phase, the system integrates sensor data, predictive analytics, and remote alerts to assist farmers in optimizing crop yield and managing pest infestations effectively. This document outlines the system's architecture, data acquisition process, sensor integration, and predictive modeling, with a focus on scalability and data accuracy. Visual representations, code snippets, and testing reports will be included for comprehensive understanding.

Index:

---

1. Project Demonstration

Overview:

The Crop and Pest Monitoring system will be demonstrated to stakeholders, highlighting its key features, sensor data integration, and real-time analytics capabilities. The demonstration will focus on system responsiveness, data accuracy, and pest detection efficiency.

Demonstration Details:

- System Walkthrough: A step-by-step walkthrough showcasing sensor data acquisition, data visualization, and alert generation for potential pest threats.

• Predictive Analytics: Demonstrating how the system utilizes sensor data to predict potential pest outbreaks and crop stress.
• IoT Integration: Real-time monitoring of temperature, humidity, and pest activity data collected from sensors deployed in the field.
• Performance Metrics: Evaluating system response time, data processing speed, and scalability in handling multiple data streams.
• Security & Privacy: Showcasing data encryption and secure data transmission protocols to ensure data integrity and privacy.

Outcome:
The demonstration will validate the system's efficacy in detecting pest activity and monitoring crop health, providing actionable insights to stakeholders.

---

2. Project Documentation

Overview:
The project documentation for the Crop and Pest Monitoring system includes detailed explanations of the system architecture, sensor integration, data processing modules, and usage guidelines.

Documentation Sections:
• System Architecture: Diagrams illustrating sensor nodes, data collection pipelines, and predictive analytics workflows.
• Code Documentation: Source code and explanations for all modules, including sensor data acquisition, data processing, and alert generation.
• User Guide: A manual for farmers and stakeholders on how to interpret data visualizations and alerts for timely interventions.
• Administrator Guide: Instructions for system maintenance, sensor calibration, and data integrity checks.
• Testing Reports: Performance metrics, data accuracy assessments, and pest detection efficacy reports.

Outcome:
Comprehensive documentation will be provided to support ongoing development, deployment, and maintenance of the monitoring system.

---

3. Feedback and Final Adjustments

Overview:

Feedback will be collected from instructors, farmers, and stakeholders to refine the system before deployment.

Steps:

- Feedback Collection: Observations and suggestions during the demonstration will be documented for further refinement.
- Refinement: Necessary adjustments will be made to enhance data accuracy, alert sensitivity, and user interface design.
- Final Testing: Comprehensive testing to confirm the system's reliability in varied agricultural conditions.

Outcome:

System refinements will be implemented to address identified gaps, ensuring a robust and reliable monitoring solution.

---

4. Final Project Report Submission

Overview:

The final report will summarize the project phases, key features, challenges, and achieved outcomes, with recommendations for future enhancements.

Report Sections:

- Executive Summary: An overview of the project, objectives, and key findings.
- Phase Breakdown: Detailed breakdown of each project phase, including sensor integration, data processing, and predictive modeling.
- Challenges & Solutions: Identifying key challenges like sensor calibration and data discrepancies, with proposed solutions.
- Outcomes: Summary of the system’s readiness for field deployment and pest monitoring accuracy.

Outcome:

A comprehensive report will be submitted to encapsulate the project’s progress, technical insights, and future scope.

---

5. Project Handover and Future Works

Overview:

Recommendations for future work will focus on expanding sensor networks, enhancing predictive models, and integrating weather data for better forecasting.

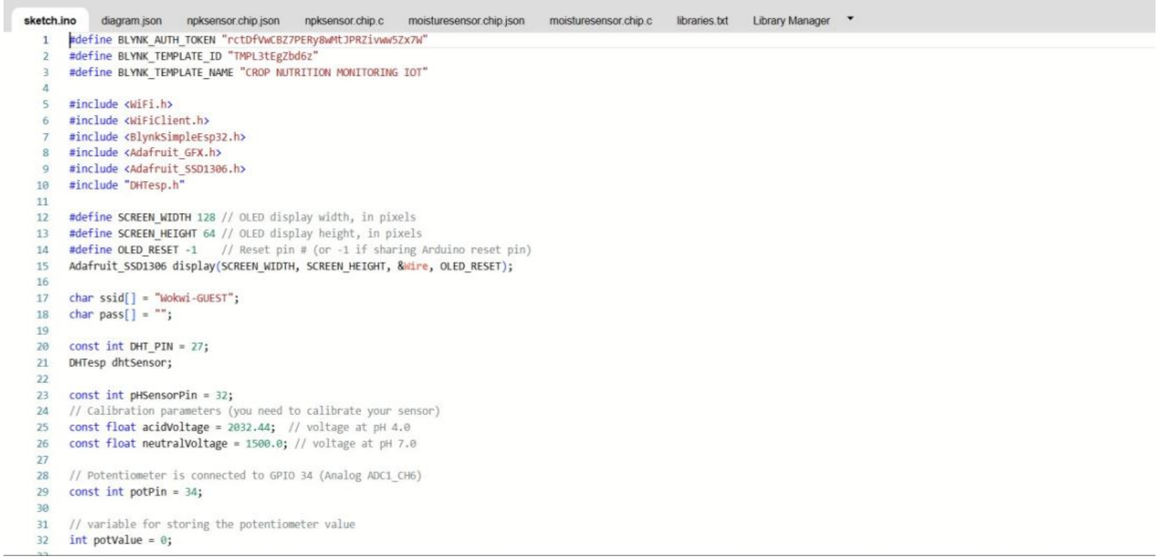
Handover Details:

- Next Steps: Scaling the system for larger farms, improving data visualization, and integrating additional pest detection sensors.

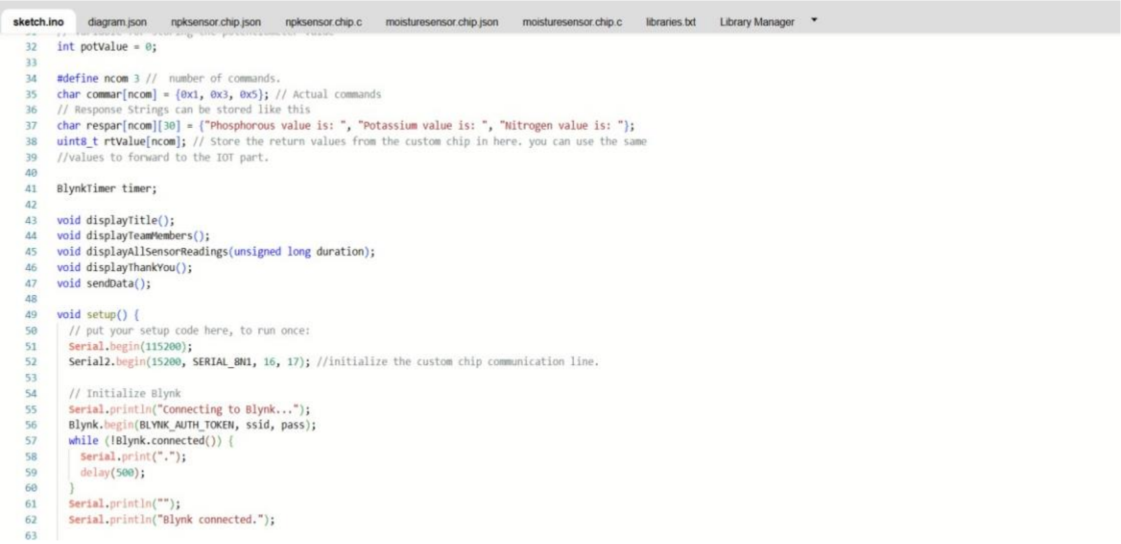
Outcome:

The project will be handed over along with guidelines for further development and field testing.

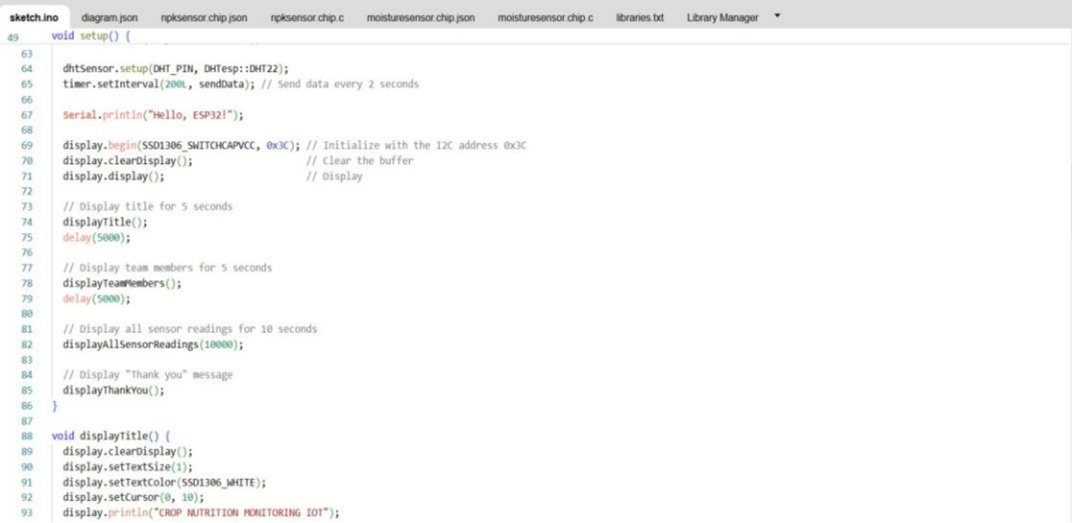
Include screenshots of source code and the working final project.



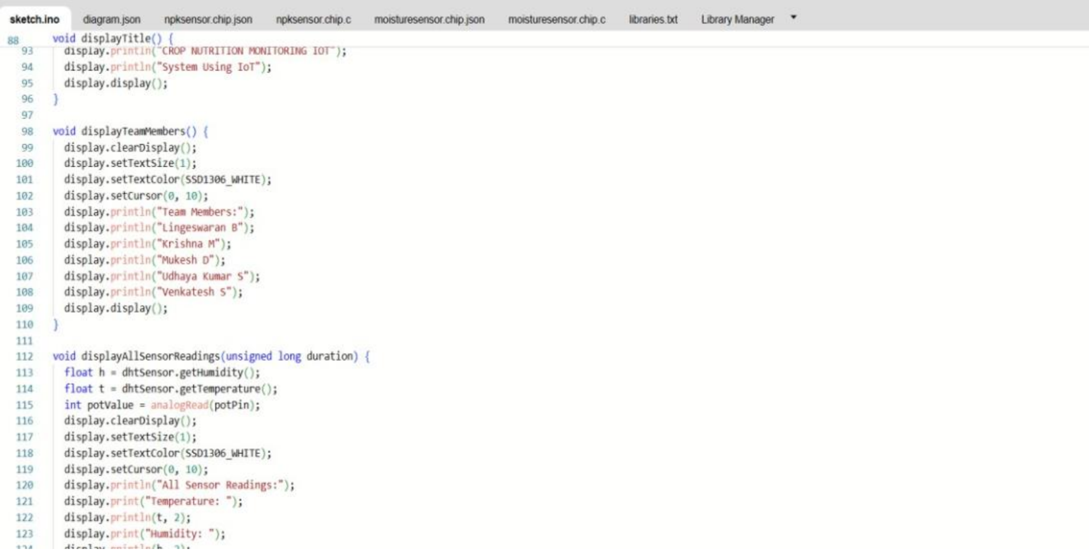
```
#define BLYNK_AUTH_TOKEN "rctDfVwCBZ7PERy8wMtJPRZivww5Zx7W"
#define BLYNK_TEMPLATE_ID "TMPL3tEgZbd6z"
#define BLYNK_TEMPLATE_NAME "CROP NUTRITION MONITORING IOT"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "DHTesp.h"

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET -1    // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

char ssid[] = "Wokwi-GUEST";
char pass[] = "";

const int DHT_PIN = 27;
DHTesp dhtSensor;

const int pHSensorPin = 32;
// Calibration parameters (you need to calibrate your sensor)
const float acidVoltage = 2032.44;  // voltage at pH 4.0
const float neutralVoltage = 1500.0; // voltage at pH 7.0

// Potentiometer is connected to GPIO 34 (Analog ADC1_CH6)
const int potPin = 34;

// variable for storing the potentiometer value
int potValue = 0;
```



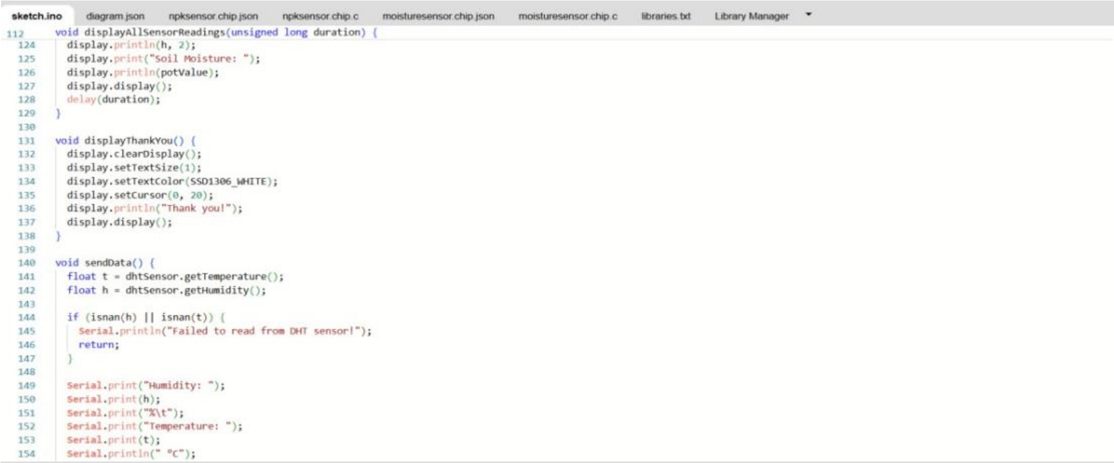
```
// Variable for storing the potentiometer value
int potValue = 0;

#define ncom 3 //  number of commands.
char commar[ncom] = {0x1, 0x3, 0x5}; // Actual commands
// Response Strings can be stored like this
char respar[ncom][30] = {"Phosphorous value is: ", "Potassium value is: ", "Nitrogen value is: "};
uint8_t rtValue[ncom]; // Store the return values from the custom chip in here. you can use the same
//values to forward to the IOT part.

BlynkTimer timer;

void displayTitle();
void displayTeamMembers();
void displayAllSensorReadings(unsigned long duration);
void displayThankYou();
void sendData();

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial2.begin(15200, SERIAL_8N1, 16, 17); //initialize the custom chip communication line.

  // Initialize Blynk
  Serial.println("Connecting to Blynk...");
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  while (!Blynk.connected()) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("Blynk connected.");

```



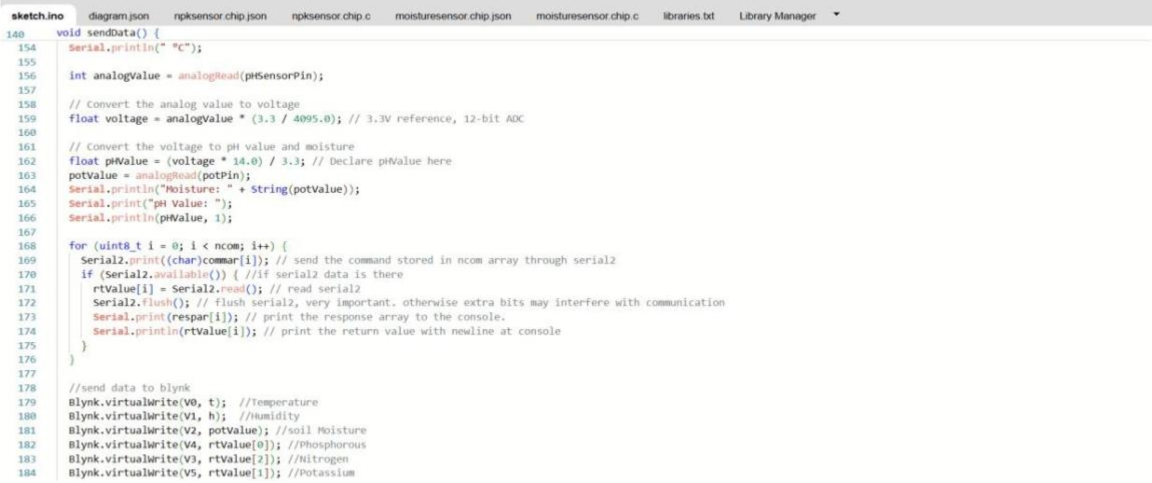
```
void setup() {

  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  timer.setInterval(200L, sendData); // Send data every 2 seconds

  Serial.println("Hello, ESP32!");

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // Initialize with the I2C address 0x3C
  display.clearDisplay();                    // Clear the buffer
  display.display();                         // Display

  // Display title for 5 seconds
  displayTitle();
  delay(5000);

  // Display team members for 5 seconds
  displayTeamMembers();
  delay(5000);

  // Display all sensor readings for 10 seconds
  displayAllSensorReadings(10000);

  // Display "Thank you" message
  displayThankYou();
}

void displayTitle() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 10);
  display.println("CROP NUTRITION MONITORING IOT");
```



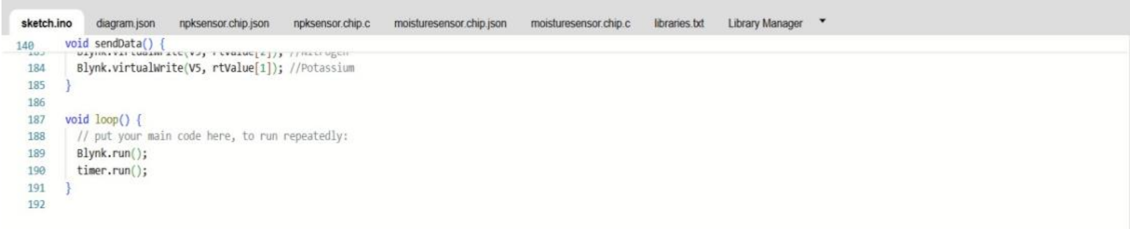
```
void displayTitle() {
  display.println("CROP NUTRITION MONITORING IOT");
  display.println("System Using IoT");
  display.display();
}

void displayTeamMembers() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 10);
  display.println("Team Members:");
  display.println("Lingeswaran B");
  display.println("Krishna M");
  display.println("Mukesh D");
  display.println("Udhaya Kumar S");
  display.println("Venkatesh S");
  display.display();
}

void displayAllSensorReadings(unsigned long duration) {
  float h = dhtSensor.getHumidity();
  float t = dhtSensor.getTemperature();
  int potValue = analogRead(potPin);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 10);
  display.println("All Sensor Readings:");
  display.print("Temperature: ");
  display.println(t, 2);
  display.print("Humidity: ");
  display.println(h, 2);
```



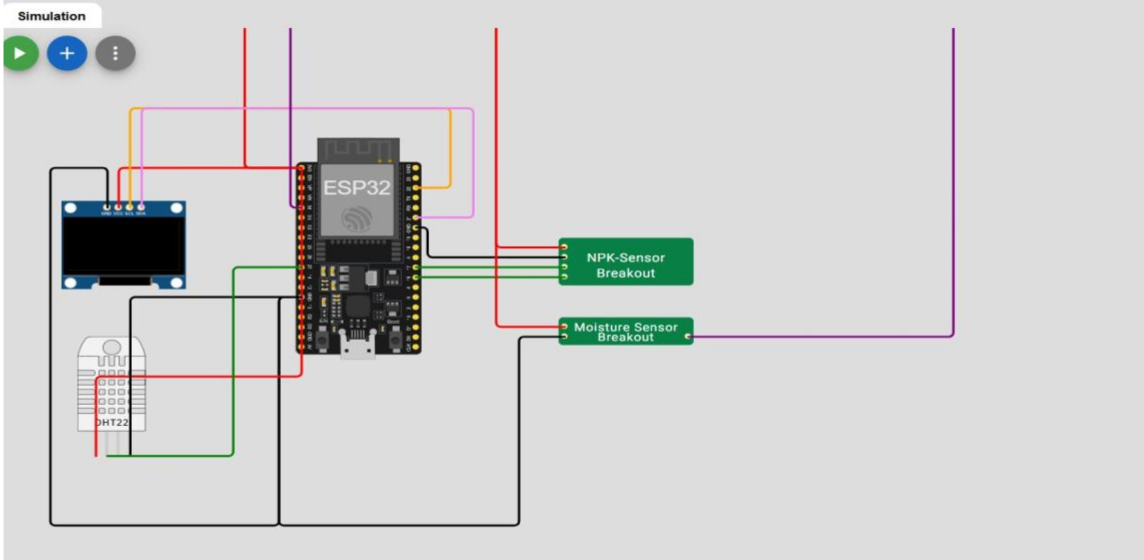
```
void displayAllSensorReadings(unsigned long duration) {
  display.println(h, 2);
  display.print("Soil Moisture: ");
  display.println(potValue);
  display.display();
  delay(duration);
}

void displayThankYou() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 20);
  display.println("Thank you!");
  display.display();
}

void sendData() {
  float t = dhtSensor.getTemperature();
  float h = dhtSensor.getHumidity();

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print("%\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" °C");
```



```
void sendData() {
  Serial.println(" °C");

  int analogValue = analogRead(pHSensorPin);

  // Convert the analog value to voltage
  float voltage = analogValue * (3.3 / 4095.0); // 3.3V reference, 12-bit ADC

  // Convert the voltage to pH value and moisture
  float pHValue = (voltage * 14.0) / 3.3; // Declare pHValue here
  potValue = analogRead(potPin);
  Serial.println("Moisture: " + String(potValue));
  Serial.print("pH Value: ");
  Serial.println(pHValue, 1);

  for (uint8_t i = 0; i < ncom; i++) {
    Serial2.print((char)commar[i]); // send the command stored in ncom array through serial2
    if (Serial2.available()) { //if serial2 data is there
      rtValue[i] = Serial2.read(); // read serial2
      Serial2.flush(); // flush serial2, very important. otherwise extra bits may interfere with communication
      Serial.print(respar[i]); // print the response array to the console.
      Serial.println(rtValue[i]); // print the return value with newline at console
    }
  }

  //send data to blynk
  Blynk.virtualWrite(V0, t);  //Temperature
  Blynk.virtualWrite(V1, h);  //Humidity
  Blynk.virtualWrite(V2, potValue); //soil Moisture
  Blynk.virtualWrite(V4, rtValue[0]); //Phosphorous
  Blynk.virtualWrite(V3, rtValue[2]); //Nitrogen
  Blynk.virtualWrite(V5, rtValue[1]); //Potassium
```



```
void sendData() {
  Blynk.virtualWrite(V5, rtValue[1]); //Potassium
}

void loop() {
  // put your main code here, to run repeatedly:
  Blynk.run();
  timer.run();
}
```


Simulation
ESP32
NPK-Sensor Breakout
Moisture Sensor Breakout
DHT22